Interactive Sketch-Based Normal Map Generation with Deep Neural Networks

Wanchao Su[1], Dong Du[2, 1], Xin Yang[3, 1], Shizhe Zhou[4, 1], and Hongbo Fu[1]
[1] City University of Hong Kong
[2] University of Science and Technology of China
[3] Dalian University of Technology
[4] Hunan University

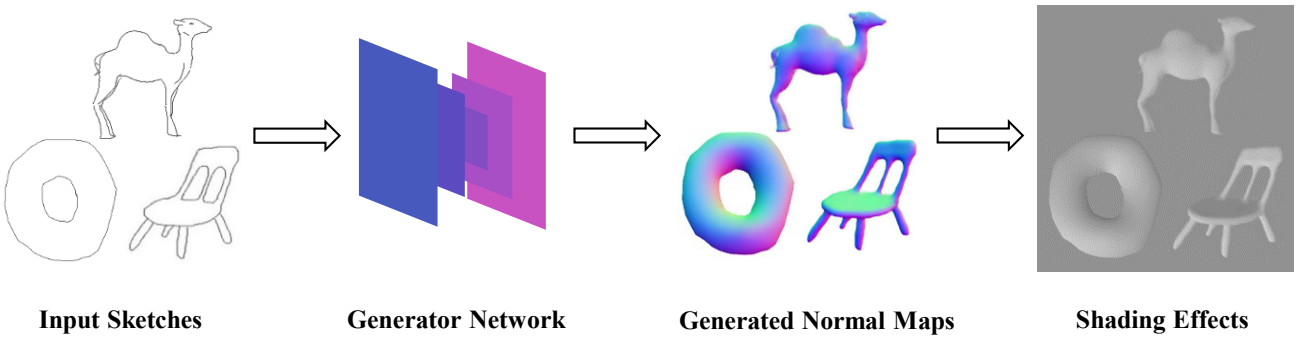

Figure 1: The overall workflow of our method. Our proposed generator network converts individual input sketches into corresponding normal maps with little or no user intervention. Here we use RGB channels to represent 3D normal components. The generated normal maps benefit various appliations such as surface relighting, texture mapping, etc. For instance here we use the normal maps for Phong shading.

ABSTRACT

High-quality normal maps are important intermediates for representing complex shapes. In this paper, we propose an interactive system for generating normal maps with the help of deep learning techniques. Utilizing the Generative Adversarial Network (GAN) framework, our method produces high quality normal maps with sketch inputs. In addition, we further enhance the interactivity of our system by incorporating user-specified normals at selected points. Our method generates high quality normal maps in real time. Through comprehensive experiments, we show the effectiveness and robustness of our method. A thorough user study indicates the normal maps generated by our method achieve a lower perceptual difference from the ground truth compared to the alternative methods.

i3D, May 2018, Montreal, Quebec, Canada

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

CCS CONCEPTS

• **Computing methodologies** → *Shape representations*; *Image representations*; • **Human-centered computing** → *User centered design*;

KEYWORDS

Sketch, Normal Map, Point Hints, Generative Adversarial Network, Wasserstein Distance

ACM Reference Format:
Wanchao Su, Dong Du, Xin Yang, Shizhe Zhou and Hongbo Fu. 2018. Interactive Sketch-Based Normal Map Generation with Deep Neural Networks. In *Proceedings of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D).* ACM, New York, NY, USA, Article 4, 10 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Normal maps take crucial parts in both academic research and commercial production. It is of great importance for many graphics applications such as shape reconstruction, surface editing, texture mapping and realistic surface rendering etc. Surface normal map is high-order differential information of the shape, and thus is not easy for human to infer accurately in the early stage of design process. Moreover, manually designing normal maps is a tedious process and often requires a lot of resource input to get high-quality results. For delicate shapes, manual methods are more problematic especially for novice designers due to the shape complexity. In commercial

production, professionals often spend hours to manually generate a detailed normal map, greatly limiting the production efficiency.

Among various representations, the sketch is an intuitive representation for designers to convey design concepts due to its diversity, flexibility, concision and efficiency. It is also a commonly adopted medium to present shapes and other geometric information. Using sketches to convey 3D information in 2D domain is a natural approach that people often employ. Since surface normals are one of the most direct ways to encode 3D information, sketch-to-normal is a predominant interpretation projecting the 2D concept to 3D space, which has been widely used in cartoon shading, digital surface modeling, gaming scene enhancement and so forth. Automatically inferring normal maps from sketches can potentially lead to a useful tool for graphics designers.

In recent years, the research community has witnessed the great capability of deep neural networks in various areas. Deep neural networks have become the common solutions behind many problems, especially for image-related conundrums. Specifically, GAN-based methods have presented a superb performance on a series of image generating problems. More concretely, for guidance-based image generation, GANs show remarkable improvements comparing with conventional deep learning methods. Since both normal information and sketch curves are well represented in the image domain, the inference from sketches to normal maps can be achieved with adoption of deep neural networks.

In this paper we present an interactive generating system, using a deep neural network framework to produce normal maps from input sketches. In our system, the sketch-to-normal map generation problem is treated as an image translation problem, utilizing a GAN-based framework to "translate" a sketch image into a normal map image. To enhance the correspondence between the input sketches and the generated normal maps, we incorporate a conditional GAN framework, which generates and discriminates images with the conditional guidance [20]. A U-net [24] architecture is applied in the Generator to pass a smooth information flow in the generating process, further increasing the pixel-wise correspondence. We employ the Wasserstein distance in our implementation to provide more effective guidance to update the network and reduce the instability of the training process.

Since a sketch is a highly simplified representation of shapes, there might be multiple shape interpretations or possible normal maps for a single input sketch. We rely on users to resolve this ambiguity problem. To achieve this, we provide a user interface so that users can give the normal information at specific points directly in the input sketch to guide normal map generation. Such an interface also enlarges the design choice of normal maps. Our system is efficient and generates the normal maps according to the input sketch and point hints in real time.

We demonstrate the effectiveness of our method on a wide range of comparisons in both quantitative and qualitative experiments. We show the superb capability of our method in generating low-error normal maps by comparing it against alternative methods, including *pix2pix* [14] and Lun et al.'s [19], evaluated on three categories of data. We validate the user interactivity by progressively increasing the point hints, and evaluate the robustness of our method by an experiment with incremental variations of input sketches. From completely new hand-drawn sketches our method can produce plausible results as well. The user studies further prove the advantages of our method in the perspective of user perception.

2 RELATED WORK

2.1 Sketch-Based Modeling

Sketch-based modeling is an active research area, and many researchers have devoted to this field and presented fruitful works [21]. Specifically, reconstructing 3D shapes or intermediates from sketches has been well studied for years but has still been drawing much attention from both computer graphics and vision researchers due to its ill-posed problem nature and the challenges (e.g., severe shape distortion of sketches especially by users without drawing training).

