DeepShapeSketch : Generating hand drawing sketches from 3D objects

Meijuan Ye[1] , Shizhe Zhou[1,2,*] , Hongbo Fu[3]
[1]*College of Computer Science and Electronic Engineering, Hunan University, Changsha, China*
[2]*Key Laboratory of High Performance Computing and Stochastic Information Processing, Ministry of Education of China*
[3]*City University of Hong Kong, Hong Kong, China*
mj@hnu.edu.cn, shizhe@hnu.edu.cn(corresponding author), hongbofu@cityu.edu.hk

***Abstract*—Freehand sketches are an important medium for expressing and communicating ideas. However creating a meaningful and understandable sketch drawing is not always an easy task especially for unskillful users. Existing methods for rendering 3D shape into line drawings such as Suggestive Contours, only consider the geometry-dependent and view-dependent information thus leads to over-regular or over-perfect results which doesn't look like a human freehand drawing. For this challenge we address the problem of producing freehand line drawing sketches from a 3D object under a given viewpoint automatically. The core solution here is a recurrent generative deep neural network, which learns a functional mapping from the suggestive contours of a 3D shape to a more abstract sketch representation. We drop the encoder of the generator, i.e., use only a decoder to achieve better stability of the sketch structure. Users can tune the level of freehand style of the generated sketches by changing a single parameter. Experiments show that our results are expressive enough to faithfully describe the input shape and at the same time be with the style of freehand drawings created by a real human. We also perform a comparative user study to verify the quality and style of generated sketch results over existing methods. We also retrain our network using several different mingled dataset to test the extendibility of our method for this particular application. As far as our knowledge this work is the first research effort to automate the generation of human-like freehand sketches directly from 3D shapes.**

***Index Terms*—Freehand sketches, Generative recurrent neural network, 3D object**

I. Introduction

Sketching is a common art form used in communication, rendering, and exchanging ideas, especially for expressing the overall shape of a certain 3D object. People in ancient times have already created line arts such as animal figures or buildings on cave rock on shells as a record of their lives. This is basically because we are able to easily envision the actual 3D shape when we observe its corresponding sketch images.

However, not every human being is well-trained for sketching or drawing. It is generally not very easy for an amateur artist to create an accurate sketch according to a given 3D object. Furthermore, the sophisticated skills used by professional artists for creating cartoons or mangas are not quite suitable for daily utility, since usually they take a lot of time to finish. It is more natural to use simple strokes to create relatively abstract patterns to convey shape-wise information or even express new design ideas. However we do not inherently have the ability to draw using just a few sketch lines to correctly describe a 3D shape. On the other hand, existing methods for directly rendering 3D shape into line drawing such as Suggestive Contours [1], only considers the geometry-dependent and view-dependent information thus leads to over-regular or over-perfect results which doesn't look like a human freehand drawing. The straightforward solution of extracting the edge map of a simple projection from 3D models also suffer from the same problem. Furthermore, such edge map results are often uncontinuous, difficult for vectorization.

To address this issue we propose a new approach to translating three-dimensional objects into freehand human sketches. To sketch the shape of a 3D object is not a simple perspective or orthogonal projection of the 3D object on the image domain. A good shape sketch contains not only the outline of the object but also the lines of the fold detail that can reflect the internal features of the object. This requires the 3D object to be converted into a 2D view with distinct features. The major idea of our approach here is to take the 3D object as input and generate sketches that are quite similar to human hand-drawings through neural network learning. We first build a GMM distribution of sketch line shapes via supervised training a Recurrent Generative Adversarial Neural Network which introduces certain levels of randomness or minor deformation to the sketch lines, and then constrain the overall orientation of the sketch object with a novel sampling method after training the neural network.

Variational Autoencoders (VAEs) [2], [3] and Generative Adversarial Networks (GANs) [4]–[6] are important deep learning models in recent years. Sketch-RNN [7], an existing deep learning method for sketch generation, used the sequence-to-sequence variational-auto-encoder (VAE). However, one drawback of VAE is that the resulting samples are usually blurred due to injected noise and imperfect elemental measurements such as squared errors. Therefore, we remove the encoder and add a discriminative network based on Sketch-RNN [7]. The generative network and the discriminative network together form our generative recurrent adversarial networks. In each training session the generative network produce data simulating an unknown distribution, and the discriminative counterpart is trained to distinguish between the generated *fake* data and the samples from the genuine observations, so as to eventually modify the generator to maximize confusing probability against the discriminator.

The main procedures of our method are listed below: We first convey the shape of 3D objects into line-drawing forms using a non-photorealistic rendering system. Here we use DeCarlo et al.'s method: Suggestive contours [1]. We render the suggestive contours of each object from many different viewing directions to bulk-produce a two-dimensional line drawing sketch image dataset. Then a vectorization process is performed on the sketch images to obtain two-dimensional coordinate information. Based on the Sketch-RNN model [7], we remove the encoder and add the discriminative network composed of a recurrent neural network. Finally a novel two-phase sampling process to construct human-like freehand vectorized sketches from the above neural network.

Our work contains contributions from 4 aspects:

- As far as our best knowledge, the problem of generating freehand vectorized sketches from a 3D object at a given viewpoint is addressed for the first time using a deep learning method, which enables cross-domain stroke-level visual understanding of a reference 3D object.
- A supervised uncycled translation network is applied to train our model. Due to the absence of the ground truth of our problem, a generative adversarial neural network is applied to verify if the generated sketch has a good balance between structure soundness and the style of an amateur human creator.
- A novel two-phase sampling process to guarantee more correct orientation of the result sketch.
- A middle-size dataset of 3D model snapshot photo and its corresponding vectorized sketches for future study on data-driven sketch generation.

