# Sketch2Relief: Generating Bas-relief from Sketches with Deep Generative Networks

Shizhe Zhou[1,2,*] Zeyu Liu[1],
[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*
shizhe@hnu.edu.cn(corresponding author), zeyuliu@hnu.edu.cn

***Abstract*—We present a novel sketch-based system for generating digital bas-relief sculptures. All existing computational methods for generating digital bas-reliefs first require the input of a three-dimensional (3D) scene, thus preventing artists from freely creating or exploring designs when 3D data are not available. Motivated by this limitation, we propose a generative adversarial network (GAN)-based sketch modeling system for generating digital bas-reliefs from freehand user sketches (see Figure 1,5). The basic tool underpinning the interface is a conditional GAN (cGAN) that digitally learns a functional map from a contour image to a 3D model for any given viewpoint of the corresponding bas-relief model. When using our system for designing bas-reliefs, the user only needs to draw 2D sketch lines without having to designate any additional hints on the lines. The interface returns bas-relief results in interactive time (500 ms per bas-relief on average). We tested the quality and robustness of our approach with extensive and comprehensive experiments. By carefully analyzing the results, we verified that our system can faithfully reconstruct bas-reliefs from a test dataset and can generate completely new reliefs from raw amateur sketches.**

***Index Terms*—Multimedia Sketching Interface, Shape Modeling, Bas-Relief Design, Generative Adversarial Neural Networks**

## I. Introduction

A relief is a sculptural technique in which the sculpted shapes are attached to a solid background, giving the visual impression that the sculpted material has been raised above the background plane. There are three major types of reliefs: high reliefs, sunken reliefs and bas-reliefs (i.e., shallow reliefs), see the figure above. Manually creating a bas-relief is cumbersome and inefficient because the process entirely relies on the artist's three-dimensional spatial imagination and craftsmanship.

Over the past decade, several computational approaches that can convert a three-dimensional (3D) model input into a digital bas-relief have been successfully developed, e.g., [1]–[4]. In one of the pioneering works, Weyrich et al. [1] proposed a method for generating a digital bas-relief based on a given 3D model using high-dynamic-range compaction, i.e., for the 3D model at a specific viewpoint, its depth field is extracted and used in a gradient domain compression procedure. The obvious limitation of the aforementioned 3D model-based methods is the requirement for an input 3D model, which sometimes becomes problematic when designing a relief when there are no corresponding 3D models available.

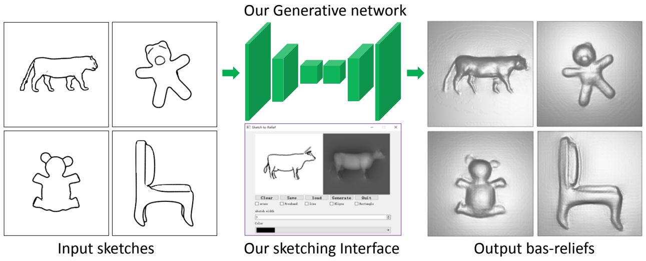

Fig. 1: We trained a deep neural network that enables users to quickly generate realistic bas-relief meshes with simple 2D sketches.

To work around this limitation, in this study, we focus on developing a novel approach for generating digital bas-reliefs, without the necessity of inputting any 3D models.Freehand sketching is arguably one of the most accessible and efficient means for expressing design ideas.It is not surprising that most of the modeling software uses a sketch interface for the design and manipulation of graphical content.The need for this type of interface inspired us to develop a sketch-based interface that is able to automatically infer a 3D bas-relief interpretation from a random 2D conceptual sketch. We believe that such a system is highly desirable for graphic designers since it not only overcomes the 3D model limitation but also releases the creativity of novice users, due to the concision and flexibility enabled by freehand sketches.

Deep neural networks have been proven to be powerful machine learning tools that enable a variety of applications both in computer graphics and digital image processing. Recent studies regarding deep learning have provided an all-important method for image synthesis, which is termed generative adversarial networks (GANs) [5]. Thus, in this paper, we propose a GAN-based sketch-to-relief generation system in which our neural network framework infers a 2D height field for the bas-relief from the corresponding hand-drawn sketches.

2375-0197/20/$31.00 ©2020 IEEE
DOI 10.1109/ICTAI50040.2020.00085

We use a synthetic approach to prepare the training data. Specifically, for a given 3D model, we extract its 2D salient feature lines from suggestive contours, ridges, valleys, etc., from an extensive set of viewpoints. For the same set of viewpoints, we use a highly efficient method to produce 2D digital bas-relief models. We then train our network model by using a large number of sketch images and the height field maps of their corresponding relief models.

We demonstrate the stabilization of our method on large-scale comparisons in both quantitative and qualitative experiments. We illustrate the refined ability of our model to generate low-error 2D height fields through comparisons with two other methods, pix2pix [8] and that of Su et al. [9], evaluated via three types of data sets. We demonstrate the stability and accuracy of our model by calculating the L1 and L2 losses of the original dataset against the generated 2D height field value. We rely on the error graph to demonstrate the accuracy of our model and evaluate the robustness of our approach by gradually increasing the strokes of the input sketch. The feasibility of our method is also measurable from the user's hand drawing. User research also proves that our approach has an advantage from a user perception perspective.