Traditionally, researchers defined comprehensive rules to turn 2D sketches into 3D shapes. For example, Lumo proposed by Scott F. Johnston [15] approximates lighting on 2D drawings based on silhouettes and internal contours. Wu et al. [31] proposed an interactive system to generate 3D shapes with strokes in the reference shapes. Shao et al. [27] utilized the cross-section lines to infer the 3D normal across the input sketch. Xu et al. [32] introduced an interactive method for designing a normal map from a 2D isophotes image. Sýkora et al. [28] proposed Ink-and-Ray, generating a "2.5D" intermediate for illumination renderings based on hand-drawn images using a set of annotations. Li et al. [18] proposed BendSketch, defining a set of rules to convert 2D strokes into 3D surfaces. Other methods enable the creation of surfaces by exploiting geometric constraints for specific types of line drawings, such as polyhedral scaffolds [26] and curvature flow lines [22]. Different from the above methods, we do not need explicit rules to encode the geometric information into the user input. Instead of generating an inflation-like intermediate shape, the network learns the potential rules from the sketch domain to the normal map domain directly, which preserves more geometric features and generates more complex shapes.

With the development of deep learning techniques, learning based methods have become effective tools in solving shape synthesis problems. Utilizing the powerful inference capability of the deep neural networks, many problems are solved easily. For example, Lun et al. [19] used multi-view sketches as input to generate depth maps and normal maps, and then combine them to generate a 3D model. Han et al. [10] proposed a system to encode 2D sketch lines to modify the averaged face representation for generation of detailed face models. While our method directly converts the sketches into the normal maps to represent shapes with no assistance of other geometric intermediates. Similar to the normal map generation procedure of Lun et al.'s [19], we aim at generating delicate normal maps to encode detailed geometric information from sketch inputs. However, Lun et al.'s [19] method generated normal map together with depth map and binary mask, which leads to inaccuracies compared to our single-objective normal map generation.

2.2 Image Translation

Traditional image translation effects are achieved by mechanisms based on handcrafted separate local image representation, e.g., image quilting [7], image analogies [11], image denoising [3], etc. Efros and Freeman [7] utilized a texture synthesis model for a corresponding input-output image pair. Hertzmann et al. [11] proposed image

analogies, which are based on a simple multiscale auto-regression to generate filtered images. (Deleted previous reference [9]) More advanced approaches use a dataset of input-output example pairs to learn a parametric translation function using generative adversarial networks (GAN) [9].

Mirza and Osindero [20] proposed Conditional GAN (CGAN), adding additional information to the random noise to generate images. Improved GAN by Salimans et al. [25] employed feature matching and minibatch discrimination to improve the output. Iizuka et al. [13] presented a GAN-based method for image completion, and used a Dilated Convolution implementation in their completion network. Recently, three works DualGAN [34], CycleGAN [38] and DiscoGAN [17] were published nearly at the same time, generating images in a bi-directional loop and ensuring the correspondence of images between the two representing domains. Laplacian Pyramid GAN (LAPGAN) [6] and StackGAN [35] proposed to generate images in a hierarchical manner. Using an Encoder-Decoder architecture, Wu et al. [30], Choy et al. [4] and Yan et al. [33] proposed networks to generate 3D shapes in a voxelized representation with a latent space vector encoded from real object images.

For surface normal representations, Eigen and Fergus [8] and Wang et al. [29] proposed convolutional deep networks to solve the surface normal prediction problem as a regression from a real scene photo input. Different from [8] and [29], our method generates normal maps for detailed object models instead of normal maps for coarse scenes surfaces or semantic labels for scene understanding. Isola et al. presented pix2pix [14], providing a general framework for image-to-image translation, which has a very similar objective to our work, converting images from an input representation to the a desired representation. Our method, specifically, convert the input sketches to normal maps with deep neural networks in the manner of image-to-image translation just like pix2pix [14].

3 METHOD

Both normal maps and sketches can be well-represented in 2D images. We thus treat the normal map generation problem as an image-to-image translation process. Essentially, the objective of our image-to-image translation is the distribution transformation problem in two image representation domains, so we choose a GAN framework for solving this problem in our method. In Section 3.1, we introduce the objective function in our method. Our network architecture is elaborated in Section 3.2 and illustrated in Fig. 2. The idetails of the user interaction are presented in Section 3.3.

3.1 Objective Function

While conditional GAN (CGAN) [20] maps a random noise vector z to an image $y : z \rightarrow y$, we use a conditional GAN that learns a mapping from a sketch image x and a noise vector z, to a normal map y:

$$G : x, z \rightarrow y. \tag{1}$$

Using the conditional setting, the data distributions that generator G and discriminator D try to approximate become conditional distributions [20]. In CGAN, the additional information is concatenated with the random noise vector in the input layer. As to the image translation problems, feeding G with the input image that guides the network to generate the output image is quite intuitive and it is an effective way to incorporate the sketch information in our method. Following [20], we define our objective function as follows:

$$\begin{aligned}\min_G \max_D V(D,G) = E_{x\sim p_s, y\sim p_n}[\log D(y \mid x)] \\ + E_{x\sim p_s, z\sim p_r}[\log(1 - D(G(z \mid x)))].\end{aligned} \tag{2}$$

Here, x represents the input sketch image, y is the corresponding normal map and z is the random noise vector with the same dimension as the latent space added to Generator G, and here p_s, p_n and p_r represent the sketch domain, normal map domain and random noise distribution, respectively.

In the original GAN, the objective for G is: $E_{z\sim p_r}[\log(1-D(G(z)))]$, but this function causes gradient vanishing. To solve the problem, the loss function can be modified as: $E_{z\sim p_r}[-\log(D(G(z)))]$, making G maximize the possibilities of generated samples being identified as real [9]. However, Arjovsky and Bottou [1] proved that optimizing such an objective function is equal to minimizing Kullback-Leibler divergence (KLD) while maximizing Jensen-Shannon divergence (JSD), but in fact, KLD and JSD are of the same direction.

Since JSD and KLD are incapable of measuring the input and output distributions for GAN training, we adopt the settings of WGAN [2], which uses the Wasserstein distance as the measurement of distributions to improve the objective function. The objective function in Eq. 2 then becomes:

$$\begin{aligned}L = E_{x\sim p_s, y\sim p_n}[D(y \mid x)] - E_{\tilde{y}\sim p_g}[D(\tilde{y} \mid x)] \\ - \lambda_{L1} L_{L1} - \lambda_{mask} L_{mask},\end{aligned} \tag{3}$$

where $\tilde{y}$ is the generated normal map with respect to the input sketch x from the generated normal map domain p_g. The previous methods of GANs have found that it is beneficial to mix the G loss with a traditional loss, for instance, $L1$ or $L2$ loss, etc [23]. Thus, we add another two parts, L_{L1} and L_{mask} (in Eq. 6) to further regulate the training process:

$$L_{L1} = E_{y\sim p_n, \tilde{y}\sim p_g}[\|y - \tilde{y}\|_1], \tag{4}$$

$$\tilde{y} = E_{x\sim p_s, z\sim p_r}[G(z \mid x)]. \tag{5}$$

For user-specified normals at certain points, we add a mask loss to further ensure the input information passes to the output (see more discussions in Section 3.3). Here we choose L1 loss since L2 loss encourages blurry results [23] [36]. L1 loss measures the difference between the generated image and the real image, and mask loss focuses more on the user selected pixels' distances.