II. Related Work

In essence our problem here is an application of deep generative neural network onto the vectorized line depiction of 3d Shapes. Many research works are related to this problem and we categorize these works into 3 areas listed below.

A. Deep Generative Models

The recent 5 years have witnessed many developments of deep generative models. There are generally three themes: VAE [2], [3], GAN [4]–[6] and other neural networks based on GAN [8]–[11]. VAE pairs a programmable network with a decoder/generative network. The Sketch-RNN model of Ha and Eck [7] also used VAE. However, VAE has a disadvantage in which the generated samples are usually ambiguous due to imperfect element measurements,e.g.,the squared error and injection noise. GAN is another very popular generative model which simultaneously trains two modules: a generative subnetwork,i.e.,generator, to synthesize data, and a discriminative subnetwork,i.e.,discriminator to differentiate between synthesized and genuine data. However, the GAN is relatively slow to converge and its generated sample data can be unsatisfactory. The quality of the generated samples can be improved by modifying its structure and loss functions,e.g.,Wasserstein GAN (WGAN) [12], McGAN [13], and Loss-Sensitive GAN [14]. There are also methods that attempt to incorporate GAN and VAE, such as VAE/GAN [15], CVAE-GAN [10]. Another type called Recurrent neural networks(RNNs) are proposed for simulating sequential data, and there are many works to combine RNN with GAN, such as C-RNN-GAN [9] and S-LSTM-GAN [16].

B. Recurrent Vector Image Generation

There is relatively little work done using neural networks to generate vector sketch images currently. Earlier, there was a work [17] that apply Hidden Markov Models(HMM) to synthesize human sketch lines. The method of Alex Graves [18] laid the foundation for generating continuous data points using a hybrid density network [19]. They showed how to use the recurrent Long Short-term Memory network to generate complex sequences with long-distance structures to approximately generate vectorized hand-written numbers. Similar networks were proposed for vectorized Kanji characters and hand drawing sketches, unconditionally or conditionally modeling the sketches as stroke actions sequences. Sketch-RNN (a kind of recurrent neural network that can construct ordinary objects based on strokes) proposed by Ha and Eck [7] outlines a strategy for generation of line art vector graphics both conditionally and unconditionally.Their proposed apporach based on a recurrent neural network can generate sketches of common objects in vector format from input images. Chen et al. [20] made two changes on the basis of Sketch-RNN: One was to use a convolutional neural network (CNN) instead of the bidirectional RNN encoder. The other was to remove the Kullback-Leibler (KL) divergence from VAE objective loss function. It had better performance than [7] in learning and generating multiple categories of sketches. A recent work proposed by Muhammad et al. [21] used reinforcement learning to generate sketches with different abstraction level from images. Song et al. [22] used a deep learning model to achieve stroke-level cross-domain visual understanding. The problem of noisy supervision caused by diversified and subjective human drawing styles was addressed by developing a new mixed-supervised-unsupervised multi-task learning framework. With a novel shortcut cycle consistency constraint the unsupervised learning was implemented more effectively.

C. Lines to Depict the Shape of 3D objects

The goal of line rendering is to capture key features on the surface of an object and use lines to represent the shape of the object. Rendering methods are generally categorized into two categories: view-independent and view-dependent. For view-independent rendering methods, the main research focuses on the extraction of ridges and valleys. Such lines do not take into account the viewing angle and are determined only by analyzing the geometric properties of the surface of the object; the view-dependent rendering methods must determine the viewing angle firstly, and the resulting line is related to the choice of viewing angle. The most commonly used is object contours, which are lines formed by the point perpendicular to the line of sight [23]. The representative method is the Suggestion Contours method developed by DeCarlo et al. [1].