To summarize, the research contributions emerging from this work are the following: 1) constructing a sketching interface for modeling bas-reliefs without the need for 3D models; 2) incorporating a GAN-based neural network behind the interface to convert 2D sketch images into bas-relief height fields; 3) carefully analyzing the performance of the network in terms of quality results; 4) a new dataset of sketch-relief for benchmarking the performance of relief-related algorithms.

## II. Related work

### A. Digital bas-relief generation from 3D models

Digital bas-relief has attracted an increasing amount of computer graphics research attention over the past decade. A pioneering work [1] presented the first computational scheme for transferring a depth field of a given 3D scene into a digital bas-relief. The core of the approach is a Poisson equation that reconstructs a height field based on a compressed divergence field. Along with this line of work, several methods have been proposed to convert a 3D scene into a bas-relief while incorporating other geometric information. For example, Alldrin et al. [10] proposed a new prior on the albedo distribution, which specified that the entropy of the distribution should be low. This prior is justified by the fact that many objects in the real world are composed of a small finite set of albedo values. Sun et al. [2] presented methods to automatically generate bas-reliefs based on an adaptive histogram equalization (AHE) starting from an input height field. We will present the details of bulk-producing bas-reliefs from 3D models as training data for our deep neural network in Section IV-A.

### B. Sketch-based modeling

Sketch-based modeling is a highly focused and productive research field, and in the past, many remarkable results have been achieved (see the survey in [11]). Based on the underlying dataset, we can generally categorize the existing methods into 2 classes, direct sketch-based modeling and data-driven sketch-based modeling.

For direct sketch-based modeling, the sketch is usually regarded as the outer border of the potential 3D object. The actual surface can be produced as an interpolation of the geometrical information attached to the sketches, e.g., normals and curvatures. For example, Igarashi et al. proposed Teddy [12] as one of the earliest real-time interactive sketch modeling interfaces. The concept of user sketch can also be extended to 3D curves; e.g., [13] used 3D curves with an arbitrary topology as handles to edit the precreated 3D shapes. Multi-view sketching interfaces have also been developed; e.g., [14] created topologically nontrivial 3D shapes using silhouettes from multiple views.

For data-driven sketch-based modeling, the sketch can either be used as a query to search the dataset for the optimal shapes or retrieve partial candidates for assembling human-made objects [15],or it can be used as a fitting target for the deformable model to approximate. We rely on a GAN to directly generate the bas-relief data without any template shapes or the need to create an intermediate surface. The recent rapid advancement of deep learning has provided sketch-based image synthesis problem areas with new and effective solutions. For example, Lun et al. [16] proposed a method for reconstructing 3D shapes from 2D sketches of line drawings. They converted multiple sketches into multiview maps, which were then consolidated into a 3D shape. Su et al. [9] used different perspectives of sketches as input to generate high-quality normal maps in real time. Han et al. [17] built a sketch interface for designing exaggerated 3D facial expressions,the core algorithm of which is a two-branch CNN that drives a bilinear human face morphable model. Although they also treat user sketch as an image, their network outputs semantic attribute as parameters to extract face instances from the morphable model, while unlike them, we directly generate the 3D shape of the bas-relief.

### C. Image translation

Traditional image translation effects are achieved by mechanisms based on handcrafted separate local image representations, e.g., image quilting [18], image analogies [19], and image denoising [20]. Efros and Freeman [18] utilized a texture synthesis model for a corresponding input-output image pair. More advanced approaches use a dataset of input-output example pairs to learn a parametric translation function using GANs [5]. Mirza and Osindero [6] proposed cGANs,adding additional information to the random noise to generate images.

Improved techniques for training GANs by Salimans et al. [21] present a number of new architectural features and training procedures, e.g., matching feature and minibatch discrimination to improve the generated images. Three mutations of GANs are published at approximately the same time – DualGAN [22], CycleGAN [23] and DiscoGAN [24] – all propose symmetric cyclic networks that accurately convert data domains in a bidirectional loops. Animesh et al. [30]