3.2 Network Design

The implementation of our network structure is illustrated in Fig. 2. We first concatenate the sketch image (3-channel RGB image) with a binary point mask (indicating position of user-specified hints, 1-channel image), resulting in $H(height) \times W(width) \times 4$ dimensional input, and then feed the stacked input into the Generator (16-layer network). For discriminator (4-layer network), each layer consists of convolution, batch normalization and leak ReLU unit to process the data flow. For generator G, we adopt an Encoder-Decoder architecture [12], which is a common choice for many image-based problems, to first extract the input information and

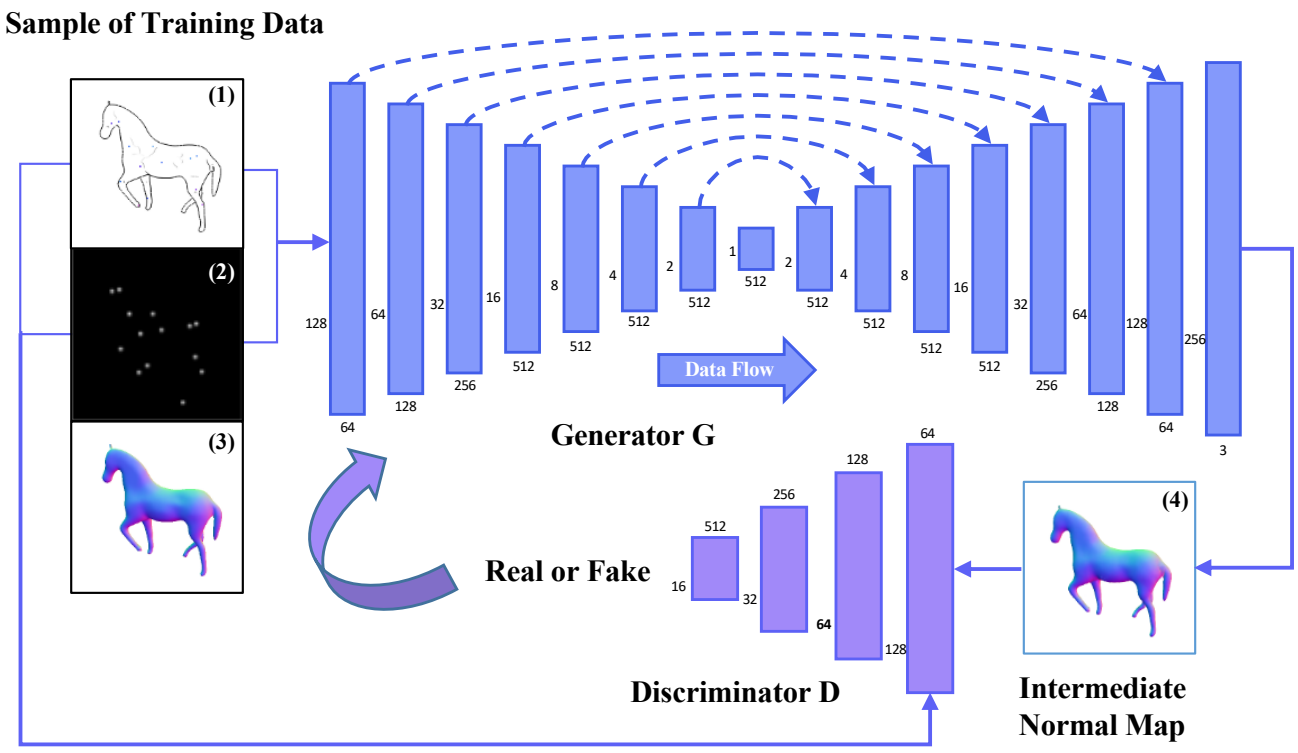

Figure 2: Network structure of our method. On the left of the figure is a sample of training data, which contains a sketch input image, a point mask and a ground truth normal map. For selected points, we set their corresponding values to 1 in the mask (2) and copy the corresponding point normals from the normal map (3) to the sketch (1). We concatenate the sketch input (1) and point mask (2) as the input of the Generator G to get the intermediate normal map (4), and then feed the intermediate normal map together with the sketch and mask as the input of Discriminator D to verify the pixel-wise "realness" of the intermediate normal map (4) compared to the ground truth (3). The discriminating information guides the Generator G to update its parameters in the training stage. In testing, sketch input and point mask are fed to Generator G only, whose output is exported as the final generated normal map. The number above or below each layer block indicates the number of layers and the number on the left of each block denotes the spatial size of the corresponding network layer.

then infer and output the normal information based on the extracted low-dimensional representation of the input.

The input passes through progressively down-sampling layers to a bottleneck layer, and after we add a random noise vector the up-sampling process begins. All the input information passes through all layers in the network. The encoder component of the generator is similar to the settings of discriminator D, while the decoder layers are composed of ReLU, deconvolution, batch normalization and dropout unit. After the encoder and decoder processing, we add a Tanh function to generate the final results. For the network training, we use the RMSProp optimizer in our implementation.

Although the low-level details differ in input and output, the high-level structure is aligned, and that is taken into consideration when we design the network. For image translation problems, it would be desirable to pass the low-level information across the layers to guide the generation [14]. Thus we incorporate the U-Net [24] in G. Specifically, connections are added after the batch normalization in the generator G between each layer i and layer $n-i$, where n is the total number of layers in G. Each skip connection simply concatenates all channels at layer i with those at layer $n-i$.

3.3 User Interaction

To incorporate user point hints, two additional losses are added together with the Wasserstein loss in Eq. 3 to further regulate the generation of the results. Users can directly assign a normal direction to a specific point in the input sketch with our user interface as shown in Fig. 3. The selected point with the normal information (assigned RGB color) is added to the sketch input, and the binary mask at the corresponding point is set to one. In every iteration of the training process, after the initial mask and the sketch are passed through the Generator, we can get a generated pre-result, and then replace the values of the masked positions with the user-specified hints in this pre-result to ensure not only the point positions but also its neighboring area present the specified normal.