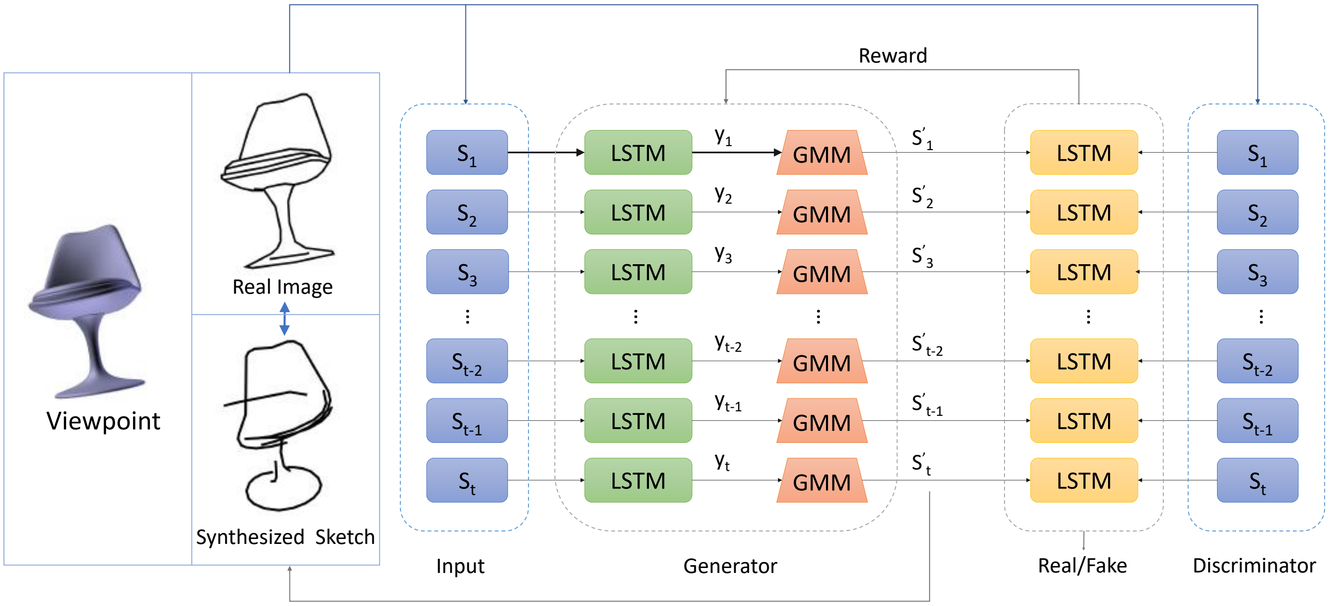

Fig. 1. Our network structure for generating sketches. See Section III-B for details.

This method was used by Sang et al. [24] and Yoon et al. [25] for several related applications such as retrieval. And it has also been used for many other applications. In addition, Judd et al. [26] proposed the Apparent Ridges method, which differed from the Suggestive Contours in which it projected the normal vector of points on the surface onto the viewing plane and calculated the points on the curved surface along the various directions in the viewing plane. According to Cole's [27], [28] comparative analysis of existing three-dimensional model line rendering algorithms, the Suggestive Contours method can obtain better model feature line segments. Therefore, this paper chooses the Suggestive Contours method to obtain the visual features of the object.

III. Methodology

In the methodology section we will introduce the preparation of our dataset, followed by describing our network structure in detail. In addition, we will describe the method of generating multi-view sketches.

A. Training Data

1) Line drawings of 3D objects: Because our network should be able to translate surface and shape information out of a given 3D object at any view direction to human-like sketches, we need to construct a training dataset which includes "snapshots" of a 3D shape from all possible angles. Furthermore, since the final output is line drawing sketches , we also need to make a linear representation of these "snapshot" images.

At the beginning, we carefully considered the dataset used by Sketch-RNN [7] and sketch-pix2seq [20], from QuickDraw [29], a public sketch database published by Google. However, almost all the sketches in the QuickDraw dataset are taken from a frontal perspective that do not meet our multi-view requirements. Thus we need to synthetically construct a new dataset specifically designed for our problem.

Considering that real human freehand sketches are often casual and spontaneously expressive, it is reasonable to make such a linear representation including not just the outer contours of the object but also the internal detail lines. Using the Suggestive Contours method [1], we can get a variety of perspective views of a 3D object, which constitute the initial form of our dataset. Specifically, we place the normalized 3D object in the central point and then rotate it randomly and uniformly by 2900 times. For each view we capture it into a 256×256 image. Generally it takes less than 20 minutes to finish capturing all images. Among all the images, we randomly select 2300 images for training, 300 images for testing and the rest 300 image for validation.

2) Vectorization and simplification: Next we use the Wintopo software [30] to vectorize all the sketch images into actual vector lines represented as a sequence of vertices (i.e., point coordinate data) and edges (i.e., connectivity information).

Because the above linear representation has a rich set of internal lines, the number of line segments is overlarge, making the subsequent training difficult. Therefore, before vectorization we first filter the line segments and remove short ones (length below 3 pixels). Here we also apply the Ramer-Douglas-Peucke method [31] to do stroke simplification. This method has a controlling parameter ε for deciding whether to discard a certain input point, depending on how aggressively the user wants to simplify the lines. In all of our experiment, we set the parameter ε to be 0.1.

Next we reform all the lines into the same data format used in QuickDraw [29], which sequentially represents each vector line as a list of coordinate offsets: $\Delta x, \Delta y$ and the binary value p indicating whether the pen is lifted away from the paper. This format is denoted as **stroke-3**. Before inputting these line data into our network, we again transform all the **stroke-3** lines into **stroke-5** format, i.e., $(\Delta x, \Delta y, p_1, p_2, p_3)$. The first two elements are the offset distances of the pen in the x and y directions from the previous point respectively. The last 3 elements represent three states of possible strokes: the first state p_1 indicates that the pen is now contacting the paper and will draw a sketch line in succession; the second state p_2 describes that the pen is going to be lifted away from the paper immediately after the current point, and the drawing of the current line will not continue; and the last pen state p_3 indicates that the drawing has finally ended.

B. Network Structure

The Sketch-RNN [7] model whose encoder is a bidirectional RNN that takes a sketch as an input and outputs a latent vector, is a Sequence-to-Sequence Variational Autoencoder (VAE) network. Its decoder is an autoregressive RNN sampling output sketches conditionally on a given latent vector. In our approach, we drop the encoder based on Sketch-RNN [7]. The reasons for this modification are as follows.

Firstly, as discussed before, VAE has the disadvantage that the resulting samples are usually blurred due to injected noise and imperfect elemental measurements such as squared errors. Secondly, as described in sketch-pix2seq [20], although the bidirectional RNN encoder is used to reduce the data dimension from a two-dimensional image to a one-dimensional sequence, additional information unrelated to the abstract structure of the sketches, e.g., the sketching speed, might affect the performance of concept learning. Meanwhile we remove the KL divergence from the objective function because the bidirectional RNN encoder has been removed and it is not necessary to specify a reasonable prior.