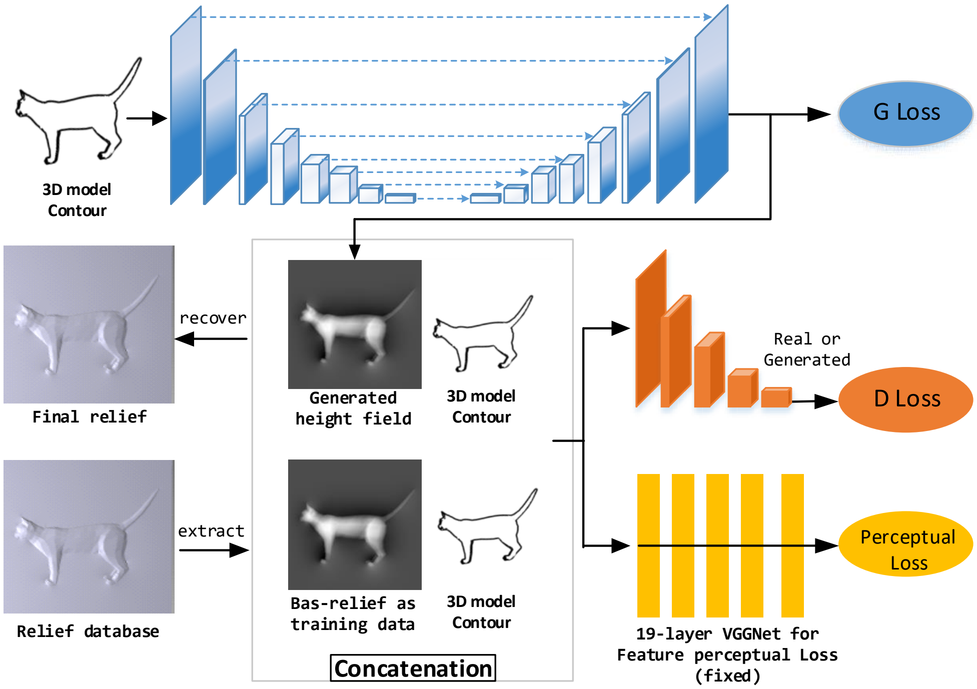

Fig. 2: Our network structure. On the upper left of the figure are the input data of the generation network. We enter the sketch into the generated model to obtain the generated false relief height field. We then extract the relief database to obtain the corresponding relief height field (similar to a 2D image). The sketch and the real relief height field and the generated relief height field are concatenated as the input feed to the discriminator network. In addition, we feed the real relief height field together with the generated height field into the VGGNet [32] to compute the perceptual loss. When testing, the user only needs to input the sketch into the generation network to obtain the final relief height field and then recover the generated height field from the relief model.

propose a training technique for training GANs by allowing the flow of gradients from the discriminator to the generator at multiple scales. Yang et al. [31] present the first text style transfer network that allows for real-time control of the crucial stylistic degree of the glyph through an adjustable parameter. Baris et al.

From a workflow point of view, our work is very similar to [9], which is also inspired by pix2pix [8] presented by Isola et al., because pix2pix proposed a general framework for solving image-to-image reinterpretations via deep learning. However, unlike [9], our method attempts to learn a reasonable mapping from the sketch line to a 2D height field that often contains many compressed geometric details. The nature of the height field and the surface normal are clearly different. Without positional information, it is impossible to recover the surface solely from the normals. However, our solution directly generates a 3D manifold surface, although it is very thin and topologically simple.

## III. Methodology

Since both the bas-relief and the contour sketch can be seen as 2D images, we regard the sketch-to-relief problem as an image-to-image translation process. Furthermore, treating the user's 2D sketches as images helps maintain a flexible sketching interface to allow a user to draw any number of lines. From a CNN point of view, the essence of the image-to-image translation is to estimate the probability distributions of two image domains. In our case, the input sketch images are grayscale images, in which white pixels are the background, and the output bas-relief images are also grayscale images, in which the pixel value stands for a normalized height field. In Section III-A, we first introduce the objective function of our GAN. The architecture of our network is elaborated in Section III-B and illustrated in Fig. 2.

### A. Objective function

To reflect the constraints of sketch lines, we use a cGAN [6] that maps a random noise vector $z$ to an image $y : z \rightarrow y$ that is constrained by the input condition $x$. We use cGAN learns a *Generator* $G$ that maps the input condition $x$ to the $y$:

$$G : x \rightarrow y \tag{1}$$

Under the conditional settings, generator $G$ and discriminator $D$ will approximate the conditional distribution of $y$ onto the distribution of the training data. For image translation problems, feeding $G$ with the input image guides the network to generate more accurate output images, which is constrained to effectively reflect the condition data $x$, i.e., the sketch image. According to Mirza et al. [6], the loss function used in cGANs is defined as follows:

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

Again, y is the 2D height field of the corresponding bas-relief, x represents the input sketch image,which is encoded to be the latent space of the generator G. $p_s$, $p_n$ represent the sketch domain and bas-relief height field domain , respectively. However, Arjovsky and Bottou [28] proved that optimizing such a loss function is similar to maximizing the Jensen-Shannon divergence (JSD), but in fact, the JSD can't effectively measure the distribution of input and output. We use the WGAN-GP objective function[29] during training. Consequently, we modify our loss function in Eq. 3 as follows:

$$\begin{aligned} Loss =& E_{x\sim p_s, y\sim p_n}[D(y|x)] - E_{\tilde{y}\sim p_g}[D(G(\tilde{y}|x)] \\ &- \lambda_{L_1}L_1 - \lambda_{VGG}L_{VGG} \end{aligned} \tag{3}$$

Here, $\tilde{y}$ is the generated relief height field with respect to the input sketch $x$ from the generated bas-relief domain. To further speed up the training of the network and improve the quality of the generated data, we use $L_{L1}$ (in Eq. 4) to calculate the L1 distance between the generated picture and the original picture, and we use the pretrained VGG network to calculate the perceptual loss $L_{VGG}$ (in Eq. 5):

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

$$L_{VGG} = E_{y\sim p_n, \tilde{y}\sim p_g}[\gamma_i \|V_i(y) - V_i(\tilde{y})\|_1] \tag{5}$$

$$\tilde{y} = E_{x\sim p_s, z\sim p_r}[G(x|z)] \tag{6}$$

Similar to Johnson et al. [27], we use VGGNet to compute the perceptual loss to enforce deep feature consistency between the output and the input images. Here, we use the L1 loss to calculate the difference between deep feature graphs. Instead of only using a single layer features, we leverage visual features at multiple scales and use the outputs of the five convolutional layers of the VGGNet. $\gamma_i = \sum_{i=1}^{5} \frac{100}{C_i^2} L_i$,where $L_i$ are the feature perceptual loss and $C_i^2$ are the square of the number of filters in each layer of the VGGNet.

