SAniHead: Sketching Animal-like 3D Character Heads Using a View-surface Collaborative Mesh Generative Network

Dong Du, Xiaoguang Han, *Member, IEEE,* Hongbo Fu, Feiyang Wu, Yizhou Yu, *Fellow, IEEE,* Shuguang Cui, *Fellow, IEEE,* Ligang Liu, *Member, IEEE*

Abstract—In the game and film industries, modeling 3D heads plays a very important role in designing characters. Although human head modeling has been researched for a long time, few works have focused on animal-like heads, which are of more diverse shapes and richer geometric details. In this work, we present *SAniHead*, an interactive system for creating animal-like heads with a mesh representation from dual-view sketches. Our core technical contribution is a view-surface collaborative mesh generative network. Initially, a Graph Convolutional Neural Network(GCNN) is trained to learn the deformation of a template mesh to fit the shape of sketches, giving rise to a coarse model. It is then projected into vertex maps where image-to-image translation networks are performed for detail inference. After back-projecting the inferred details onto the meshed surface, a new GCNN is trained for further detail refinement. The modules of view-based detail inference and surface-based detail refinement are conducted in an alternating cascaded fashion, collaboratively improving the model. A refinement sketching interface is also implemented to support direct mesh manipulation. Experimental results show the superiority of our approach and the usability of our interactive system. Our work also contributes a 3D animal head dataset with corresponding line drawings.

Index Terms—Sketch-based 3D Modeling, Graph Convolutional Neural Network, Animal-like Character Heads.

1 Introduction

How to quickly generate 3D models remains a widely concerned problem in the field of computer graphics. While there exist professional 3D modeling software packages, such as Maya and ZBrush, they are designed for skillful users. In contrast, due to the simplicity of sketching, sketch-based interfaces for 3D modeling are potentially more accessible by users, even for those with little training in drawing. However, inferring 3D geometry from sparse sketches is a very challenging task due to inherent ambiguities. Traditional approaches [1], [2] allow users to draw or manipulate curve handles (e.g., contours and feature lines) of the target shape and reconstruct a 3D model by using heuristic geometric constraints like continuity. This usually requires a heavy amount of user interaction to produce a complex model.

Recently, deep learning techniques have been utilized in building the mapping from 2D line drawings to 3D shapes, greatly improving the efficiency of sketch-based modeling. For example, with the help of advanced volume-based shape decoders (which are built upon 3D Convolutional Neural Networks (CNNs)), the system proposed in [3] can instantly convert users' rough sketches into 3D shapes. However, the high computational cost of 3D convolutions hampers it from producing high-resolution models. Turning the problem to be a task of generating normal/depth images from multi-view sketches can improve the quality of synthesized models, but existing systems in this track either tend to output results with inconspicuous details [4], [5] or require careful drawings in multiple views [6]. These challenges become easier to tackle if we focus on the shapes of specific categories, e.g., urban models [7] and human faces [8], which can be represented in parametric spaces. A deep regression model can thus be exploited for generating quality models.

In this paper, we aim to design an interactive modeling system for a new shape category - animal-like character heads. This type of model is very popular in character modeling. However, different from human faces, the variety and complexity of animal heads make it difficult to represent them in a low-dimensional parametric space. This greatly prevents deep regression models from getting accurate modeling (Please refer to Sec. 6.2.2 and Fig. 16). Thanks to *Pixel2Mesh* [9], the shape space can also be implicitly modeled by learning the deformation from a template mesh. More specifically, given an input sketch and a template mesh, it first extracts features from the sketch image and then projects them onto the vertices of the mesh, which are fed into a Graph Convolutional Neural Network (GCNN) to learn the deformation. The Chamfer distance (CD) is used as the primary loss to minimize the closeness of the deformed mesh with the ground truth, forming a supervised learning mechanism. Unfortunately, in our setting this method is difficult to generate fine details, making synthesized results still far from satisfactory. This is due to two issues: 1) graph convolution, as an approximated convolution operation de-

- *D. Du and L. Liu are with the University of Science and Technology of China. X. Han, F. Wu, and S. Cui are with the Chinese University of Hong Kong, Shenzhen, and Shenzhen Research Institute of Big Data. H. Fu is with the City University of Hong Kong. Y. Yu is with the University of Hong Kong.*
- *This work was done during D. Du visiting Shenzhen Research Institute of Big Data and the Chinese University of Hong Kong, Shenzhen.*
- *Corresponding author: X. Han (hanxiaoguang@cuhk.edu.cn).*

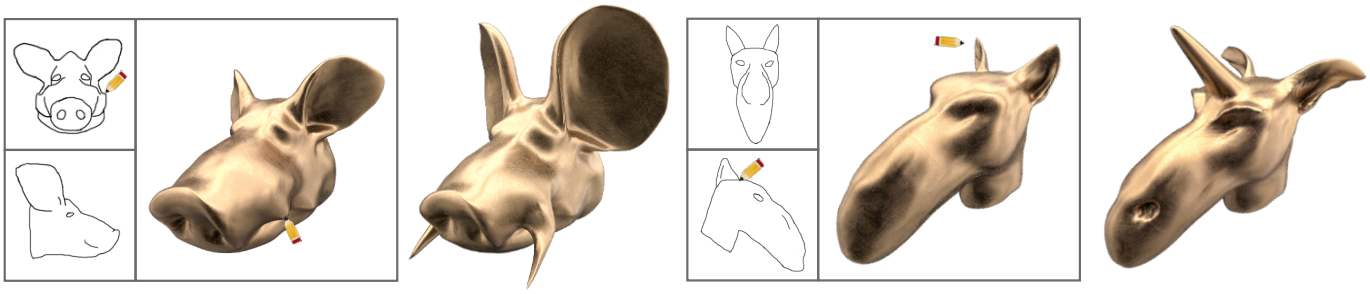


Fig. 1: We present *SAniHead*, an interactive sketching system for amateur users to easily model animal-like character heads. For each example, our novel view-surface collaborative mesh generative network turns hand-drawn sketches in dual views (Left) into a 3D model (Middle) in real time for further editing (Right). On average, the modeling process for each example takes around 8 minutes, including 3 minutes for sketching and 5 minutes for post-manipulation.

fined over a discrete surface representation (i.e., mesh), is hard to model the features of high-frequency signals; 2) since the used CD loss defines the measurement on sparse vertices, it tends to prevent generating geometric details, which contribute only a small portion to the loss value.

As witnessed, recent image decoders based on CNNs can easily infer high-frequency details [10], [11], [12], [13]. Inspired by this, to address the above-mentioned issues, our key idea is to generate a coarse shape using surface-based decoders and synthesize geometric details via image-based decoders, as illustrated in Fig. 2. Specifically, our approach firstly exploits a modified method of *Pixel2Mesh* [9] to convert input sketches depicting a target shape in one or two views [1] to a coarse mesh. To synthesize the details, we project the coordinate information of its vertices into the front, left-side, and right-side views, forming three images. Commonly used image-to-image translation frameworks, such as *pix2pix* [14], can then be utilized to infer pixel-wise displacement maps of vertices from the images. Due to possibly inconsistent inference across views, directly projecting the generated displacements back onto the surface would lead to artifacts on the deformed mesh. We thus propose a novel GCNN and apply it on the surface to reduce the artifacts and refine the geometry. The presented *view-based detail synthesis* and *surface-based detail refinement* can also be conducted in an alternating manner to further improve the quality of synthesized models. Our whole network, called as view-surface collaborative mesh generative network, performs in an end-to-end fashion, where the view and surface domains collaboratively help generating the final mesh. Both qualitative and quantitative results show the effectiveness of the proposed method and its superiority over all existing approaches for shape inference.