In addition to the above modifications, we also add a discriminative network after the decoder to further guarantee that the generated sequences of data are maximally similar to the real sequence of inputs. The discriminator $\mathbb{D}$ consists of a RNN because all input data are sequential. In this situation, the part of the decoder and GMM is equivalent to a generative network, and the whole framework becomes a GAN. So the network we construct is a RNN with adversarial training. According to the description of [4], the generator G is iteratively trained to generate data as indistinguishable as possible from the genuine data, and the discriminator $\mathbb{D}$ is constructed to verify the generated sample. The architecture of our network is illustrated in Figure 1: the Real image is obtained from a given view direction of the 3D object using suggestive contours [1]. The S_t is the input sequence generated from the Real image, and y_t is the data generated from the decoder. $S_t^{'}$ is the sampling result at that time step. A sequence of generated points $S^{'}$ forms the Fake sketch.Index t is the time step. The part of the decoder, i.e., the green LSTM and GMM (Gaussian mixture model) are equivalent to the generator (G) for generating the sequence data, and the discriminator ($\mathbb{D}$) is trained to verify whether the generated sample is acceptable enough. Here we define the objective function as follows:

$$\min_{G} \max_{\mathbb{D}} E_{x \sim p_r}[\mathbb{D}(x)] - E_{\tilde{x} \sim p_g}[\mathbb{D}(\tilde{x})] \tag{1}$$

where x is the real sequence from the training data and $\tilde{x}$ is the sequence generated by G. p_r and p_g are data distribution and model distribution produced by the generator G, respectively.

C. Two-phase Sampling

We now describe in detail our sampling process we specifically design for generating a complete line draw sketch from the trained GAN. Let us denote the total number of points on a line draw sketch as Θ. Our sampling can be regarded as a two-phase process. For the first phase, we compute the first N points. For the second phase, the rest $\Theta - N$ points are generated.

Specifically, in the first phase we directly copy the **stroke-5** points S_t from the input vectorized suggestive contour sketch as the first N points of the output. Whereas, for the second phase, we iteratively feed the output of our GAN at time step $t-1$, i.e, $S_{t-1}^{'}$ to be the input of GAN at time step t to generate $S_t^{'}$, similar to Sketch-RNN [7]. In other words, the GAN recursively generates parameters for categorical and GMM distribution during each timestep.

The effect of the first phase is to constrain the overall direction of the final result sketch to be as close as possible to the user intended viewpoint. It is natural to imagine these first N points as "nails" or anchor points, which "pin", i.e., fix not only themselves but also all rest $\Theta - N$ point to form the correct sketch conforming to the given viewpoint. Ideally in geometry, 3 non-collinear points determines a Cartesian system. We double-check the sketch vectorization to confirm that not consecutive 3 points are co-linear. So we set $N = 3$, i.e., we use the first 3 points as anchors in this paper.

Our aforementioned assumptions are confirmed by our experiments. E.g., discarding the first phase, i.e., setting $N = 0$, always leads to entirely wrong orientation of the sketch, see the two "$N = 0$" rows in Figure 3. Only 2 anchor points can not constrain the orientation either, since the GAN can still occasionally generate wrong sketches that are reflective symmetric to the required view direction, see such a highlighted case in the red circled of Figure 3. On the contrary, $N = 3$ guides the sampling process to generate correct orientation, see Figure 3.

IV. Experiments

According to the description of Sketch-RNN [7], when an encoder is incorporated, it is called conditional generation, and conversely, it is unconditional when the encoder is not used. We perform several experiments on conditional and unconditional vector image generation using the network with or without the discriminator $\mathbb{D}$, i.e., we compare the results of our modified network with their corresponding results of the original Sketch-RNN model, see Table I. As shown in Figure 2, the comparison results visually demonstrate that