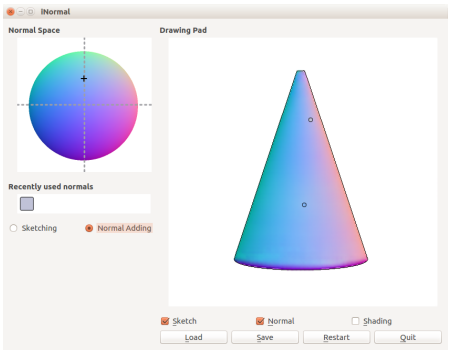

Figure 3: Our user interface. Users can select the positions in the drawing pad (right) and assign desired normals to them using the normal space (left).

We then smooth the mask with a Gaussian kernel to ensure the neighboring pixels correspond to the point hints. For the mask loss, we first differentiate the generated image and real image in L1 manner, then pixel-wisely product ($\odot$ in Eq. 6 below) the residual image with the filtered mask, and finally get the average input point loss of the generated result. The mask loss is formulated as:

$$L_{mask} = \frac{1}{\sum mask} E_{y\sim p_n, \tilde{y}\sim p_g}[\|y - \tilde{y}\|_1 \odot mask]. \quad (6)$$

To simulate the user-specified normals at certain points in preparation for the training data, we adopt the method in *ideepcolor* by Zhang et al. [37] For each image, the number of input points is generated by a geometric distribution with probability $p = \frac{1}{8}$. Each point location is sampled from a 2D Gaussian distribution with $\mu = \frac{1}{2}[H, W]^T$, $\Sigma = diag([(\frac{H}{2})^2, (\frac{W}{2})^2])$ of the normal map area (i.e., non-white pixels), where H and W are the height and width of the normal map. To ensure the generated normal maps have clear boundaries, we also draw point hints from another geometric distribution with probability $p' = \frac{1}{2}$ in the non-normal region (i.e., corresponding to white pixels). Adding the point hints not only enables interactive control of normal maps but also accelerates the network convergence procedure since the normals at the user-specified points act as additional guidance in the training stage. An example of the training data input is illustrated on the left of Fig. 2.

4 EXPERIMENTS

To get the corresponding sketch images and normal maps, we adopt suggestive contours [5] proposed by DeCarlo et al. to generate line drawings directly from a 3D model under multiple views and render its corresponding normal maps. The 3D shapes we use to generate the training and testing data in the following experiments are *Chair* and *FourLegs* from Kalogerakis et al. [16] since they have complicated geometry and structure, and are representative man-made and organic models, respectively. *FourLegs* contains 24 models (19 for training and 5 for testing) and *Chair* consists of 20 models (16 for training and 4 for testing). We first make the 3D models' orientation upright, then rotate them horizontally and evenly sample 72 views for each model in a circle change for both training and testing models. We repeat the above process with a altitude angle of 25° of the upright models to enlarge the dataset. In addition, we construct a dataset of primitive shapes, containing 50 basic 3D shapes (rotate with altitude angles of 0°, 30° and 60°, resulting in 216 views for each model since representative perception of the primitive models are in all directions) for evaluations. The images for training and testing are 256×256 pixels in size. It takes about one day to train the network (*FourLegs*: $\sim 28hrs$, > 1300 images; *Chair*: $\sim 26hrs$, > 1100 images). In the testing phase, it takes $0.025s$ to generate one normal map using our network on average.

We illustrate some results of the testing models with the same view angles as the training data in Fig. 4. We can see that our method produces results with clear boundaries and smooth normal textures. We also plot the error maps of the results compared to the ground truth normal maps. Converting to degree value, our method achieves loss at pixel-wised 1.9° on average for testing results. The largest errors often appear in the boundary and the sudden change of normal within a smooth area.

Using the *Chair* and *FourLegs* datasets we compare our method with two closely related works: *pix2pix* [14] and Lun et al.'s [19], since they have similar objectives to ours. We train the *pix2pix* [14] models with the same set of sketch-normal map image pairs and set Lun et al.'s [19] output view number to 1 to generate single view results instead of multi-view ones. In this experiment, we incorporate no additional user input for both training and testing. We compare the test results in a quantitative way: we compare the difference between the results by different methods against the ground truth using three metrics: L1, L2 distance and angular difference. The quantitative comparison results are presented in Table 1.

Table 1: Errors of different methods. We compare the generated results by pix2pix [14], Lun et al.'s method [19] and ours to the ground truth normal maps. The values here are the averaged pixel-wise differences of the normal areas in the generated images (256×256 pixels).

Dataset	Loss Type	pix2pix	Lun et al.'s	Ours
Chair	Angular	21.270°	29.765°	**11.517°**
	L1	0.235	0.307	**0.154**
	L2	0.179	0.242	**0.105**
Four Legs	Angular	36.253°	35.236°	**19.649°**
	L1	0.393	0.379	**0.254**
	L2	0.294	0.278	**0.171**

We use the mask of the ground truth to prune the background (non-normal) area out and calculate the error within the object (normal) areas, since we care about the accuracy of the normal map of a sketched object. We first normalize the output normal maps to unit length and compute the pixel-wise angular difference (in degrees) against the normalized ground truth normal maps. For the pixel-wise L1 and L2 losses, we simply calculate the differences between the corresponding normal vectors of the normal regions in the manner of L1 and L2 norms, respectively.

Our method uses the Wasserstein distance, leading to lower errors than pix2pix [14], which uses the original GAN loss. The