Finally, we design a two-stage sketching interface for real-time shape modeling. As illustrated in Fig. 1, in the first stage, users can sketch a target shape from one or two views. Our system instantly generates and updates a 3D model for the input sketch(es). The second stage provides a set of tools, such as handle-based deformation and brush-based sculpting, for directly manipulating the surface. User evaluations show that our system is easy-to-use, especially for novices.

In summary, the main contributions of this work are:

- A novel easy-to-use sketching interface for modeling animal-like 3D character heads. Our system allows users to draw on canvases from scratch to obtain quality synthesized results with instant feedback. It also provides plentiful tools to post-manipulate generated surfaces.
- A novel view-surface collaborative mesh generative network that can effectively recover a 3D mesh from sketches. The proposed algorithm well overcomes the challenges with all existing methods to generate geometric details.
- The first 3D animal head model dataset, containing 220 well-collected models in 19 animal categories. The number of further augmented models is 16,800. Each model is also associated with line-drawing renderings in orthographic views, and 1,200 models even with hand-drawn sketches. We will make the dataset as well as the interactive system public to the research community.

2 Related Work

Sketch-based Shape Design. How to easily create 3D shapes via a sketching interface is a long-standing research problem in the field of computer graphics [15], [16]. One of the seminal works is [17], which presented an approach for using curves to control 3D surface deformation. Igarashi et al. [1] proposed an interactive sketching system that automatically generates a 3D shape from 2D user-drawn silhouettes in real-time. A large body of follow-up works [2], [18], [19], [20], [21], [22] were then conducted. The strokes drawn by users in these systems are adopted to directly constrain the positions for certain vertices of a target shape. The recent works [23], [24], [25], [26] attempted to dig out more geometric information from input 2D sketches by studying the relationship between cross lines and surface normal. Since input strokes only provide sparse constraints on certain vertices, geometric assumptions like surface smoothness are often utilized for surface recovery in other regions. However, since no strong geometric assumptions can be made for animal-style models, these works tend to fail in our setting.

1. Our system provides two canvases, front view and side view. Users can draw on either or both. On the canvas of the side view, the user can only draw under one of the left-side or the right-side view, and the sketch on the other side is directly obtained via flipping the drawn one.

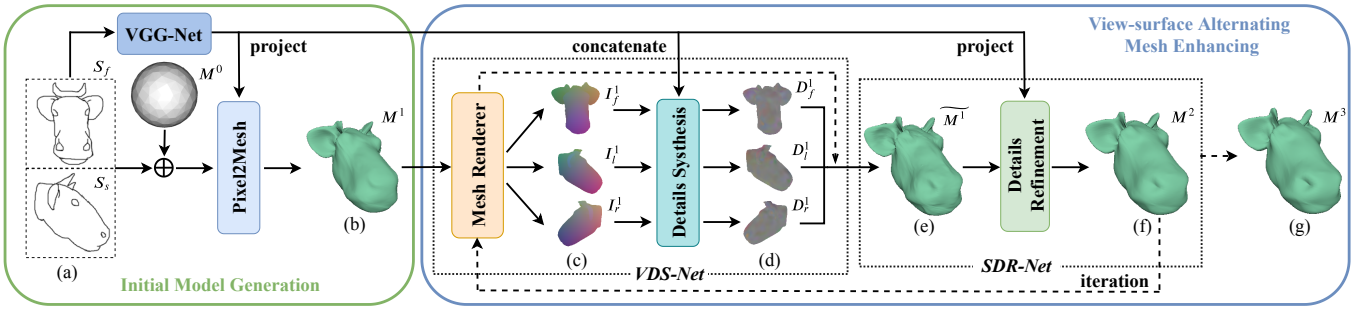


Fig. 2: The proposed view-surface collaborative mesh generative network, which can be trained in an end-to-end fashion. (a) Given dual-view sketches (S_f, S_s) and a pre-defined sphere template mesh M^0. (b) The stage of initial mesh generation produces a coarse mesh M^1, using a modified *Pixel2Mesh*. (c) A differentiable mesh renderer is then used to render M^1 into three vertex maps in the front, left and right views (denoted as I_f^1, I_l^1, and I_r^1, respectively). (d) An image-to-image translation network is then applied on the three images, generating three vertex displacement maps (D_f^1, D_l^1, and D_r^1). (e) The displacements are then unprojected onto the surface, resulting in a new mesh $\widetilde{M^1}$, which contains more geometric details. (f) A GCNN is further applied on $\widetilde{M^1}$ for detail refinement to produce a mesh M^2. (g) After several (2 in this example) iterations of applying *VDS-Net* and *SDR-Net*, we obtain the final mesh M^3. Note that, the input sketches are also used to provide information for both *VDS-Net* and *SDR-Net*. See Section 3 for more details. Our network is also applicable for single-view inputs.

Sketch-based shape retrieval [27], [28] uses an input sketch as a query to search for geometrically the most similar shape from a predefined database of 3D models. However, the results are usually far from input sketches due to the limited capacity of the database. For the objects that can be well-represented by the part assembly, like man-made models, several works [29], [30], [31], [32], [33] proposed to utilize strokes to retrieve appropriate part candidates for subsequent assembly. Compared to sketch-based part assembly, our approach is superior in three folds: 1) our system allows sketching in an arbitrary order with no requirement on the part information, thus greatly improving the efficiency of user interaction; 2) our system does not require geometry composition between parts, which often causes artifacts; 3) a database of pre-segmented 3D models, which requires a heavy amount of labor for manual segmentation, is not needed in our system.

Image-based Shape Decoders. Recently a lot of methods attempted to leverage the powerful learning abilities of deep neural networks for decoding 3D geometry from 2D images. Some works [34], [35], [36] took a volumetric representation of CNNs for the generation of 3D shapes. However, because of the high computational cost, such methods are hard to produce high-resolution outputs, though this issue has been alleviated by Octree-based CNNs [37] and its adaptive version [38]. Point cloud, as one of the most popular geometry representations, has been utilized in [39], [40] as the output formula for shape recovery by training a deep regression model. There also exist some studies [41], [42] on reconstructing 3D shape by synthesizing multi-view depth maps or a single spherical map. However, it is still difficult to produce a clean mesh with all the above-mentioned approaches. Some recent methods [9], [43], [44] attempted to directly learn a 3D shape with a mesh representation as the output. They all take extra template meshes as inputs and then focus on learning mesh deformations. Since the shape of animal heads is homeomorphic to the ball, *Pixel2Mesh* [9] is very suitable for modeling the shape space by learning deformation from a sphere mesh. Unfortunately, it is hard to generate geometric details. Our work tackles this challenge by presenting a novel view-surface collaborative mesh decoder, which is capable to model not only diverse shapes but also fine details.