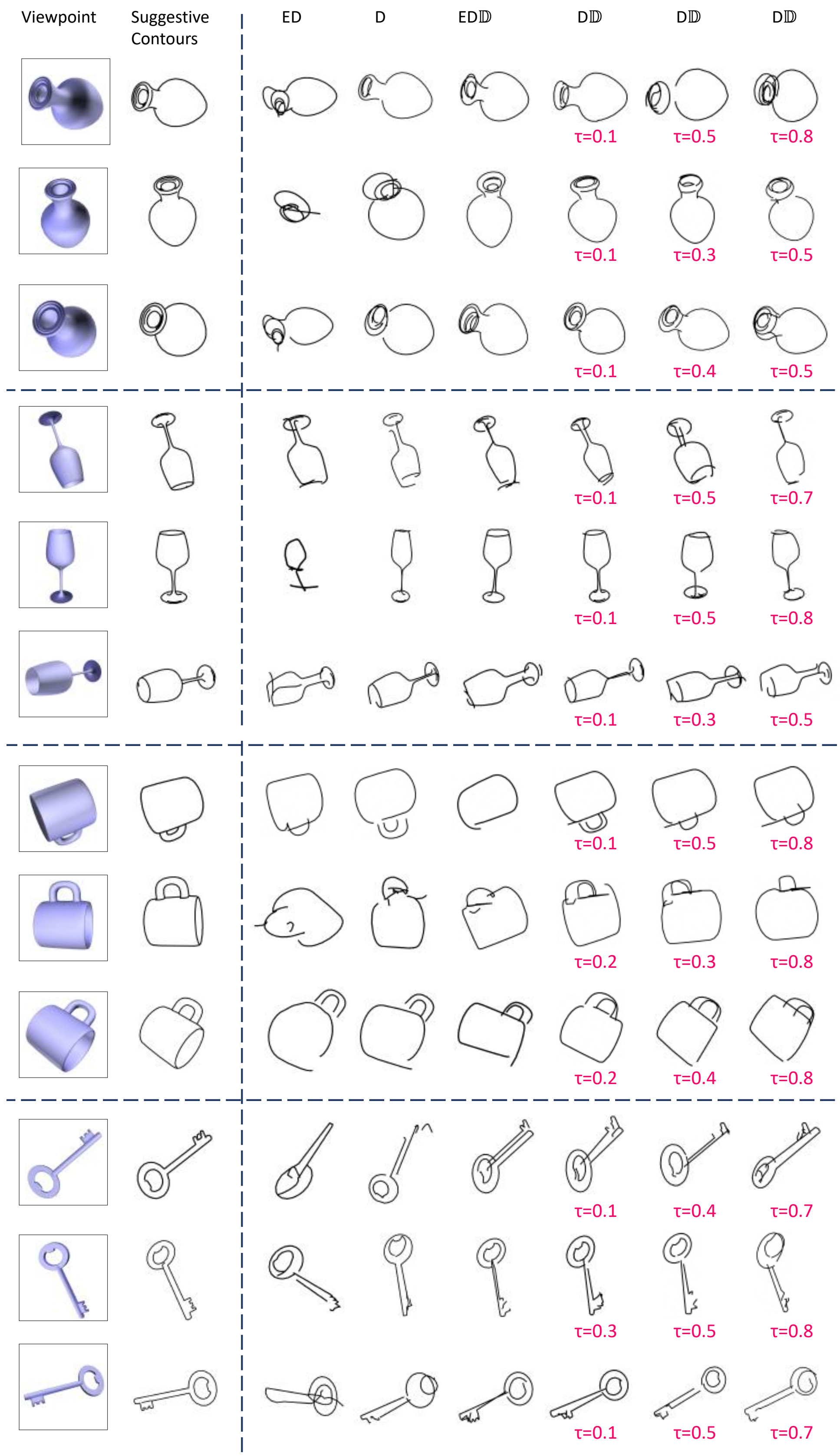

Fig. 2. Comparison of the 4 generative models on Table I. From left to right: 3D rendering, corresponding suggestive contours [1], and sketch results of ED, D, ED𝔻, D𝔻 with 3 different τ value.

TABLE I

MODELS TRAINED FOR COMPARISON. ED: THE MODEL WITH A RNN ENCODER AND A RNN DECODER; D: THE MODEL ONLY WITH A RNN DECODER; ED$\mathbb{D}$: THE MODEL WITH A RNN ENCODER, A RNN DECODER AND A RNN DISCRIMINATOR; D$\mathbb{D}$: THE MODEL WITH A RNN DECODER AND A RNN DISCRIMINATOR, NO ENCODER.

Model name	**Encoder**	**Decoder**	**Discriminator**
ED	✓	✓	
D		✓	
ED$\mathbb{D}$	✓	✓	✓
D$\mathbb{D}$(ours)		✓	✓

TABLE II

USER STUDY:PROPORTION OF USER VOTES ON SKETCHES GENERATED BY FOUR DIFFERENT MODELS.

Model name	**ED**	**D**	**ED$\mathbb{D}$**	**D$\mathbb{D}$(ours)**
Vote	6.5%	22.1%	22.2%	49.2%

our method can generate more reasonable and stable freehand sketches of a given 3D objects.

A. Data Processing

We capture the suggestive contours rendering of each model from different viewpoints to get two-dimensional line drawing images with size of 256×256. Then, the vectorization process is performed to obtain two-dimensional coordinate information. We also filter the vector line segments using the method described in the section III-A2. Note here the value of p of the **stroke-3** format is determined according to the start and end points of the line segment.

B. Implementation Details

For fairly comparison, we use LSTM for the encoder, decoder and discriminator. The encoder RNNs consist of 256 nodes, and the total number of nodes contained in both the decoder D and the discriminator $\mathbb{D}$ are set to be 512. We also apply recurrent dropout [32] to maintain a probability of 90%. We optimize using Adam [33] with the learning rate of the generator and discriminator to be 0.001, and the batch size to be 100. we also perform simple data augmentation by multiplying two IID random factors chosen uniformly from within 0.90 and 1.10 onto the offset columns $(\Delta x, \Delta y)$. We conduct comparative experiments on four network models, see Table I. Apart from their structural differences, i.e., with or without encoder or discriminator, all the common parameters used in these four network models are exactly the same.

C. Results and Analysis

For comparison, we first train four different generative network structures (see Table I) using the same training data, i.e., sketches from all 2300 viewpoints. First we test them on the 3D vase, wine glass, mug and key objects. For each of them, we generate three viewpoints to test these four network structure, see Figure 2. All the sketch results are produced with $N = 3$ (see Section III-C). It can be clearly observed that the sketches generated by the ED$\mathbb{D}$, D$\mathbb{D}$ networks, i.e., with a discriminator, have stronger stability of the sketch structure while at the same time exhibit reasonable freehand drafting styles created by real humans. The sketches generated by the ED and D model without a discriminator are more likely to become cluttered. We conclude that adding a discriminator to the network can effectively help generate sketches with correct overall shapes.