### B. Network structure

Our full structure of our network is presented in Figure 2. The input is a 256*256 size sketch image (3-channel RGB image). We then feed the sketch into the generator $G$. For discriminator $D$, there are 5 layers, each consisting of convolution, batch normalization and leaky rectified linear units (ReLUs) to process the data stream. For generator $G$, we adopt an encoder-decoder architecture [28], which is a universal choice based on image generation problems.

In image translation problems, many studies show that it is beneficial to share low-level information between the input and output, e.g., to directly interconnect the extracted information over two distant layers. For example, we can easily observe the input sketch and the bas-relief of the output share the same boundary position. Since our generator $G$ is symmetric, we adopt a U-net structure, i.e., we add skip connections over the generator $G$ (see Figure 2). Specifically, we add skip connections after the batch normalization in the generator $G$ in between each layer $i$ and layer $n-i$, where $n=16$ is the total numbers of layers in G. These skip connections concatenate all channels at layer $i$ with those at layer $n-i$.

## IV. Training data

Since our design ideology of generating reliefs from 2D draft sketches is rarely practiced in the real world, it would be ineffective to obtain datasets from human artists. We instead use a synthetic approach to build our training and test datasets.

### A. Bas-reliefs

Our neural network requires a large number of bas-reliefs with different viewpoints as training data. However, it is not easy to find the data needed from online datasets. Manually creating them one-by-one using inefficient software is also impractical. Therefore, instead, we synthetically bulk-produce a bas-relief dataset for a small collection of 3D models using our previous graphics processing unit (GPU)-based relief generation method.It follows the same mathematical workflow as [1], but using a different high dynamic range (HDR) compression function.

Specifically, for a given 3D scene(or a single 3D model), we extract its depth field $h(x,y)$ defined in a 2D rectangle screen domain $\pi$, and we compute the gradient of depth $g(x,y) = \nabla(x,y)$. The gradient can be represented as $g(x,y) = \|g\|\hat{g}(x,y)$, where $\hat{g}$ denotes the unit vector. For each point in $\pi$, we compress the magnitude of the gradient vector while preserving its direction, i.e., $g'(x,y) = C(\|g(x,y)\|)\hat{g}(x,y)$, where $C$ is a HDR compressing function defined as:

$$C(x) = \begin{cases} \alpha \tan^{-1}(\beta x) & 0 \le x \le \phi_{thred} \ , \\ 0 & else \ . \end{cases}$$

The parameter $\alpha$ and $\beta$ control the degree of compression over different value ranges.

[1] uses the logarithmic function $C' = \frac{1}{\eta} ln(1+\eta x)$ as their compression function, where $\eta$ is normally between 0.5 and 10; greater values corresponding to a stronger compression. Throughout this paper, we use $C(x)$ with $\alpha$=1.5 and $\beta$=0.8. Then, the compressed gradient $g'$ is used for computing the divergence in the Poisson equation:

$$\begin{cases} \Delta r(x,y) = \nabla \cdot g' & \text{if } (x,y) \in \pi \\ r(x,y) = 0 & \text{if } (x,y) \in \partial\pi \end{cases} \tag{7}$$

The solution $r(x,y)$ is the height field of the result bas-relief. We implement the aforementioned algorithm on a GPU using framebuffer computing technology to obtain a relatively high efficiency.

For fast bulk production, we run our program on an NVIDIA GTX-2080Ti and make it automatically rotate the 3D model. For each different viewpoint, its corresponding bas-relief is computed on the GPU and saved on the hard drive as an OBJ mesh file. In Table 1, we measure this normalization error by computing a relative error E on the four types of 3D model groups.Based on the statistics, we conclude that this quantization error is neglectable.

**Table 1.**

Relief-to-image-to-relief quantization error statistics. From each of the four types of 3D model groups, we randomly choose 100 data point to calculate their average error ratio. We observe that for all types of 3D models, the ratio does not exceed five percent.

| Dataset | Teddy | Chair | Four legs | Head |
|---|---|---|---|---|
| Error | 4.11% | 4.29% | 4.65% | 3.26% |

### B. Sketch images

To obtain the corresponding sketch images, we adopt suggestive contours [29] as proposed by DeCarlo et al. to generate line drawings directly from a 3D model with multiple views that are exactly the same as those we use for generating bas-reliefs. The size of the sketch image is also 256×256, the same as the relief image.

Note that we also attempted the traditional data enhancement methods, i.e., to mirror, flip and rotate a sketch-relief image pair, to enlarge the diversity of the training data. However, we find that trained networks exhibit no significant change from such enhancement. The quality of the generated results and the limitations remain the same.

## V. Results and Analysis

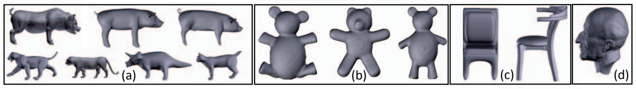

Fig. 3: Our dataset includes four types of 3D models.

We tested our approach on 4 types of 3D model groups: (a) Four Legs, (b) Teddy, (c) Chairs and (d) Human Head, as shown in the figure above. For each 3D model, we use 80% of all 2500 views, i.e., 2000 sketch image + bas-reliefs image pairs for training, and the remaining 500 views are used for testing. In the testing phase, for better reflecting the robustness of our model, we tested not only 3D models in the 4 groups but also some 3D meshes outside the training data, see Fig. 11 and 5.It generally takes about 0.5s to generate one height field image, and takes about 2s to convert it into a mesh file.