Deep Sketch-based Modeling. Sketch-based modeling often requires semantic interpretation of freehand sketches. The technique of CNNs is suitable for this task due to its powerful ability of high-level feature extraction. In this context, Nishida et al. [7] trained a CNN-based regression model to map input sketches to the parameters of procedural modeling for buildings. They also developed an interactive system for real-time modeling. Huang et al. [45] proposed a similar network to infer procedural modeling of man-made objects. By incorporating a deep regression model to reason the parameters of 3D face morphable model from sketch images, a user interface for 3D face modeling is developed in [8]. However, these methods only work for the shape categories which can be formulated using a parametric representation. To deal with more general shapes, the work [3] learned an end-to-end network to take a sketch as input and output its corresponding 3D shapes in a volume-based representation. Due to both the computational complexity and the high memory cost, their results are usually of low resolution. An unsupervised learning strategy was also exploited in [46] for the same task. Most recently, by using the efficient image-to-image translation methods, Lun et al. [4] and Li et al. [6] proposed to first transform input sketches into depth/normal maps and then to fuse them to form completed 3D models. The difference is [6] only predicts depth maps from the sketched views, while [4] produces 12 views given one or two input sketches. To the best of our knowledge, our work is the first to implement an end-to-end sketch-to-mesh translation network with deep learning to support interactive mesh modeling. This choice is made for two reasons. First, due to their shape diversity, it is not easy to build a low-dimensional parametric space for animal-like heads, making a deep regression model hard

to result in the parameters for accurate modeling. Second, a large number of view-based depth maps are usually needed to represent the models we are dealing with, causing a heavy amount of user interaction or post-processing with the existing approaches.

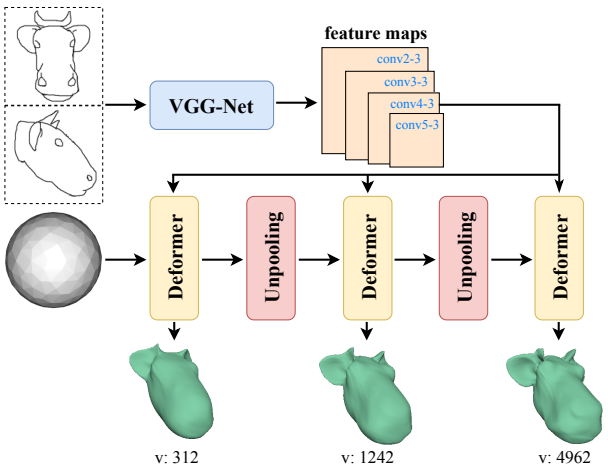


Fig. 3: The proposed modified *Pixel2Mesh* architecture, which is used in our first stage for generating a coarse mesh from input sketches.

3 Mesh Inference from Sketches

Before describing the design of our sketching interface (Section 5), we introduce our algorithm for sketch-based mesh inference, which directly converts sketch images in one or two views to a 3D shape in mesh representation. Note that although our approach theoretically supports inputting sketches in multiple arbitrary views, we set the canvases in only the front view and (left or right) side view because: 1) they are generally sufficient for users to provide the desired shape information; 2) this setting also better matches users' drawing habits. As illustrated in Fig. 2, the proposed architecture consists of two stages: the method of *Pixel2Mesh* is firstly utilized to convert input sketches into a coarse mesh as initialization; geometric details are then refined with a proposed view-surface alternating mesh enhancing mechanism. The details will be explained in the following subsections.

3.1 Initial Mesh Generation

In the first stage, the algorithm of *Pixel2Mesh* [9] is borrowed to covert input sketch images into a coarse mesh. We first give a brief review of *Pixel2Mesh* and then explain how to adapt it to our settings.

Pixel2Mesh. Taking an image I of an object O and a template mesh M^0 (it can be a sphere or an ellipsoid) as input, *Pixel2Mesh* tries to learn a mechanism for deforming M^0 to fit the shape of O. To do so, a coarse-to-fine pipeline is designed, containing three deformation blocks with two unpooling layers intersected between them, as illustrated in Fig. 3. Each deformation block treats the mesh as a graph and applies a GCNN onto it to learn vertex-wise displacements. Through subdivision operations, the unpooling blocks progressively increase the resolution of the deformable mesh. This coarse-to-fine mechanism greatly improves the ability to model complicated shapes. In parallel, a VGG-16 network is used for feature extraction from I. In each deformation block, vertices on the mesh are orthogonally projected into the extracted feature maps of the VGG-16 network in four specific layers ('conv2-3','conv3-3','conv4-3','conv5-3'), to capture the target shape information of O. We recommend the readers to refer to [9] for the detailed architecture setting. Note that, our model takes a sphere mesh with 312 vertices as input and outputs a mesh with 4,962 vertices, which is much denser than the version in [9] where the output mesh contains 2,466 vertices. This is because the shape of animal heads usually contains more geometric details than man-made objects, which are the focus of [9].

Dual-view Inputs. Single-view input is usually not enough to determine a 3D shape, while it is hard to draw consistently for multiple views especially for novices. We thus take a trade-off by using dual-views, say the front view and one side (left or right) view. Since the two views are orthogonal, the inconsistency will produce minor influences on the mesh inference. When a user inputs both a front-view sketch S_f and a side-view sketch S_s (either left-side view or right-side view, please refer to Section 5), we concatenate S_f and S_s to be a 2-channel image I^0 and directly feed it into the network of *Pixel2Mesh*. Fig. 3 illustrates the detailed architecture. It is worth noting that we have also tried an alternative way to deal with dual-view inputs: two VGG-16 networks (with weights sharing) were used to extract features from S_f and S_s independently, which were then concatenated before projecting onto the mesh. However, we found this alternative solution did not provide better results through experimental comparisons.

3.2 View-surface Alternating Mesh Enhancing

Denote the generated mesh from the stage of initial mesh generation as M^1. We found that *Pixel2Mesh* usually fails to infer geometric details. To address this issue, we try to leverage the powerful generation capability of CNNs in the image domain, and propose a view-surface alternating mesh enhancing framework, which consists of two cascaded modules, namely, a view-based detail synthesis network (*VDS-Net* for short), and a surface-based detail refinement network (*SDR-Net* for short).

3.2.1 View-based Detail Synthesis

As shown in Fig. 2, *VDS-Net* firstly uses a mesh renderer to project M^1 into three images from the front, left-side, and right-side perspectives, named as I_f^1, I_l^1, and I_r^1, on which the high-frequency details are inferred via image-based generative networks.