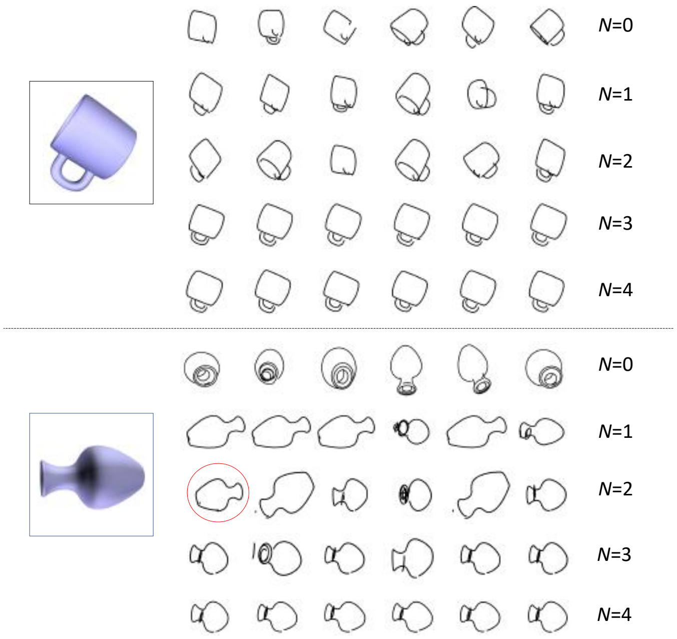

Fig. 3. The effects of parameter N. See Section IV-C for details.

We also conduct a user study to qualitatively analyze the generated sketch results. Then we invite 40 users to participate in the survey. First we prepare 60, i.e., 4×3×5 sketches, which are generated by the four network models in Table I, including three 3D objects: vases, wine glasses, hammers, and randomly select 5 different viewpoints for each 3D object. These 60 sketches are shown in Figure 6. Note that we set the temperature parameter τ for ED and D as 0.1, because even when the parameter is 0.1, the degree of randomness is very high. We set the parameter τ for ED$\mathbb{D}$ and D$\mathbb{D}$ as 0.3 to reduce their regularity. For each row of Figure 6, we first randomly shuffle the order of each sketch of the 4 networks, then we ask each user to vote for the sketch which looks more similar to human hand-drawings and more correctly describe the input shape. The statistic results are shown in Table II, which shows that very few people voted for ED, a small number of people voted for D and ED$\mathbb{D}$, and most people voted for D$\mathbb{D}$. It is easy to see that the user study results comply with our previous results and analysis that the network with a discriminator produces more aesthetically pleasing and acceptable sketches that are more preferred by average users.

We also test the effect of different values of the parameter N during our two-phase sampling step (Section III-C). Let us recall that the first N generated points are used as the anchor points to constrain the orientation of the generated sketch to be similar to the input condition. The test results are shown in Figure 3, where we show cases of 2 different viewpoints using our D$\mathbb{D}$ model when N equals 0, 1, 2, 3 and 4 respectively. For each viewpoint, 6 sketches are randomly sampled from our D$\mathbb{D}$ model. When $N = 0$, the experimental result of the generation is completely random, and its angle of view is totally unaffected by the input. As N increases, the angle of the generated sketch is closer to the input. We also observe that setting an overlarge N could generate over-constrained sketches without the satisfactory degree of abstraction to express human hand drawing style. Note that we set the temperature parameter τ for all result of Figure 3 as 0.2.

In addition to the parameter N, we use another parameter τ, denoted as temperature parameter,to control the degree of randomness in results. We test the effect of different τ and its impact is shown in Figure 4. Here we sample on two 3D objects: wine glass and hammer, where we increase τ from 0.1 to 0.8 by 0.1 interval. The sketch results are color-coded according to τ, i.e., τ = 0.1 is shown in blue, and τ = 0.8 in red. The valid range of value of τ is between 0 and 1. The results clearly suggest that with a increasing τ the degree of randomness of the sketch increases gradually, which complies with our prediction here.

Finally we test the replicability of our approach by using different training datasets. Specifically we retrain our D$\mathbb{D}$ network model using 3 different mixed datasets, see Figure 5. Here Dataset A consists of 50% Quickdraw sketches and 50% sketches from our synthetic dataset; Dataset B consists of 66% Quickdraw sketches and 34% sketches from our synthetic dataset; Dataset C consists of 100% Quickdraw sketches. Unfortunately we discover that even though the Quickdraw dataset clearly captures freehand sketch style, the mixed training fails to generate better results as universally exhibited in Figure 5. Even though all the results here are produced using our two-phase sampling method with $N = 3$ and the smallest possible $\tau = 0.1$, the retained network cannot correctly produce sketches with the same angle as the input. Please compare Figure 5 with Figure 2. The underlying reason for this is that most of the Quickdraw data are from those popular viewpoints that the average users prefer to select when they are asked to make a draft sketch of a certain object. For instance, most of the sketches of the wineglass in Quickdraw are from an upright and frontal viewpoint. This means the Quickdraw dataset does not provide enough information describing a 3D shape from any viewpoints.