First, we show in Figure. 4 select test results using sketch images from the test dataset. We can observe that most of the generated reliefs faithfully reflect the shapes of the sketched objects. When a sketch line is sparse and clean, the boundary of its corresponding object in the relief is as well, whereas when the lines become cluttered, those boundaries become unsmooth or even broken, e.g., Figures 4 $f4$ and $d6$. We can also notice a minor deviation in the viewing direction between the sketch image and relief, e.g., Figure 4 $f10$.

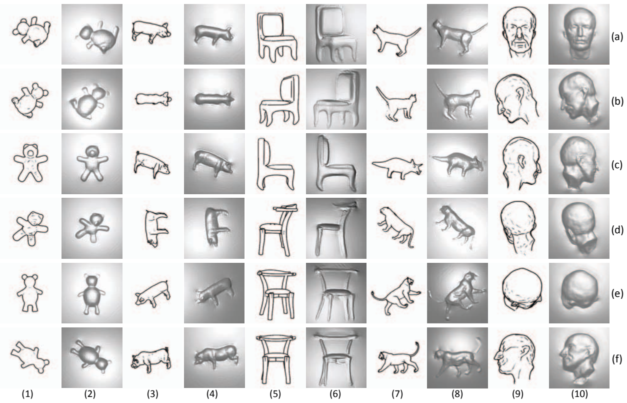

Fig. 4: Bas-relief results generated by our system using sketch images from the test dataset. For each column, the left side is the sketch input, and the right side is the corresponding rendered bas-relief model. From top to bottom, the results are roughly organized in order of increasing complexity.

We also invited some amateur users to draw freehand sketches to test our system; the results (see Figure 5) show that the system can infer reasonable bas-reliefs with completely new sketch inputs. Since we only use synthetic data as our training data, it is not surprising to observe that the object shape drawn by the users somehow *morphed* to approximate the contour shape of the networks' underlying 3D model types, e.g., Figure 5 $column6$.

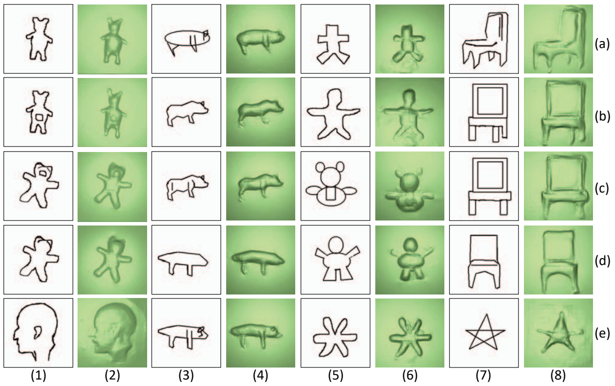

Fig. 5: Result bas-reliefs generated from amateur user freehand sketches using our system. From a1 to b1, b3 to c3, and d3 to e3 the user adds sketch lines to update the results, while from c1 to d1 the user erases a sketch line to do so.

An advantage of our system over the traditional methods is that we allow the user to directly edit relief shape details by simply drawing (see Figure 6) or removing (see Figure 5) the detailed lines on the sketch image. In contrast, the traditional 3D model-based methods require the users to deform or rebuild the 3D shape itself, which is inconvenient and unintuitive.

### A. Absolute Error

To analyze our network, we first compute the absolute pixel error for each pair of generated relief image $H^g$ and its

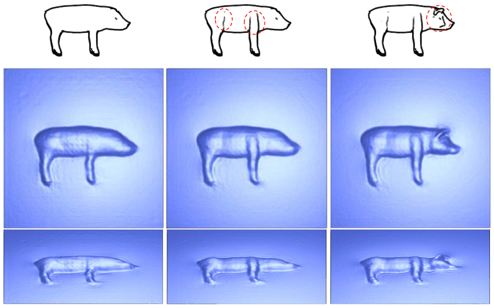

Fig. 6: Interactive incremental refinements by adding additional strokes, which lead to additional geometry details in the bas-relief. The modified strokes are marked with some red circles.

corresponding ground truth image $H$ from the test dataset. The absolute error $E_A$ is defined as

$$E_A = \sum_i \|H_i^g - H_i\| \tag{8}$$

Figure 7 displays some example absolute error maps. In each row, we show 4 different viewing directions within the same 3D model type. Overall, the bas-relief generated by our model accurately reflects the characteristics of the input sketch and has correct boundaries. For area with salient line features, i.e., a foreground object, the error becomes larger, especially when the input 3D model exhibits obvious relief ambiguities (see Figure 13). However, even in this case, the resulting bas-relief has an acceptable quality because our training data contain almost no ambiguous relief examples. The reason for a pure background to have a nonzero error is because the solution of the Poisson equation (see Section IV-A) can barely maintain its zero value, except for those points right on the domain border.