Vertex-map Rendering. We first introduce how the mesh renderer generates vertex maps to form the images. Similar to the definition of a normal map, a vertex map encodes the coordinates of the visible mesh vertices into a colorful image. Sepcifically, given a mesh $M = (V, E)$, where $V = \{v_i, i = 1, ..., N\}$ represents the vertex set and E indicates the edge set, a vertex map from a viewpoint c, denoted as I_c, is calculated: all visible faces are firstly projected onto a plane, forming a 2D mesh; for each vertex p on it (we

assume it is the projection of a 3D vertex $v = (x, y, z)$), we assign an RGB value $(\phi(x), \phi(y), \phi(z)) \times 255$ (ϕ is an operator of normalization, specifically taking x as an example, $\phi(x) = (x - x_{min})/(x_{max} - x_{min})$) to this location; the colors of other pixels inside a face are finally interpolated from the vertices of the associated face using the barycentric coordinates. Note that there exist other ways to represent a mesh in a viewpoint, such as depth maps and normal maps. We select a vertex-map representation because of its simplicity and capability to support end-to-end training (please refer to Section 3.3 for the details about why the vertex-map representation makes this mesh renderer easily differentiable).

Displacement-map Inference. We then discuss how to synthesize details over the three images. Our work solves the problem of detail synthesis on M^1 by learning desired movements for its vertices. For each vertex v_i, we need to infer a displacement vector d_i. Such displacements defined over the surface can also be represented by three 2D color-maps (along the x-axis, y-axis, and z-axis), named as *displacement-maps*, which can be created in a similar way to the method for rendering vertex maps. We denote the resulted color-map images from the front, left-side, and right-side views as D_f^1, D_l^1, and D_r^1, respectively. In this manner, the problem becomes how to infer D_p^1 from I_p^1, where p stands for one of the three viewpoints (i.e., front, left-side, and right-side). We adopt the commonly used *pix2pix* model [14] for this image-to-image translation problem: a U-Net is used for image translation and a conditional discriminator is adopted to better generate sharp details. Note that, the same pix2pix model is utilized for all different views.

Sketch Assistance. To provide sufficient information of local details, for each viewpoint $p \in \{f, l, r\}$, the corresponding sketch S_p is also integrated into the translation procedure from I_p^1 to D_p^1. In particular, we first use a sequence of *conv+pool* layers to convert S_p to a feature map, and then concatenate it to the bottleneck feature of the U-Net, assisting for detail decoding. It is worth mentioning that besides a sketch in the front view, our system only requires either a sketch in the left-side view s_l or a sketch in the right-side view s_r. To support sketch assistance for all views, we simply treat S_l (S_r) the mirror image of S_r (S_l), which is also helpful to generate a symmetric shape. Note that we have another option to use only 2 views for detail enhancement: for example, we can firstly generate only D_f^1 and D_l^1, and then obtain D_r^1 as the mirror of D_l^1. However, based on our experiments, directly applying such D_r^1 to M^1 tends to produce severe artifacts. This is mainly because of the misalignment between the mirror of D_l^1 and the right-side of M^1, since *Pixel2Mesh* usually generates asymmetrical outputs. We also extend our system for the single-view sketch input by synthesizing the line drawings of the mesh generated by single-view *Pixel2Mesh* module for unavailable views with the method of Suggestive Contours [47], then fetch them to *VDS-Net*.

3.2.2 Surface-based Detail Refinement

After obtaining the displacement maps D_f^1, D_l^1, and D_r^1, we back-project them onto M^1 to guide the movement of its vertices for detail generation on the surface. This step gives rise to a new mesh $\widetilde{M^1}$ with details enhanced. Due to the possible projection errors and the inconsistency of the synthesized details in different views, $\widetilde{M^1}$ tends to be not very smooth. A GCNN defined on the mesh is further trained to eliminate such noise and also help to refine the generated geometric details.

Displacement Unprojection. We perform the unprojection by projecting each vertex onto the 3 displacement maps to obtain the corresponding displacement vectors. Some vertices might be seen from more than one view, resulting in more than one displacement vector. We simply average them to form the final movements. Specifically, for each vertex v^1 of M^1, we project it to D_f^1, D_l^1, D_r^1 and respectively get the corresponding color values, which are directly converted to 3D displacement vectors $\mathbf{d}_f^1$, $\mathbf{d}_l^1$, $\mathbf{d}_r^1$. If the projected point in view p lies outside the valid region of D_p^1, we set $c_p = 0$ (c_p is a binary variable that indicates if p lies inside the valid region) and reset $\mathbf{d}_p^1 = \mathbf{0}$; otherwise, we set $c_p = 1$. Then, we add the average displacement vector $\mathbf{d}^1 = (\mathbf{d}_f^1 + \mathbf{d}_l^1 + \mathbf{d}_r^1)/(c_f + c_l + c_r)$ to v^1, obtaining $\widetilde{v^1} = v^1 + \mathbf{d}^1$. This thus forms $\widetilde{M^1}$.

GCNN-based Refinement. We consider $\widetilde{M^1}$ as a graph and train a GCNN. It takes vertex coordinates of $\widetilde{M^1}$ as input signals over the graph nodes and uses 12 *graph-conv* layers to extract vertex-wise features, which are followed by a regression layer and converted to 3D vertex-wise displacements. By adding the generated displacements onto $\widetilde{M^1}$, we obtain a smoother mesh M^2. Using the ground-truth models as the supervision to train this GCNN not only reduces the noise but also enhances details.

Sketch Assistance. To take full advantage of the input sketches, the extracted VGG-features of them are also integrated into the GCNN-based refinement network, further providing desired detail information. To achieve this goal, we directly project the feature map of 'conv5-3' to the second *graph-conv* layer.

3.2.3 Cascaded Mechanism for Further Enhancement

Note that our view-based and surface-based detail synthesis methods are both based on the input, this mechanism supports incremental learning naturally. Inspired by the algorithm of [48], we propose two cascaded refinement strategies to further improve the results.

Cascaded Displacement Inference. The first strategy adopted in the task is translating I_p^1 to D_p^1. We modify the generative network of the pix2pix model to perform in a cascaded manner: the output of the U-Net is directly concatenated to I_p^1, followed by another U-Net to generate D_p^1. This procedure is conducted repeatedly. In our implementation, we use two cascaded U-Nets since it performs better than a single U-Net and obtains similar results to three U-Nets, based on our experiments.

Cascaded Detail Inference and Refinement. We connect the modules of *VDS-Net* and *SDR-Net* in a cascaded way and perform them repeatedly. In particular, a *VDS-Net* converts the initially generated M^1 to $\widetilde{M^1}$, which is then mapped to M^2 by a *SDR-Net*. After that, another *VDS-Net* can be utilized to translate M^2 to $\widetilde{M^2}$, which is used as input to another *SDR-Net* to get M^3. Our experiments show that adopting two iterations strikes a good balance

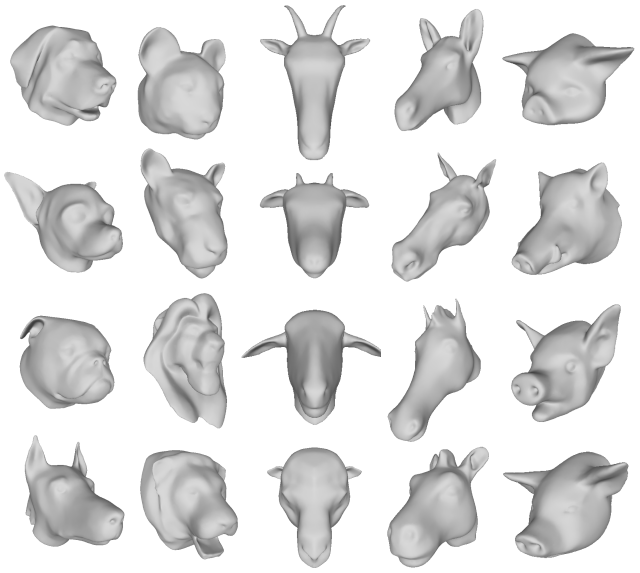


Fig. 4: Some sampled models in five sampled animal categories, in our dataset, are shown.

between quality and efficiency. M^3 is the final output of our implementation.