Our approach has several limitations: Firstly, our method needs to train an individual neural network for every different 3D object, so it will be much powerful if we can achieve cross-modal generation in the future. Secondly, even though we specifically optimize for the balance between structural correctness and level of abstraction and distortion for our results, the actual human freehand drawings can still express much larger degree of diversity and abstraction than our results. Augmenting our dataset using real human freehand sketch from multiple viewpoints will be helpful for this issue. Thirdly, our approach has a preprocess of converting the 3D object into 2D image information that depends on third-party softwares.

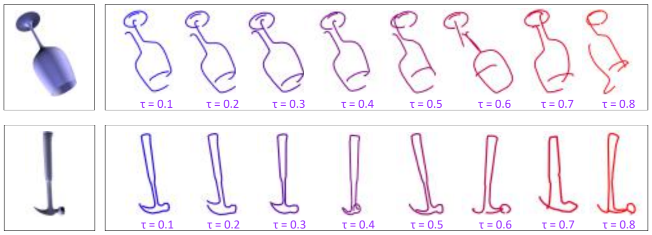

Fig. 4. Sketches with an increasing degree of freehand style generated using parameter τ increased from 0.1 to 0.8.See the electronic PDF for color image.

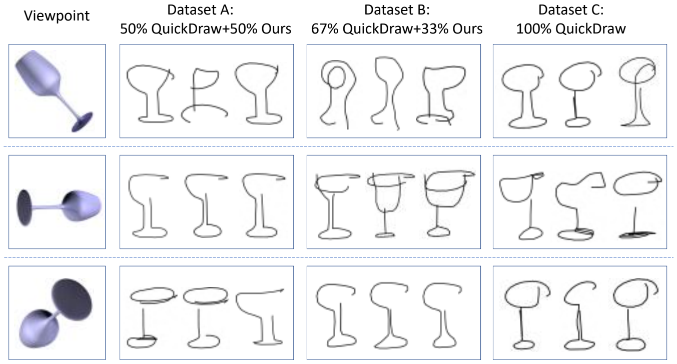

Fig. 5. Sketches generated when the Quickdraw data is added to our dataset. Note the wrong orientation of the sketches.

V. Conclusion

In this paper, we proposed a stroke-level 3D object-to-sketch synthesizing neural network that enables stroke-level cross-domain visual understanding from a reference 3D shape. The overall method is a supervised uncycled translation network trained for sketch generation. In order to increase structural stability, we remove the encoder and add a discriminator to better approximate the input shape. Our approach also allows users to tune the degree of randomness of the generated sketch by simply changing a single temperature parameter. We compare the performance of the four network models, i.e., Encoder with Decoder (ED), Decoder (D), Encoder and Decoder with Discriminator (ED$\mathbb{D}$), Decoder with Discriminator (D$\mathbb{D}$). It is shown that our model achieves better results in generating free hand drawing sketch of 3D objects.

For future work, we can try building a real human freehand drawing sketches database consisting of all possible viewpoints of various 3D models. We will also try to avoid the preprocessing process to enable full end-to-end sketch generation. Another direction is the exploration of other new neural network structure and vectorial representations of sketch lines for this topic.

Acknowledgment

Shizhe Zhou are supported by the grant of Science Foundation of Hunan Province (No. 2018JJ3064), National Science

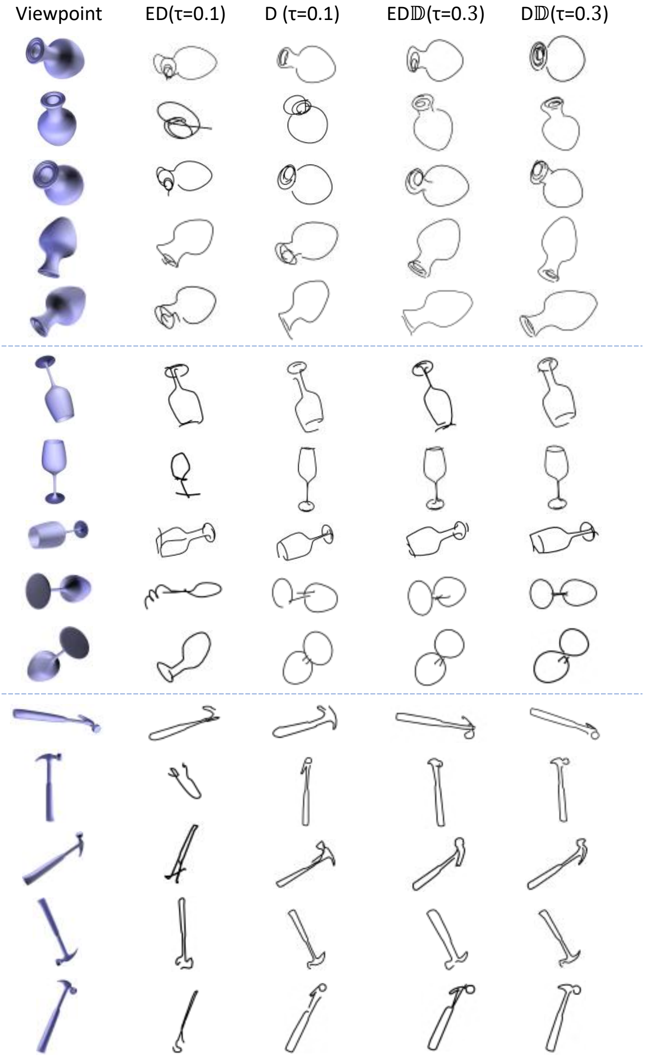

Fig. 6. Data for user study. The statistic results are shown in Table II.