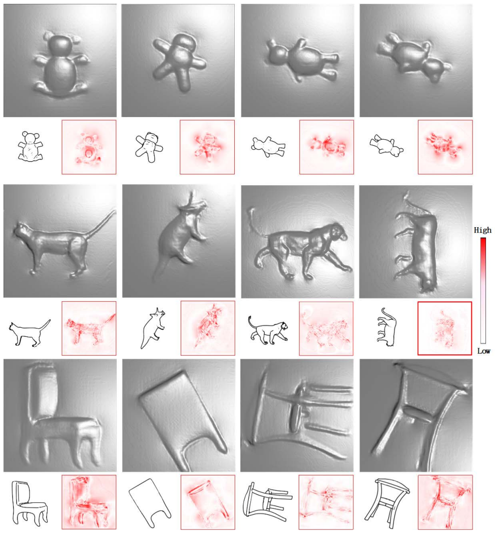

Fig. 7: Bas-reliefs generated with sketch in the test dataset; their error maps are shown in the lower-right images of each group. Each error map is independently normalized and color-coded from pure white to pure red.

We compare our method with two closely related works: pix2pix [8], Su et al. [9].We train the pix2pix [8] and Su et al. [9] directly using our dataset. Note that Su et al. [9] aims at generating vectorial normal maps which can accept point mask constraints. While in our case of generating relief height field we need no extra input other than sketch lines, reducing the dimensionality of input.

Because we are essentially generating images, we compute the difference between the resulting image and its ground truth using the $L1$ and $L2$ versions of the absolute error $E_A$. The quantitative results are reported in Table V-A.

**Table 2.**

The absolute errors $E_A$ (see Equation V-A) relative to the ground truth image for 3 different methods: pix2pix [8],Su et al. [9] and ours. All the values here are the average error over all test images from 3D model types (a) Four Legs and (c) Chairs.

| Dataset | **Loss Type** | **pix2pix** | **Su et al.** | **Ours** |
|---|---|---|---|---|
| Chair | L1 | 972802 | 528663 | **302047** |
| | L2 | 5095.18 | 3066.03 | **1874.38** |
| Four Legs | L1 | 488801 | 392177 | **269373** |
| | L2 | 2644.77 | 2113.34 | **1519.33** |

To further demonstrate the difference in the results produced by the different methods, we visually compare the error maps generated by different methods in Figure 8. It can be observed, more intuitively, that our method produces fewer errors than pix2pix [8] and the method from Su et al. [9]. Our network fits better on the boundaries of the objects, producing bas-reliefs with smoother contours.

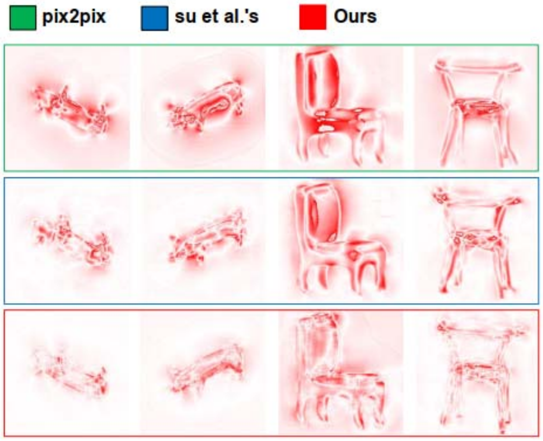

Fig. 8: Visualization of the $E_A$ error maps of the different methods. We selected four examples from the test dataset for each group.

### B. Relative Error

To further analyze the ability and quality of describing a compressed 3D shape by our generative neural network, we need to precisely measure the actual shape-wise loss on the generated reliefs. However, the absolute error $E_A$ indiscriminately totals all the pixels, including the blank area where no relief object exists; nevertheless, the visual quality of a bas-relief is primarily determined by the actual projection area of the 3D model, especially by its contour regions. Furthermore, $E_A$ only computes local pixelwise differences, which do not reflect the variation in the geometric detail variations of the relief surface. To measure the real shape-wise loss of the generated relief, we define the relative pixel error $E_R$ based on the Laplacian operator:

$$E = \frac{\sum_i \epsilon_i}{A_{relief}} \tag{9}$$

$A_{relief}$ is the size of the area with most salient relief shape variation; see Figure 9. To correctly compute $A_{relief}$, we need to carefully exclude blank areas, which often contain very minor value fluctuations that drift it slightly away from the base plane, leading to accumulated error. To this end, we define $\epsilon_i$ using thresholding of the gradient and height:

$$\epsilon_i = \begin{cases} |\Delta H_i^g - \Delta H_i| \; if \; \|\nabla H_i\|_1 > \gamma_1 \; or \; |H_i - H_{base}| > \gamma_2 \\ 0, i \notin A_{relief} \;\; else \end{cases} \tag{10}$$

where $\Delta H_i = H_{right} + H_{left} + H_{top} + H_{down} - 4 \times H_i, \nabla H_i = (H_{right} - H_i, H_{top} + H_i)$. For all examples, we set $\gamma_1 = 3.8, \gamma_2 = 38$.

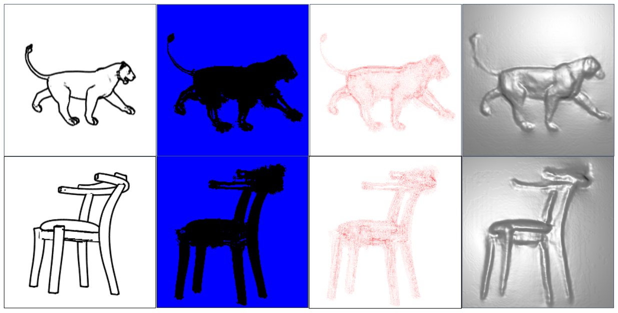

Fig. 9:From left to right: sketch image; salient region $A_{relief}$ (shown in black); relative error map, i.e., $E_A$; generated relief.