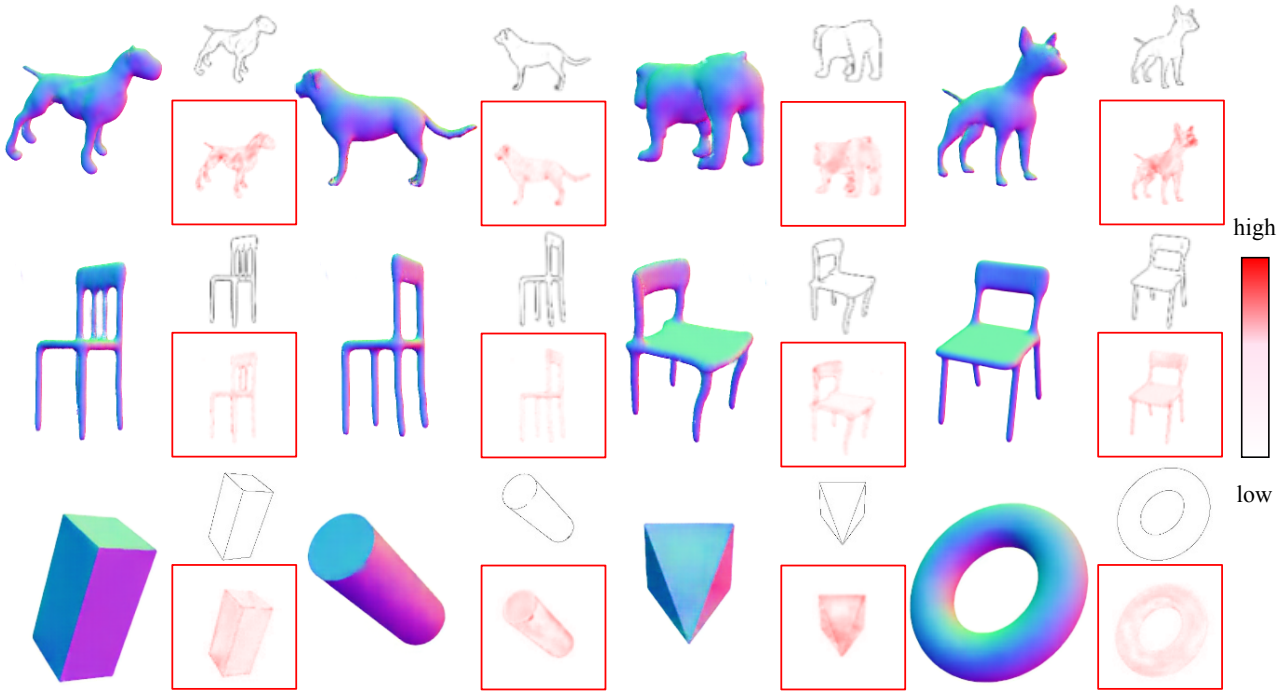

Figure 4: Examples of generated normal maps using our method. On the upper right of each group is the sketch input, and on the lower right is the corresponding error map compared to the ground truth. We visualize the angular losses of the generated normal maps, here the red channel corresponds to the error of the generated normal map, and white is zero error.

Wasserstein distance can effectively guide the network to convergence. Lun et al.'s network [19] uses L2 distance as a part of objective loss that produces blurry results, and causes larger errors than ours. In Table 1, we can see that our method outperforms the other two, and has significantly lower errors. Through paired samples t-test for individual averaged error values per image in the test dataset by three methods (same errors reported in Table 1), we confirm our method significantly (*Chair*: *pix2pix* vs. ours: $t = 13.94$, $p < 0.01$; Lun et al.'s vs. ours: $t = 23.19$, $p < 0.01$. *FourLegs*: *pix2pix* vs. ours: $t = 22.77$, $p < 0.01$; Lun et al.'s vs. ours: $t = 27.87$, $p < 0.01$.) reduces the error values in the generated normal maps.

To further show the difference between the results by different methods, we visualize the error maps of different methods in a box plot of 144 views of each testing models in the *FourLegs* dataset in Fig. 5. In addition, we also choose 3 testing error maps to illustrate the different error levels in Fig. 5. From the box plot, we can see that the errors of our method are more concentrated compared to *pix2pix* [14] and Lun et al.'s [19]. Our method achieves a more subtle difference from the ground truth. While the averaged pixel-wise error values in Table 1 are very small, from the error maps in Fig. 5, we can easily perceive visual difference between the error maps by different methods. Our method achieves better performance in generating smooth normal maps, especially for the complex boundary areas. See the supplemental materials for more generated results using different methods.

We run a validation experiment to test how our method is robust against the changes of view angle. We train the network with

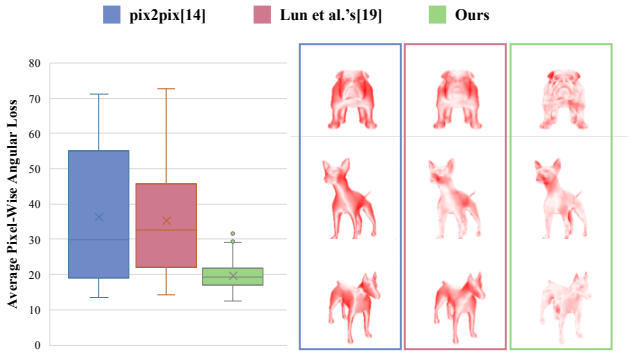

Figure 5: The visualization of average pixel-wise angular loss using different methods. The box plot shows the distribution of error values of different methods. From the test results we choose three views of different testing models as examples to visualize the losses. Error maps marked by different color rectangles are generated by the corresponding methods. The average angular losses are in degrees.

the *FourLegs* and *Chair* dataset described above. In the test stage, we experiment on the test models under different altitude angles (from 0° to 330°), and same horizontal sampling scheme as the training data. We also evaluate the cross-class generation capability of our framework by training the network using one dataset or a

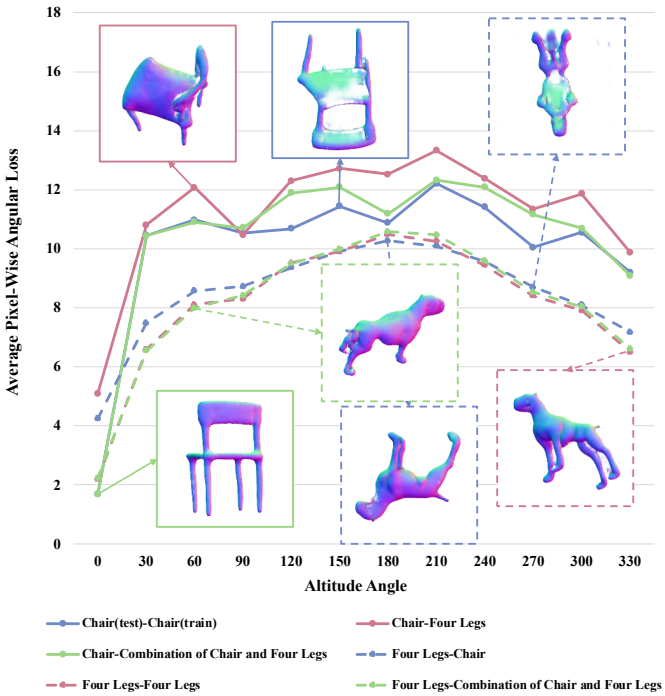

Figure 6: Evaluation against different viewpoints and networks with different training data. The solid lines correspond to the test results with Chair data and the dashed lines illustrate the test results of Four Legs data. The blue and red curves represent the results using the networks trained with the Chair and Four Legs datasets, respectively. The green lines denote the results by the network using a combination of Chair and Four Legs as the training data. The normal maps with different color frames are the testing results using the networks with specific rotation angles. The angular loss for each generated result is calculated across the whole image.

combination of the two datasets, but testing on the other dataset model. The results are illustrated in Fig. 6.

We choose two models from the *Chair* and *FourLegs* test models. Each of them are rendered with different altitude angles. From the curves in Fig. 6, it can be seen that with the increase of the altitude angles, the error values first go up, after reaching a peak, the values decrease gradually generally for all the three networks trained using different data, due to the used two altitude angles (0° and 25°) for rendering the training data and the periodic nature of angle. In other words, generally if the underlying viewpoints of the input sketches are closer to those used for rendering training data, the inferred normal maps are more accurate. It is expected that using the same category of data for training and testing leads to the best results (the red dashed curve and the blue solid curve).