3.3 End-to-End Training

Our framework is finally formed as: *Pixel2Mesh* $\rightarrow$ *VDS-Net* $\rightarrow$ *SDR-Net* $\rightarrow$ *VDS-Net* $\rightarrow$ *SDR-Net*. In this section, we discuss how to train the entire network.

Differentiable Renderer. The key process to guarantee end-to-end training is to design a differentiable layer to project a triangle mesh M to be a vertex-map I, establishing the correspondence between M and I. Then, the loss back-propagation is automatically accomplished using the automatic differentiation library, e.g., TensorFlow. In our setting, the value of a pixel $p \in I$ is the coordinate which can be calculated by a linear combination of the vertices of the corresponding face of M. Therefore, the loss can be easily back-propagated passing through this projection layer, given the known projection matrix (one of the front, left-side, or right-side).

Training Procedure. In Section 4, we will describe the details of how to prepare a dataset of meshes for training and a method to generate input sketches S_f and S_s (S_l and S_r are randomly synthesized when generating training samples) for each mesh M^g. Our whole network takes input sketches and tries to deform a template mesh to fit M^g. The training is conducted in the following steps:

1) We first train *Pixel2Mesh* separately, which takes S_f and S_s as input and outputs a coarse mesh M^1. The loss functions are the same as [9], including Chamfer distance (CD) loss measuring the closeness of the output mesh to the ground-truth mesh M_g, normal loss ensuring local orientation consistency between the output and the ground-truth, Laplacian loss encouraging the smoothness of the output, and finally, edge length loss penalizing the flying vertices.

2) The first *VDS-Net*, which takes I_p^1 as input and outputs D_p^1, is also trained separately. We use the L1 loss to measure the closeness of D_p^1 with the ground-truth displacement map D_p^g, and cGAN [14] loss to make the output map as sharp as possible. To support the training, we generate the paired data (I_p^1, D_p^g) as follows: For each training sample, the trained *Pixel2Mesh* is first applied to generate M^1, which is then directly rendered into three views, producing $I_p^1, p \in \{f, l, r\}$. After that, the popular ICP strategy is used to register M^1 to M^g. This builds a vertex-wise correspondence map, with which each vertex on M^1 can easily get a 3D displacement vector to reach M_g. D_p^g can thus be projected from the displacement field.

3) For each training sample, we use the trained *VDS-Net* to convert M^1 to be $\widetilde{M^1}$. Then, the paired data $(\widetilde{M^1}, M^g)$ are used to train *SDR-Net* and obtain M^2, where all of the four loss functions in Step 1 are used.

4) Finally, we concatenate all sub-networks in a complete one and fine-tune them in an end-to-end manner. Before fine-tuning, we first copy the parameters of *Pixel2Mesh*, the first *VDS-Net* and *SDR-Net* from their pre-trained models, and directly set the parameters of the second *VDS-Net* and *SDR-Net* as the same as those of the first *VDS-Net* and *SDR-Net*, respectively.

4 Dataset Construction

To support the training of our network, we built a dataset of 3D animal heads and their corresponding 2D sketches. The procedure is described in the following subsections.

4.1 3D Model Collection

We first collected 3D animal models, not restricted to heads, from the Internet. Due to the limited categories of the publicly available animal models, we also recruited two artists to design uncommon animal heads. Finally, we obtained 220 models in 19 categories, including cat, dog, rabbit, camel, etc.

Mesh Reconstruction. The models collected from the Internet are usually of non-manifold structures. Moreover, they often involve undesired internal structures. To obtain the shell of a mesh, we process it in four steps: 1) we render the model into multi-view depth maps (20 views used); 2) the depth maps are fused to form a point cloud; 3) a mesh is reconstructed using the method of Poisson mesh reconstruction [49]; 4) a skilled artist is recruited to clean the reconstructed model by removing artifacts and enhancing feature regions, such as eyes, mouths, noses, and ears. Finally, we obtain a closed manifold mesh.

Head Cut and Alignment. For each model with a full-body, we cut its head manually along the neck and complete it to be a closed mesh. After that, all meshes are further aligned manually to have a consistent scale and orientation. Fig. 4 shows several models in sampled categories (all the category names are listed in the supplemental material, where some sampled models for each category are also shown).

4.2 Mesh Augmentation

To improve the generalization ability of our method, we increase the models' diversity for data augmentation using the following three approaches.

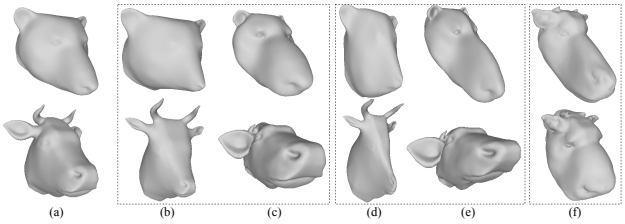


Fig. 5: Mesh augmentation. (a) two sampled meshes; (b)(c) after surface-domain deformations with two random Laplacian scaling; (d)(e) two further random space-domain deformations applied to the meshes in (b)(c). (f) shows two resulted models by first interpolating the two models in (a) randomly and then perform a random surface-domain deformation followed by a random space-domain deformation.

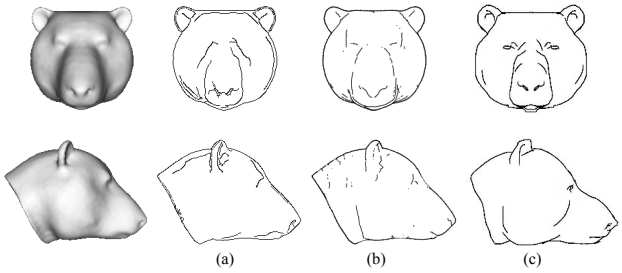


Fig. 6: Given a model, we collect its sketches by (a) Canny edge detection; (b) suggestive contour rendering; and (c) human drawing.

Surface-domain Deformation. Inspired by the method used in [8], we firstly perform surface-domain deformation using mesh exaggeration techniques as follows. The Laplacian of the i-th vertex v_i on a mesh can be defined as $Lap_i = \sum_{j \in N(i)} w_{ij} ||v_i - v_j||_2^2$, where w_{ij} follows the well-known cotangent weight formula [50]. Thus, the exaggeration can be carried out randomly by applying a random scaling value on all Lap_i and obtaining the scaled Laplacians, from which the deformed mesh can be recovered by solving a sparse linear equation system. We perform 7 times of random deformation per model, then extend the 220 models to 1,760 ones.