Foundation of China (No. 61303147) and HPCSIP Key Laboratory,Ministry of Education, China. Hongbo Fu was partially supported by grants from the City University of Hong Kong (Project No. 7005176 (SCM)), and the Centre for Applied Computing and Interactive Media (ACIM) of School of Creative Media, CityU. We gratefully acknowledge the support of NVIDIA Corporation.

References

[1] D. Decarlo, A. Finkelstein, S. Rusinkiewicz, and A. Santella, "Suggestive contours for conveying shape," *Acm Transactions on Graphics*, vol. 22, no. 3, pp. 848–855, 2003.

[2] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *International Conference on Learning Representations*, 2014.

[3] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," *arXiv preprint arXiv:1401.4082*, 2014.

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

[5] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[6] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *arXiv preprint arXiv:1606.03498*, 2016.

[7] D. Ha and D. Eck, "A neural representation of sketch drawings," *arXiv preprint arXiv:1704.03477*, 2017.

[8] J. Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *IEEE International Conference on Computer Vision*, 2017, pp. 2242–2251.

[9] O. Mogren, "C-rnn-gan: Continuous recurrent neural networks with adversarial training," *arXiv preprint arXiv:1611.09904*, 2016.

[10] J. Bao, D. Chen, F. Wen, H. Li, and G. Hua, "Cvae-gan: Fine-grained image generation through asymmetric training," *arXiv preprint arXiv:1703.10155*, 2017.

[11] T. Kim, M. Cha, H. Kim, J. Lee, and J. Kim, "Learning to discover cross-domain relations with generative adversarial networks," *ICML*, pp. 1857–1865, 2017.

[12] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[13] Y. Mroueh, T. Sercu, and V. Goel, "Mcgan: Mean and covariance feature matching gan," *arxiv preprint arXiv:1702.08398*, 2017.

[14] G. J. Qi, "Loss-sensitive generative adversarial networks on lipschitz densities," *arxiv preprint arXiv:1701.06264*, 2017.

[15] A. B. L. Larsen, S. K. Snderby, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," *arXiv preprint arXiv:1512.09300*, 2015.

[16] A. Adate and B. Tripathy, "S-lstm-gan: Shared recurrent neural networks with adversarial training," in *Proceedings of the 2nd International Conference on Data Engineering and Communication Technology*, 2019, pp. 107–115.

[17] S. Simhon and G. Dudek, "Sketch interpretation and refinement using statistical models," in *Eurographics Workshop on Rendering Techniques, Norkping, Sweden, June*, 2004, pp. 23–32.

[18] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[19] C. M. Bishop, "Mixture density networks," Citeseer, Tech. Rep., 1994.

[20] Y. Chen, S. Tu, Y. Yi, L. Xu, Y. Chen, S. Tu, Y. Yi, L. Xu, Y. Chen, and S. Tu, "Sketch-pix2seq: a model to generate sketches of multiple categories," *arXiv preprint arXiv:1709.04121*, 2017.

[21] U. R. Muhammad, Y. Yang, Y.-Z. Song, T. Xiang, and T. M. Hospedales, "Learning deep sketch abstraction," in *CVPR*, 2018.

[22] J. Song, K. Pang, Y.-Z. Song, T. Xiang, and T. M. Hospedales, "Learning to sketch with shortcut cycle consistency," in *CVPR*, 2018.

[23] A. Hertzmann, "Introduction to 3d non-photorealistic rendering: Silhouettes and outlines," *Non-Photorealistic Rendering. SIGGRAPH*, vol. 99, no. 1, 1999.

[24] M. Y. Sang, M. Scherer, T. Schreck, and A. Kuijper, "Sketch-based 3d model retrieval using diffusion tensor fields of suggestive contours," in *International Conference on Multimedea*, 2010, pp. 193–200.

[25] S. M. Yoon and A. Kuijper, "View-based 3d model retrieval using compressive sensing based classification," in *The 7th International Symposium on Image and Signal Processing and Analysis, ISPA 2011, IEEE*, 2011, pp. 437–442.

[26] T. Judd, F. Durand, and E. H. Adelson, "Apparent ridges for line drawing." *Acm Transactions on Graphics*, vol. 26, no. 3, p. 19, 2007.

[27] F. Cole, A. Golovinskiy, A. Limpaecher, H. S. Barros, A. Finkelstein, T. A. Funkhouser, and S. Rusinkiewicz, "Where do people draw lines?" in *Acm Transactions on Graphics*, 2008, pp. 1–11.

[28] F. Cole, K. Sanik, D. Decarlo, A. Finkelstein, T. A. Funkhouser, S. Rusinkiewicz, and M. Singh, "How well do line drawings depict shape?" in *Acm Transactions on Graphics*, 2009, pp. 1–9.

[29] T. K. J. K. N. F.-G. Jonas Jongejan, Henry Rowley, "The quick, draw!" -A.I.Experiment, 2016, https://quickdraw.withgoogle.com.

[30] SoftSoft Ltd, "Wintopo." [Online]. Available: http://wintopo.com/

[31] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, no. 2, pp. 112–122, 1973.

[32] S. Semeniuta, A. Severyn, and E. Barth, "Recurrent dropout without memory loss," *arXiv preprint arXiv:1603.05118*, 2016.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.