In contrast, applying the network trained on the cross-class dataset to a test model causes more severe errors, though the reconstructed normal maps still roughly reflect the desired shape (i.e. the red solid line and dashed blue line in Fig. 6). It is encouraging that combing multiple categories of objects in training does not

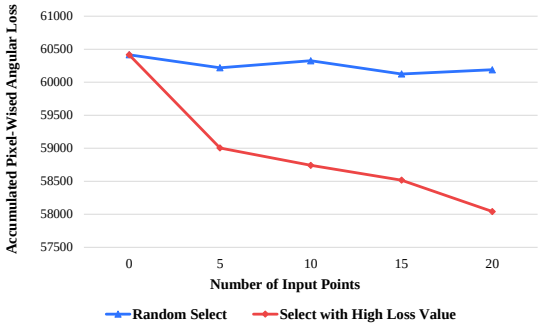

Figure 7: The evaluation of the point hints. Generally the more point hints we use the more accurate normal maps we can get. A more careful selection of point hints (at the points with the largest errors) will lead to desired results more quickly.

significantly influence the reconstruction accuracy. It can be seen from Fig. 6 that the difference between the red dashed curve and green dashed curve are nearly unnoticeable. The same trend also appears in the blue solid and red solid lines of altitude angles within [0°, 60°]. In the future we plan to train a network by using a large number of object categories as the training data so that the same trained network can be applied to sketches of different categories. Our current experiment results show that the training with multiple object categories does not require additional training time. More testing results are illustrated in the supplementary materials.

To test the effectiveness of the user-specified point hints, we train the network with a random selection scheme of point hints as mentioned in Section 3.3 and evaluate the performance in terms of the number of point hints. The evaluation results are illustrated in the curve chart in Fig. 7 for the averaged accumulated angular losses with respect to the normal areas for each normal map. The blue curve represents the results of normal maps with randomly selected points in the normal map area, and the red line indicates the results of points selected with respect to the top N (the number of input points) largest error points in the corresponding error maps. We get the error maps by comparing the ground truth normal maps to the generated normal maps from the clean sketch image (with no user-specified point hint) input. We anticipate users will specify normals at points with the large errors between the currently generated normal map and a desired normal map in their minds.

The data in Fig. 7 is generated using a test *Chair* model with 144 different views. The values in Fig. 7 are the average of the 144 views in each group, we can see that generally the loss values in both the red and blue lines go down with the increase of the numbers of point hints, which validates that our system generates better normal maps with the incorporation of user input hints. From the inclinations of the two curves we can know that selecting points with larger loss values more effectively reduce the generated errors. From the shape of the red line, we can see that the error value will remain a relatively constant low level after the large error region is eliminated.

As shown in Fig. 8 we also evaluate our tool with respect to the incremental changes of the input sketch. From the results we

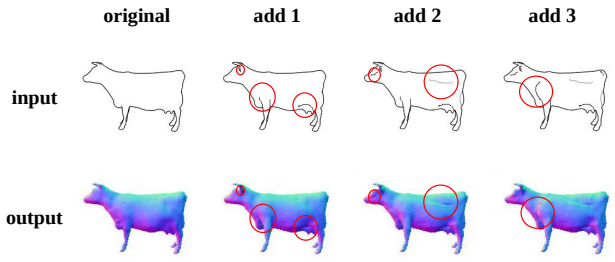

Figure 8: Incremental refinement by adding additional strokes, which lead to additional geometry details in the normal map. The modified strokes and corresponding parts in the generated normal maps are highlighted using red circles.

can see that our system responds well with the modification of the strokes. When we add an additional stroke in the sketch input, the resulting normal map shows the corresponding details. On the contrary, when we remove a stroke from the input sketch, the geometric details will be smoothed out in accordance with sketch modification. More results of the modification in sketches are presented in the supplemental materials.

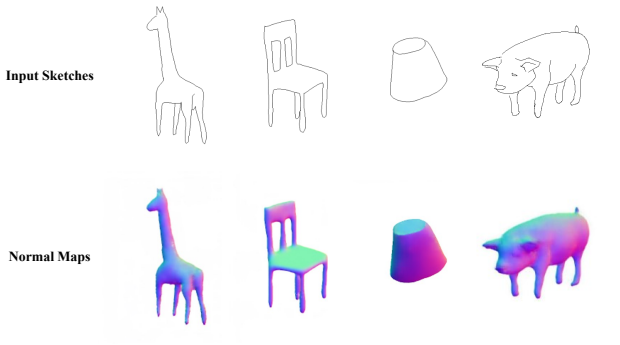

Figure 9: Test of hand-drawn sketches. Our system can infer reasonable normal maps with completely new sketch inputs.

We also test our method with completely new sketches provided by both experts with drawing skills and novices in sketching in all the three classes of datasets. Several hand-drawn sketches and corresponding generated results are shown in Fig. 9. The normal maps in Fig. 9 preserve the characteristics of the training datasets, showing smooth normal areas and strong correspondence with the input sketches. For completely new sketches, our system can provide plausible results with proper training data. For more results of the freehand drawn sketches, see the the supplemental materials.

We conduct a pilot study among 12 users regarding the user perceptual loss for the rendered normal maps generated by different methods using Phong shader. In the first task we ask the participants to choose the closest normal map given an input sketch and the

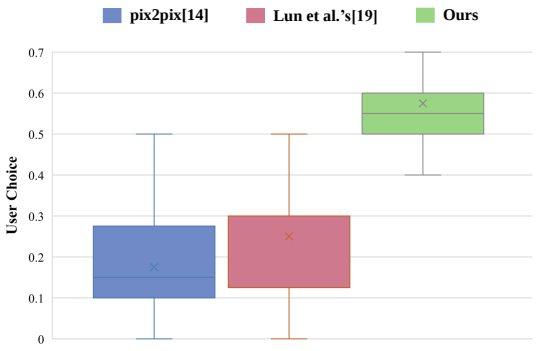

Figure 10: Plot of user preferences for closest normal map to the ground truth. The blue, red and green boxes represent the distributions of the probabilities of the user choices for pix2pix [14], Lun et al.'s [19] and our method, respectively.

ground truth as reference to test the perceptual losses of the three methods, the task contains 10 sets of randomly selected normal maps from the test models using our method, *pix2pix* [14] and Lun et al.'s [19]. The results are summarized in Fig 10.

The green box in Fig. 10 shows the highest average proportion among the three methods, achieving 57.5% in identifying the normal maps generated by our method as ones closest to the ground truth. The average proportions for pix2pix and Lun et al.'s method are 17.5% and 25.0%. The t-tests (Ours vs. *pix2pix* [14]: $t = 7.22$, $p < 0.01$; Ours vs. Lun et al.'s [19]: $t = 5.96$, $p < 0.01$) proves that our method is statistically better than the other two methods in generating perceptually lower-error normal maps.