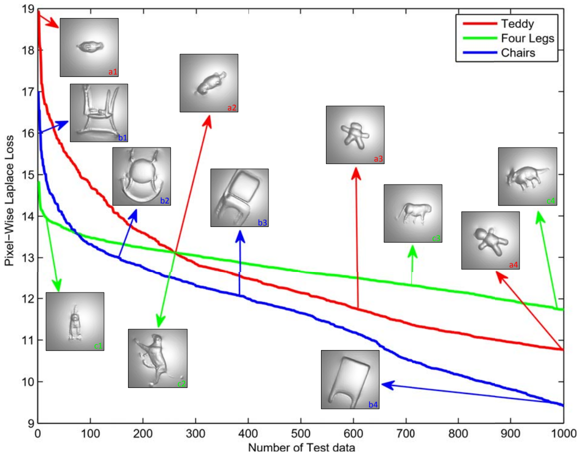

Fig. 10: Sorted relative error curve. For each 3D model type, we compute the pixel-wise laplace loss,i.e.,$E_R$ for 1000 test data and display 4 representative results from the largest error to the smallest error.

We test 1000 data points from the Teddy, Four Legs and Chair 3D model types and sorted their relative error $E_R$ as a plot; see Figure. 10. The common trend of the 3 curves clearly shows that the more the contour lines overlap with each other, the larger $E_R$ becomes, e.g., Figure. 10 a1, b1 and c1; When the number of sketch lines decreases or the average distance of the lines becomes larger, $E_R$ decreases. Interestingly, as the error decreases, the input sketch becomes closer to the common viewing direction that we usually choose for manually creating a bas-relief. Another similar conclusion is that the less ambiguous the bas-relief is with respect to its corresponding sketch image, the smaller the error value is.

In Figure. 11 we test our networks using suggestive contours of 3D models that are not contained in our training set. For the first two columns we use animals that are not in group (a) Four Legs, and for the third and fourth columns we use teddy bears with poses different from the group (b) Teddy. We can see that even though the input sketches are with totally new shapes and posed that are not inside the training set, the predicted bas-reliefs are still very convincing. In the last column we use a sketch from [16] that has large discrepancy both in shape and topology against our group(c) Chairs, our network cannot continue to output height field with high accuracy. This implies that topology seems to have more impact on the consistency with the distribution of the training data than pose and shape.

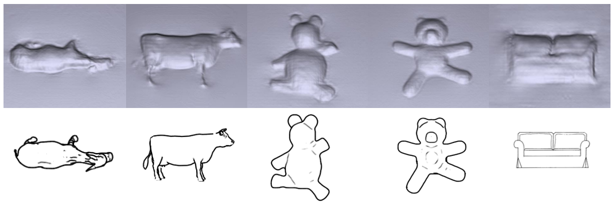

Fig. 11: Results using sketches of 3D models not contained in our training set. Note here the teddy bears are with totally new poses. The last sketch is from [16].

We also test our network on completely new freehand line drawings created by amateur users, shown in Figure. 5.Our network can accurately generate visually realistic bas-reliefs for most user-entered sketches.

## VI. Conclusions and Future works

In this work, we propose a novel sketching interface to generate bas-relief models. Underlying the interface is a GAN trained from synthetic contour images and relief height field data. Our generation network uses a structure similar to the U-Net network [7]. For the loss function, we use WGAN_GP [26] to reduce the gradient by measuring the Wasserstein distance. We introduce VGG19 to calculate the perceptual loss, enabling the network to converge more quickly and smoothly for the resulting picture. The experiments show that our network can generate not only rotund 3D objects but also highly articulated objects, such as chairs and tigers; see Figure 9 and 4. We also invited amateur users to draw completely new sketches, and our system could still produce reasonable results; see Figure 5. In summary, our GAN framework is more stable than pix2pix [8] and Su et al. [9], and is more suitable for relief height field generation than [16].

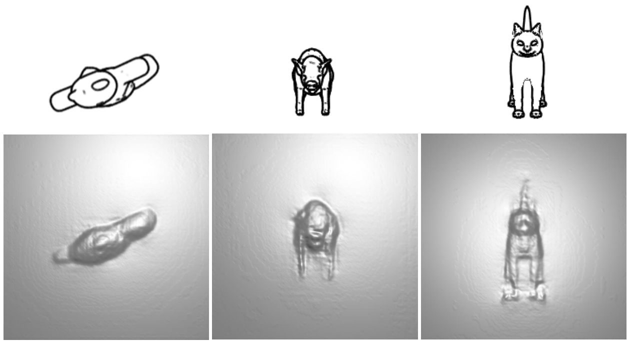

Fig. 12: Failure cases: unsatisfactory results are generated when the input sketches exhibit severe overlap, preventing the network from extracting and generating an accurate height field. Furthermore, these input sketches are often from very singular viewpoints that are seldom chosen by real artists.

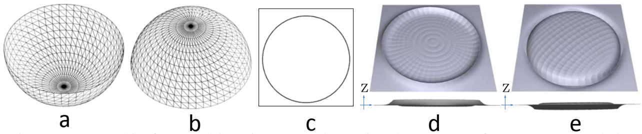

Fig. 13: Relief Ambiguity: A hemisphere surface (a) and its inverted version (b) share the same contour line, i.e., a circle (c), under the top-down vertical view direction. However, they generate different bas-relief surfaces in (d) and (e),respectively. The sideviews of the height field are also displayed at the bottoms of (d) and (e); the x-axis in the small coordinate cross represents the XOY plane, i.e., Z = 0.