Space-domain Deformation. To further increase the model diversity of the dataset, for each mesh from the 1,760 ones, we perform 4 extra random deformations in the space domain by using the t-FFD approach [51]. 8,800 meshes are finally obtained. The results after surface-domain and space-domain deformations on two sampled meshes are shown in Fig. 5(b-e).

Model Interpolation. As seen, our collected models are of extremely diverse shapes, making the established space sparse and hyper non-uniform. This greatly limits the generalization capability of the trained model. To address this issue, we further augment our dataset by interpolating the models. This is conducted in three steps: 1) for each model from the 220 animal heads, we manually annotate 7 landmark positions: left and right ear tips, left and right eye centers, nose tip, upper and lower lip mid-point (please refer to the supplemental material for the illustration); 2) based on the labeled landmarks, the algorithm of [52] is exploited to build dense correspondences for all the 220 models; 3) 8,000 paired models are then randomly selected. For each pair, one model is created by firstly applying a random interpolation and then performing a random surface-based deformation followed by a space-domain deformation. Two sampled interpolated models are shown in Fig. 5(f). Finally, 16,800 models are obtained.

4.3 Sketch Collection

For each mesh in the dataset, we synthesize its sketches on both front and side views (including left-side and right-side) for building sketch-mesh pairs.

Sketch Synthesis. Given a mesh, we firstly render its line drawings under specific viewpoints using the method of Suggestive Contours [47]. To increase the diversity, another rendering technique is also adopted: the shading map is firstly rendered and then Canny edges [53] are extracted as the sketches.

Hand-drawn Sketches. We also recruited 3 artists to draw the sketches for 1,200 randomly selected meshes in an image-tracing setup. For each model, both front-view and side-view (left or right is randomly chosen by the artist) drawings were created. It is worth mentioning that, as these hand-drawn sketches in different views are usually inconsistent. Training with them can improve the robustness of our network. Fig. 6 shows both the synthetic sketches and hand-drawn sketches for a sampled mesh in two views.

5 User Interface

In this section, we introduce the design and implementation of our user interface. A two-stage sketching mode is designed for coarse-to-fine modeling. Both a mouse and a graphics tablet are allowed for sketching inputs, while a pen tablet is highly preferred for its input accuracy. Fig. 7 shows a complete modeling sequence for a representative example. Please also refer to the attached video for the demonstration of the interaction procedure, where a novice user, who has little painting experiences and is also unfamiliar with our system, is invited to conduct the sketching.

5.1 Sketching on View

To start with, our system provides two empty canvases: one for front-view sketching and the other for side-view sketching. As illustrated in Fig. 7, the front view of an animal head stands for the perspective of its front face and the side view can be used for either the left-side or right-side viewpoint. We trained two models: single-view and dual-view shape inference. Our system automatically chooses the right trained model according to the canvas(es) the user is interacting with.

Single-view Sketching. Users can choose either canvas to start the drawing. Both drawing and erasing operations are allowed and can be performed in an arbitrary order. After each operation, the shape is instantly updated and displayed on the 3D view panel. When drawing on the side-view canvas, as shown in Fig. 7, either left-side sketching or right-side sketching is allowed.

Dual-view Sketching. Usually, single-view drawing is not enough to faithfully depict a target shape. Our system provides the dual-view sketching mode, where the user

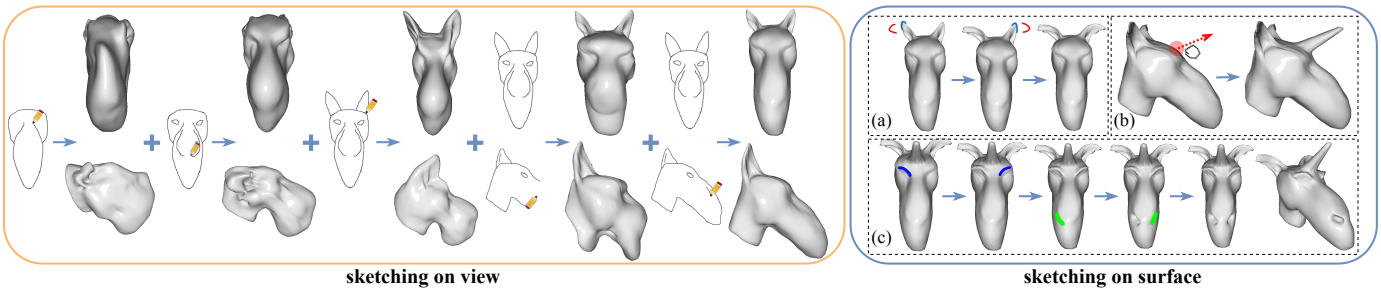


Fig. 7: Using our interactive sketching system, the user firstly sketches on the front-view canvas and then the side-view canvas, where the predicted 3D shape is updated and displayed in real-time. After that, sketching on the surface is also allowed for direct mesh manipulation. Three manipulation operations are provided: (a) editing with curve-handles; (b) editing with vertex-handles; (c) surface sculpting.

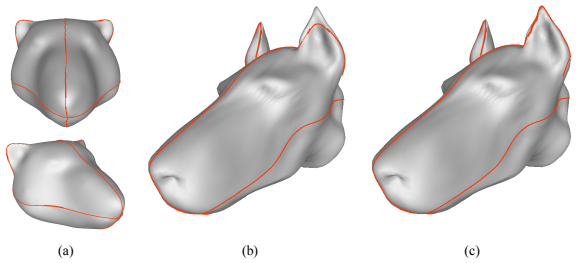


Fig. 8: (a) We define several curve handles on a template model; (b) During sketching procedure, once a mesh is inferred, the curve handles are simultaneously provided; (c) The pre-calculated curve handles will be automatically tuned, by snapping onto high curvature lines.

can draw on both the front- and side-view canvases. To avoid potential conflicts between two different views, an alternative 2D line drawing of the synthesized 3D model can be rendered with the Suggestive Contour method [47] in the new view before canvas switching. This is only performed for the first-time switching, to avoid the conflict between the synthesized sketches and the existing ones when the user moves back to the previous drawing canvas. Therefore, the user only needs to make changes in the rendered drawing. Note that the left-view is set as default for side-view sketching. When users wish to draw in the right view, they need to press a shortcut-key to signal an explicit view change. This is because the VDS-Net requires the view information for conducting projection. In this mode, the 3D shape is updated according to both two input sketches.

5.2 Sketching on Surface

To increase the controllability of the target shape for users, our system also provides extra sketching tools for directly manipulating the surface. Two tools are provided: handle-based editing and brush-based sculpting.

5.2.1 Handle-based Editing

Similar to [2], when performing handle-based editing in our system, users firstly select a vertex-handle or curve-handle together with a region-of-interest (ROI), then deform the surface inside the ROI by moving the handle.

Editing by Vertex-handles. Our interface allows the user to select a vertex-handle by a mouse click and perform editing in a drag-and-drop fashion. The ROI is determined with a geodesic disk centered at the picked vertex, where the radius has a default value and can be fine-tuned by the user.