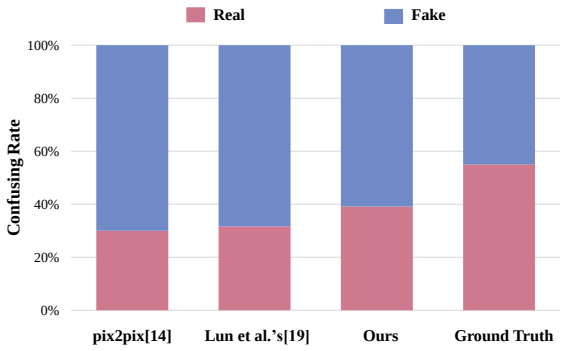

Figure 11: Confusing rate for normal maps generated by 4 different methods. The red part in each bar represents the proportion of normal maps identified as real using a specific method, and the blue part denotes the fake rate.

To investigate the accuracy of user perception in interpreting the normal maps, we do another user experiment of normal map confusing rate. In this experiment, we render 10 normal maps randomly selected in the test set for *FourLegs* with 4 methods: *pix2pix* [14], Lun et al.'s [19], ours and the ground truth. We request the users to label if a given normal map is real or fake in a random order. The test results are plotted in Fig. 11.

The average confusing rate (ratio of normal map identified as real) of our method is 39.17% compared to 55% of the ground truth. The confusing rates for pix2pix [14] and Lun et al.'s [19] are 30% and 31.7%, respectively. Our method achieves slightly higher confusing rate than the other two methods but still lower than the ground truth. In addition, the participants in this task give a difficulty factor of 7.08 out of 9 (1 means easiest and 9 means hardest), indicating the efforts required in this task are quite high and people find difficult in interpreting the rendered normal maps directly. See the individual user results and testing images in the supplemental materials.

5 DISCUSSIONS

In sum, we have presented an interactive method for normal map generation. The implementation of conditional GAN framework encodes sketch input to the latent representation and decodes the latent space representation into normal map. By adopting the U-Net architecture, our network presents a smooth data transition. The incorporation of the Wasserstein distance provides precise gradient information for the network training process. User-specified point hints allow more direct control of the normal map and effectively eliminate the ambiguity of the sketch representation. The effectiveness of our technique has been demonstrated by qualitative and qualitative experiments. Our method outperforms the alternative solutions based on *pix2pix* [14] and Lun et al.'s [19].

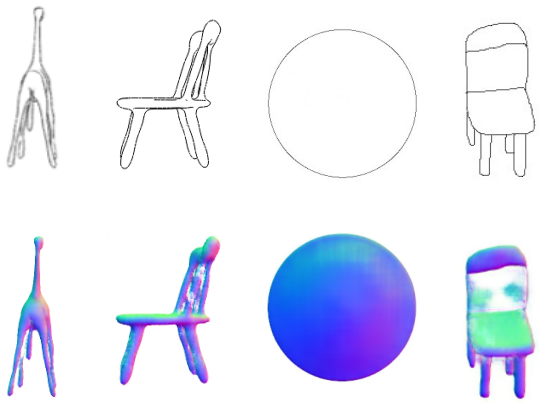

Figure 12: Less successful cases with corresponding sketches. There are some generated results with obvious visual artifacts: one of the legs of giraffe is disconnected, the back area of the chair is messy, the surface normal of the sphere is not evenly distributed, and in the back region of the hand drawn chair.

Despite the good performance in many cases, our method does not perform well in certain cases. In Fig. 12, we can see that our method produces obvious visual artifacts in complex sketch areas and delicate areas. There are also generated artifacts with patches outside the drawn sketch containing spuriously predicted normals. This is a common phenomenon for GAN-based methods and may be solved by adding more user input hints. For the large area of uneven normals, we can also consider utilizing some post-processing techniques to reduce the artifacts. Furthermore, the testing results are highly dependent on training data, providing relatively worse results for cross-class generation as discussed previously. For the user-specified point hints, since the normal maps are highly structured representations of 3D shapes, users cannot arbitrarily assign a point normal far from the potential candidates.

One of the future directions might be to explore the GAN training techniques to improve the capability of our current networks, reducing the artifacts in the generated results. Since the current method can only generate smooth normal maps, another potential direction can be the post processing technique to add more details in the generated normal maps according to the user design. Our current user interface enables user-specified point hints, it might be interesting to incorporate the stroke-based normal assignation for more compact user interaction.

ACKNOWLEDGEMENT

We thank the anonymous reviewers for their suggestion. This work was partially funded by the RGC of Hong Kong SAR (Project No. CityU11204014, CityU11200418), National Natural Science Foundation of China (Project No. 91748104, 61303147), Science Foundation of Hunan Province (Project No.2018JJ3064), the Hong Kong Scholars Program, Ministry of Education of China HPCSIP Key Laboratory and supported by NVIDIA Corporation with the donation of the Titan Xp GPU.

REFERENCES

[1] M. Arjovsky and L. Bottou. 2017. Towards Principled Methods for Training Generative Adversarial Networks. *ArXiv e-prints* (Jan. 2017). arXiv:1701.04862
[2] M. Arjovsky, S. Chintala, and L. Bottou. 2017. Wasserstein GAN. *ArXiv e-prints* (Jan. 2017). arXiv:1701.07875
[3] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. 2005. A Non-Local Algorithm for Image Denoising. In *CVPR '05*. 60–65.
[4] Christopher B. Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 2016. 3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction. In *ECCV '16*. 628–644.
[5] Doug DeCarlo, Adam Finkelstein, Szymon Rusinkiewicz, and Anthony Santella. 2003. Suggestive Contours for Conveying Shape. *ACM Trans. Graph.* 22, 3 (July 2003), 848–855.
[6] Emily L Denton, Soumith Chintala, arthur szlam, and Rob Fergus. 2015. Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks. In *Advances in Neural Information Processing Systems 28*. 1486–1494.
[7] Alexei A. Efros and William T. Freeman. 2001. Image Quilting for Texture Synthesis and Transfer. In *SIGGRAPH '01*. 341–346.
[8] David Eigen and Rob Fergus. 2015. Predicting Depth, Surface Normals and Semantic Labels With a Common Multi-Scale Convolutional Architecture. In *ICCV '15*.
[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Eds.). 2672–2680.
[10] X. Han, C. Gao, and Y. Yu. 2017. DeepSketch2Face: A Deep Learning Based Sketching System for 3D Face and Caricature Modeling. *ACM Trans. Graph.* 36, 4 (July 2017).
[11] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. 2001. Image Analogies. In *SIGGRAPH '01*. 327–340.
[12] G. E. Hinton and R. R. Salakhutdinov. 2006. Reducing the Dimensionality of Data with Neural Networks. *Science* 313, 5786 (2006), 504–507.
[13] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. 2017. Globally and Locally Consistent Image Completion. *ACM Trans. Graph.* 36, 4 (July 2017), 107:1–107:14.
[14] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-Image Translation with Conditional Adversarial Networks. In *CVPR '17*. 5967–5976.
[15] Scott F. Johnston. 2002. Lumo: Illumination for Cel Animation. In *NPAR '02*. 45–52.