The limitations of our approach are:1) Due to the limited image resolution,when the sketch lines have high density, the results tends to contain unsatisfactory surfaces; see Figure 12.The reason is that in these cases, even the training data are not very neatly shaped. When the input sketch has a complex form, the network may generate a discontinuous height field. As the features of the sketch become too condensed, the network may be too confused to correctly infer the underlying structure of the shape. 2) Our networks do not learn a unified mapping function for all types of the 3D models, which is a situation that we want to improve by introducing transfer learning in the future.

## VII. Acknowledge

This work is supported by the grant No.62076090, No.61303147 of National Science Foundation of China, No.2018JJ3064 of Science Foundation of Hunan Province and the 2020 Young Talents project of Hunan Province,China.We gratefully acknowledge NVIDIA for its GPU donation.

## References

[1] T. Weyrich, J. Deng, C. Barnes, S. Rusinkiewicz, and A. Finkelstein, "Digital bas-relief from 3d scenes," in *SIGGRAPH '07*, 2007.

[2] X. Sun, P. L. Rosin, R. R. Martin, and F. C. Langbein, "Bas-relief generation using adaptive histogram equalization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, pp. 642–653, 2009.

[3] Z. Ji, X. Sun, S. Li, and Y. Wang, "Real-time bas-relief generation from depth-and-normal maps on gpu," *Comput. Graph. Forum*, vol. 33, pp. 75–83, 2014.

[4] Y.-W. Zhang, C. Zhang, W. Wang, and Y. Chen, "Adaptive bas-relief generation from 3d object under illumination," *Comput. Graph. Forum*, vol. 35, pp. 311–321, 2016.

[5] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial networks," *CoRR*, vol. abs/1406.2661, 2014.

[6] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *CoRR*, vol. abs/1411.1784, 2014.

[7] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015.

[8] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5967–5976, 2017.

[9] W. Su, D. Du, X. Yang, S. Zhou, and H. Fu, "Interactive sketch-based normal map generation with deep neural networks," in *PACMCGIT*, 2018.

[10] N. G. Alldrin, S. P. Mallick, and D. J. Kriegman, "Resolving the generalized bas-relief ambiguity by entropy minimization," *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–7, 2007.

[11] E. V. Brazil, R. Amorim, M. C. Sousa, L. Velho, and L. H. de Figueiredo, "Sketch-based modeling and adaptive meshes," *Computers . Graphics*, vol. 52, pp. 116–128, 2015.

[12] T. Igarashi, S. Matsuoka, and H. Tanaka, "Teddy: a sketching interface for 3d freeform design," in *SIGGRAPH Courses*, 1999.

[13] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa, "Fibermesh: designing freeform surfaces with 3d curves," in *SIGGRAPH '07*, 2007.

[14] A. R. Rivers, F. Durand, and T. Igarashi, "3d modeling with silhouettes," in *SIGGRAPH '10*, 2010.

[15] X. Xie, K. Xu, N. J. Mitra, D. Cohen-Or, and B. Chen, "Sketch-to-design: Context-based part assembly," *Comput. Graph. Forum*, vol. 32, pp. 233–245, 2013.

[16] Z. Lun, M. Gadelha, E. Kalogerakis, S. Maji, and R. Wang, "3d shape reconstruction from sketches via multi-view convolutional networks," *2017 International Conference on 3D Vision (3DV)*, pp. 67–77, 2017.

[17] X. Han, C. Gao, and Y. Yu, "Deepsketch2face: A deep learning based sketching system for 3d face and caricature modeling," *ACM Trans. Graph.*, vol. 36, pp. 126:1–126:12, 2017.

[18] A. A. Efros and W. T. Freeman, "Image quilting for texture synthesis and transfer," in *SIGGRAPH*, 2001.

[19] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. Salesin, "Image analogies," in *SIGGRAPH*, 2001.

[20] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*.

[21] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *NIPS*, 2016.

[22] Z. Yi, H. Zhang, P. Tan, and M. Gong, "Dualgan: Unsupervised dual learning for image-to-image translation," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2868–2876, 2017.

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

[24] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim, "Learning to discover cross-domain relations with generative adversarial networks," in *ICML*, 2017.

[25] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," *CoRR*, vol. abs/1701.04862, 2017.

[26] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *NIPS*, 2017.

[27] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *ECCV*, 2016.

[28] G. E. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks." *Science*, vol. 313 5786, pp. 504–7, 2006.

[29] D. DeCarlo, A. Finkelstein, S. Rusinkiewicz, and A. Santella, "Suggestive contours for conveying shape," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 848–855, 2003.

[30] Animesh Karnewar, Oliver Wang, and Raghu Sesha Iyengar. MSG-GAN: multi-scale gradient GAN for stable image synthesis. *CoRR*, abs/1903.06048, 2019.

[31] Shuai Yang, Zhangyang Wang, Zhaowen Wang, Ning Xu, Jiaying Liu, and Zongming Guo. Controllable artistic text style transfer via shape-matching GAN. *CoRR*, abs/1905.01354, 2019.

[32] Karen Simonyan, Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.