Editing by Curve-handles. Before editing the surface geometry with a curve-handle, users need to specify a curve on the 3D surface. To do so, users are asked to firstly pick a view and then directly draw a curve on the 2D screen. Our system then automatically locates the curve on the surface by intersecting with the overlapped triangles. Note that, as in [2], this will result in some new vertices and produce a new meshing. We provide two ways for manipulating, thus the user can 1) directly drag one of the vertices on the curve to define the deformation, the same as [2]; 2) pick an appropriate view and draw a new curve to define the target position for the vertices on the curve-handle, similar to [8]. Our system also gives a default setting for determining the ROI region of a curve-handle, which is also allowed to be tuned.

Predefined Curve-handles. Practically, users tend to pick some feature lines of an animal head as the curve-handles, such as ear contours. However, manually selecting these 3D curves usually requires careful and time-consuming interactions. To alleviate users' loads, our system provides some predefined curve-handles together with the generated 3D model, as illustrated in Fig. 8. The key issue to implement this function is how to infer topology-consistent meshes for different sketch inputs, that is, the vertices on the same semantic feature lines should share mesh-ids across different models. Thus, we fine-tune our model: 1) pick one of the resulted meshes of our trained network in the training dataset and annotate the 7 landmark vertices (as in Sec. 4.2); 2) fine-tune the mesh inference network with an extra loss function, to constrain the landmark vertices of the generated mesh targeting the labeled ones on the ground-truth surfaces. Although such a fine-tuning operation produces more consistent topology, the pre-calculated handles are still not accurate, as shown in Fig. 8(b). We further optimize the curves by snapping them onto high curvature regions, to better match sharp feature lines. Besides, our system provides a tool for users to directly modify the position of pre-calculated curve-handles.

Real-time Editing. After determining the vertex-handles or curve-handles and their ROI regions, a Laplacian matrix is constructed and pre-decomposed for further real-time editing, as implemented in [55].

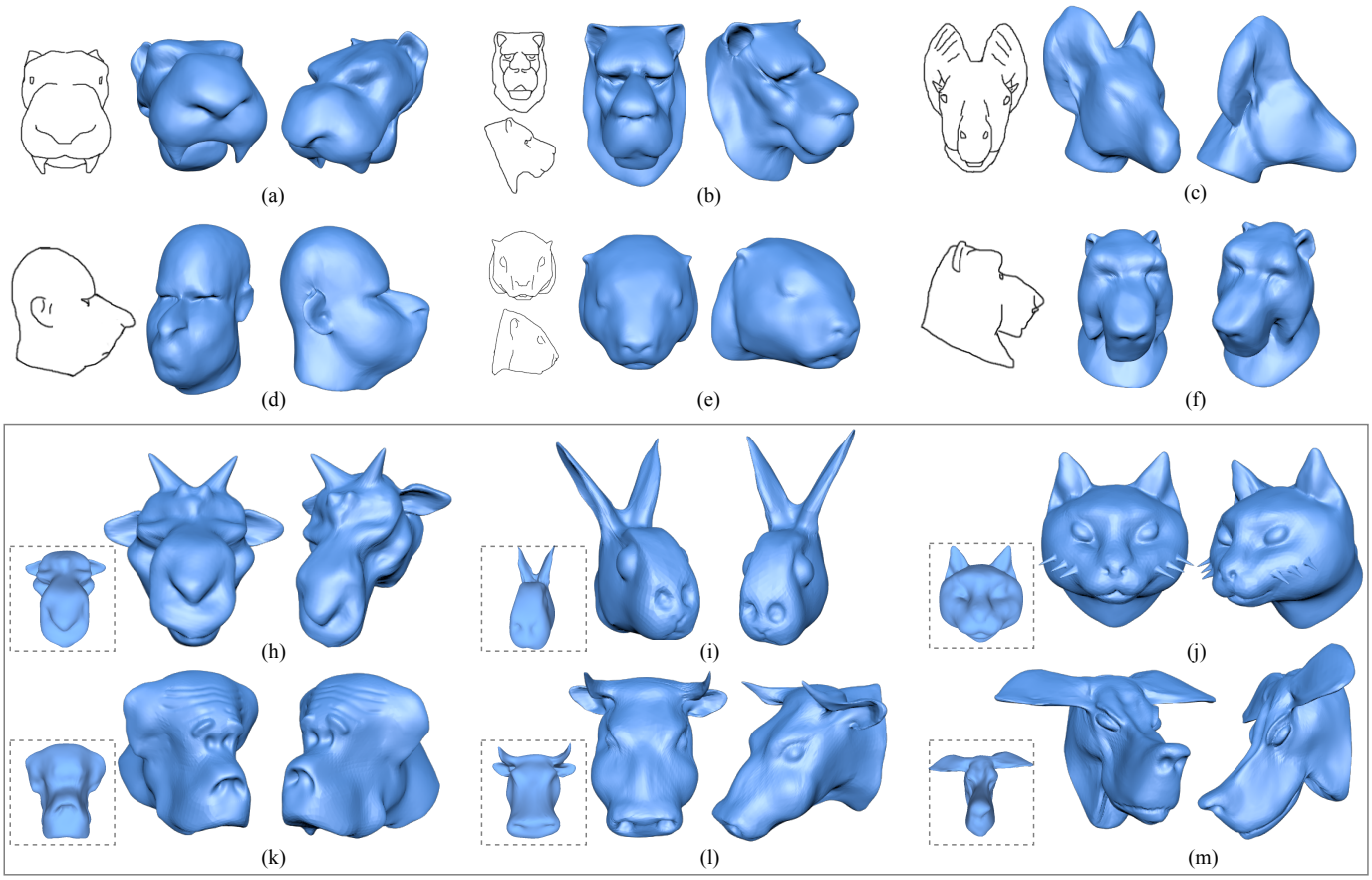


Fig. 9: Gallery results of our proposed sketching system. The models (a)-(f) are created by amateur users, who have no modeling experience but own good drawing skills, using only the mode of sketching on views, without post-manipulation. Specifically, for the dual-view sketching, the users could choose to render the other view of a 3D model synthesized based on the sketch in one view to assist the drawing. On average, around 3 minutes were need to create each model. The models (h)-(m) are produced by a user who is familiar with our system. Both sketching on views and sketching on surface are used. It took around 10 minutes to design each model on average. For reference, we provide the models (in the dotted boxes) immediately inferred by our network.

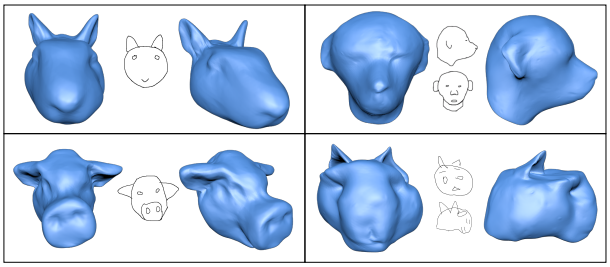


Fig. 10: We test the proposed mesh inference algorithm on freeform sketches drawn by a child to further demonstrate its generalization capability. The left two cases are generated with single-view input, while the right ones are inferred from dual-view sketches.