[16] Evangelos Kalogerakis, Aaron Hertzmann, and Karan Singh. 2010. Learning 3D Mesh Segmentation and Labeling. *ACM Trans. Graph.* 29, 4 (July 2010), 102:1–102:12.

[17] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim. 2017. Learning to Discover Cross-Domain Relations with Generative Adversarial Networks. *ArXiv e-prints* (March 2017). arXiv:1703.05192

[18] Changjian Li, Hao Pan, Yang Liu, Xin Tong, Alla Sheffer, and Wenping Wang. 2017. BendSketch: Modeling Freeform Surfaces Through 2D Sketching. *ACM Trans. Graph.* 36, 4 (July 2017), 125:1–125:14.

[19] Zhaoliang Lun, Matheus Gadelha, Evangelos Kalogerakis, Subhransu Maji, and Rui Wang. 2017. 3D Shape Reconstruction from Sketches via Multi-view Convolutional Networks. *CoRR* abs/1707.06375 (2017). arXiv:1707.06375

[20] M. Mirza and S. Osindero. 2014. Conditional Generative Adversarial Nets. *ArXiv e-prints* (nov 2014). arXiv:1411.1784

[21] Luke Olsen, Faramarz F. Samavati, Mario Costa Sousa, and Joaquim A. Jorge. 2009. Sketch-based modeling: A survey. *Computers & Graphics* 33, 1 (2009), 85 – 103.

[22] Hao Pan, Yang Liu, Alla Sheffer, Nicholas Vining, Chang-Jian Li, and Wenping Wang. 2015. Flow Aligned Surfacing of Curve Networks. *ACM Trans. Graph.* 34, 4 (July 2015), 127:1–127:10.

[23] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. 2016. Context Encoders: Feature Learning by Inpainting. In *CVPR '16*. 2536–2544.

[24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *MICCAI '15*. 234–241.

[25] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. 2016. Improved Techniques for Training GANs. *ArXiv e-prints* (June 2016). arXiv:1606.03498

[26] Ryan Schmidt, Azam Khan, Karan Singh, and Gord Kurtenbach. 2009. Analytic Drawing of 3D Scaffolds. *ACM Trans. Graph.* 28, 5 (Dec. 2009), 149:1–149:10.

[27] Cloud Shao, Adrien Bousseau, Alla Sheffer, and Karan Singh. 2012. CrossShade: Shading Concept Sketches Using Cross-section Curves. *ACM Trans. Graph.* 31, 4 (July 2012), 45:1–45:11.

[28] Daniel Sýkora, Ladislav Kavan, Martin Čadík, Ondřej Jamriška, Alec Jacobson, Brian Whited, Maryann Simmons, and Olga Sorkine-Hornung. 2014. Ink-and-ray: Bas-relief Meshes for Adding Global Illumination Effects to Hand-drawn Characters. *ACM Trans. Graph.* 33, 2 (April 2014), 16:1–16:15.

[29] Xiaolong Wang, David Fouhey, and Abhinav Gupta. 2015. Designing Deep Networks for Surface Normal Estimation. In *CVPR '15*.

[30] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. 2016. Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling. In *NIPS '16*. 82–90.

[31] Tai-Pang Wu, Chi-Keung Tang, Michael S. Brown, and Heung-Yeung Shum. 2007. ShapePalettes: interactive normal transfer via sketching. In *SIGGRAPH '07*. 44.

[32] Q. Xu, Y. Gingold, and K. Singh. 2015. Inverse Toon Shading: Interactive Normal Field Modeling with Isophotes. In *SBIM '15*. 15–25.

[33] Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. 2016. Perspective Transformer Nets: Learning Single-View 3D Object Reconstruction without 3D Supervision. (12 2016).

[34] Z. Yi, H. Zhang, P. Tan, and M. Gong. 2017. DualGAN: Unsupervised Dual Learning for Image-to-Image Translation. *ArXiv e-prints* (April 2017). arXiv:1704.02510

[35] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas. 2016. StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks. *ArXiv e-prints* (Dec. 2016). arXiv:1612.03242

[36] Richard Zhang, Phillip Isola, and Alexei A. Efros. 2016. Colorful Image Colorization. In *ECCV '16*. 649–666.

[37] Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S Lin, Tianhe Yu, and Alexei A Efros. 2017. Real-Time User-Guided Image Colorization with Learned Deep Priors. *ACM Trans. Graph.* 9, 4 (2017).

[38] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. 2017. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *ArXiv e-prints* (March 2017). arXiv:1703.10593

APPENDIX

To illustrate the qualitative comparison of the generated results by different methods, we add some of the generated normal maps in Fig. 13. We add the complete comparison result in the supplemental materials.

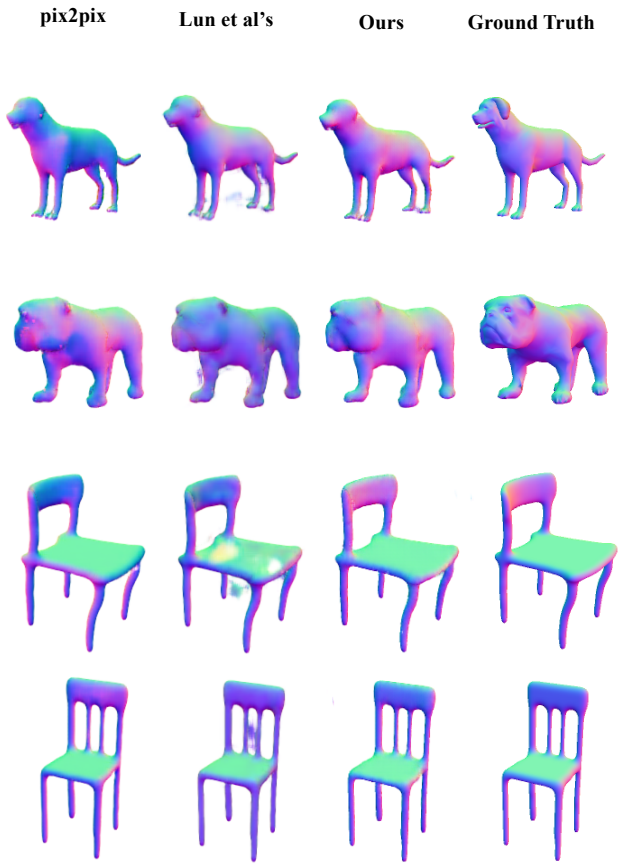

Figure 13: Side-by-Side comparison of results by different methods. For the complete set of testing results, please consult the supplemental materials.