5.2.2 *Brush-based Sculpting*

Borrowing from [56], our system also provides both extrusion and engraving brushes for surface sculpting. To avoid artifacts, a subdivision is performed online for the ROI region when applying these brushes on the surface. The method of [57] is adopted for subdivision. A brush for smoothing is also provided.

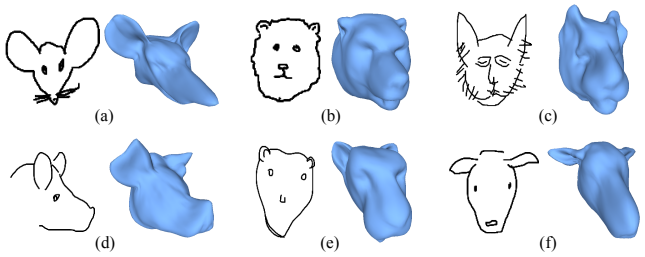


Fig. 11: The evaluation of our method for single-view inputs with the Sketchy database [54]. The results are sampled from the categories of (a) mouse, (b) bear, (c) cat, (d) pig, (e) lion, and (f) sheep.

6 Results and Evaluation

Our sketching system has been fully implemented on a PC supporting both mouse and pen inputs. We used TensorFlow to train and test our mesh inference network. Specifically, we trained the whole model for 50 epochs with a learning rate of 10^{-4}, which took around 2 days. 90% sketch-mesh pairs are utilized as the training set and the remaining for testing. Each forward of the mesh generator takes 70 ms on average on a 3.6GHz Intel processor with a

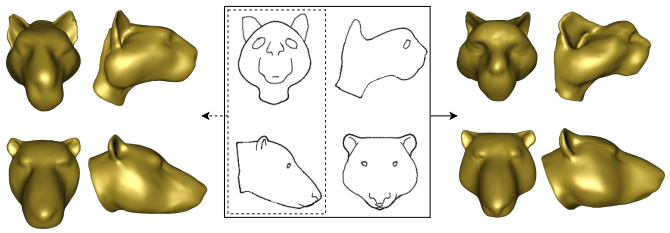


Fig. 12: Visual comparisons on shape inference with single- or dual-view sketches. For each model, two views are shown. The models in the left-side are generated by single-view sketches inside the dotted box, while the models in the right-side are inferred with dual-view sketches.

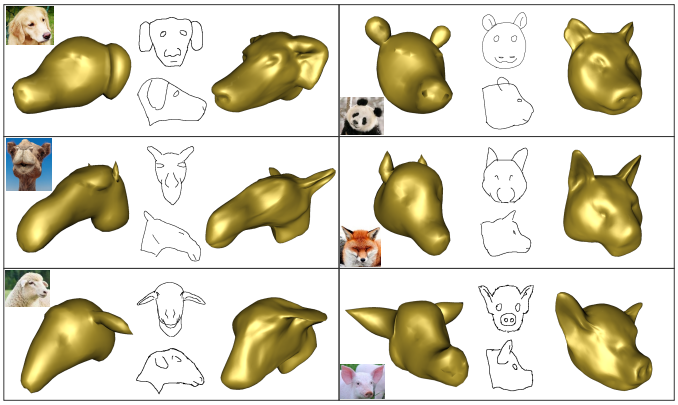


Fig. 13: Six amateur users were invited to conduct modeling using both the Fibermesh and our system, for comparison. Each user was given a referenced image and asked to create a similar shape. For each user, the created model with Fibermesh is shown in the left-side while the one created using our system is displayed in the right-side. The drawn dual-view sketches when using our system are also shown.

GeForce TITAN Xp GPU. In the mode of sketching on the surface, each deformation operation costs on average 50 ms, supporting real-time editing.

6.1 Qualitative Results

Results Gallery. The first two rows in Fig. 9 show several representative results of our sketch-based mesh inference algorithm. For each example, we show the input sketch(es) and the resulting 3D model in two viewpoints. Some of the examples required only single-view sketches as input, while the others take dual-view sketches as input. It shows that the inferred 3D shapes all resemble the input sketches well. More modeling results of 3D animal-like heads are shown in the bottom two rows, where sketching on the surface is utilized for mesh manipulation. As seen, our system allows the creation of the models with diverse shapes. It took around 3 minutes on average for creating one model of the first two rows, and 10 minutes on average for one model of the last two rows.

Single-view VS. Dual-view. In Fig. 12 we use two examples to show the results of our model inference, from single-view and dual-view sketches. For each example, the models inferred from one of the two sketches and both of them are shown. As expected, dual-view sketching is better for users to control the target shapes.

Modeling Capability for Freeform Sketches. To further demonstrate the generalization capability of our proposed mesh inference model, we test it on several freeform sketches that are drawn by a 5-year old child with several months of painting experiences. Fig. 10 shows four examples where two views are displayed per model and the corresponding sketches are also given. The left two cases are generated with single-view inputs, while the right ones are inferred from dual-view sketches. Since there is no notable sketch database for animal heads, we compromise to evaluate our algorithm on the Sketchy database [54], which is the first large-scale collection of sketch-photo pairs and contains human drawings of various animal categories. The sketches involving only head content are selected for our single-view modeling. Fig. 11 shows several representative results. As seen, evenly the line drawings are very casual, the generated models still show reasonable matches with the shape of given sketches.

6.2 Comparisons

We conducted comparisons between our proposed system and state-of-the-art techniques in the following two aspects.

6.2.1 Comparisons on Sketching Interfaces

There exist several systems which are potentially alternatives for modeling animal-like heads. We broadly divide them into three categories based on their user interaction to create 3D models: 1) directly sketching 3D curves on a model surface to control its geometry, such as the operation of FiberMesh [2]; 2) first sketching elaborate patch boundaries and strokes representing the major bending directions, and then generating a curvature field that conforms to the user strokes for the subsequent freeform surface derivation [26]; 3) more freeform sketches are allowed to infer the 3D geometry of various representations (e.g., voxel, depth map, normal map, implicit field, and mesh), sometimes followed by necessary post-processing to optimize the shapes [4], [6]. Our proposed system falls into the third category. However, most of these systems are not available to the public, including [4], [6], [26]. Since the interaction of the systems in the third category is similar and the evaluation of the corresponding methods will be discussed in the following section, we finally choose FiberMesh [2] as the baseline for the comparison. Moreover, we also conduct an informal comparison with ZBrush[2], which is a general-purpose commercial application for creating detailed 3D models.

We acknowledge that both FiberMesh and ZBrush are more general (beyond animal-like heads) than SAniHead. We are interested in exploring our tool specialized for animal-like heads is more powerful for novice users to model shapes in this category than FiberMesh or ZBrush.

Comparison with FiberMesh. Six users were invited to do the evaluation. Two of them had good drawing skills, while others with limited training in drawing. They neither were from a 2D/3D graphics background, nor had any modeling experience. Each participant was asked to use the two systems (i.e., FiberMesh and SAniHead) on a specific

2. https://pixologic.com/

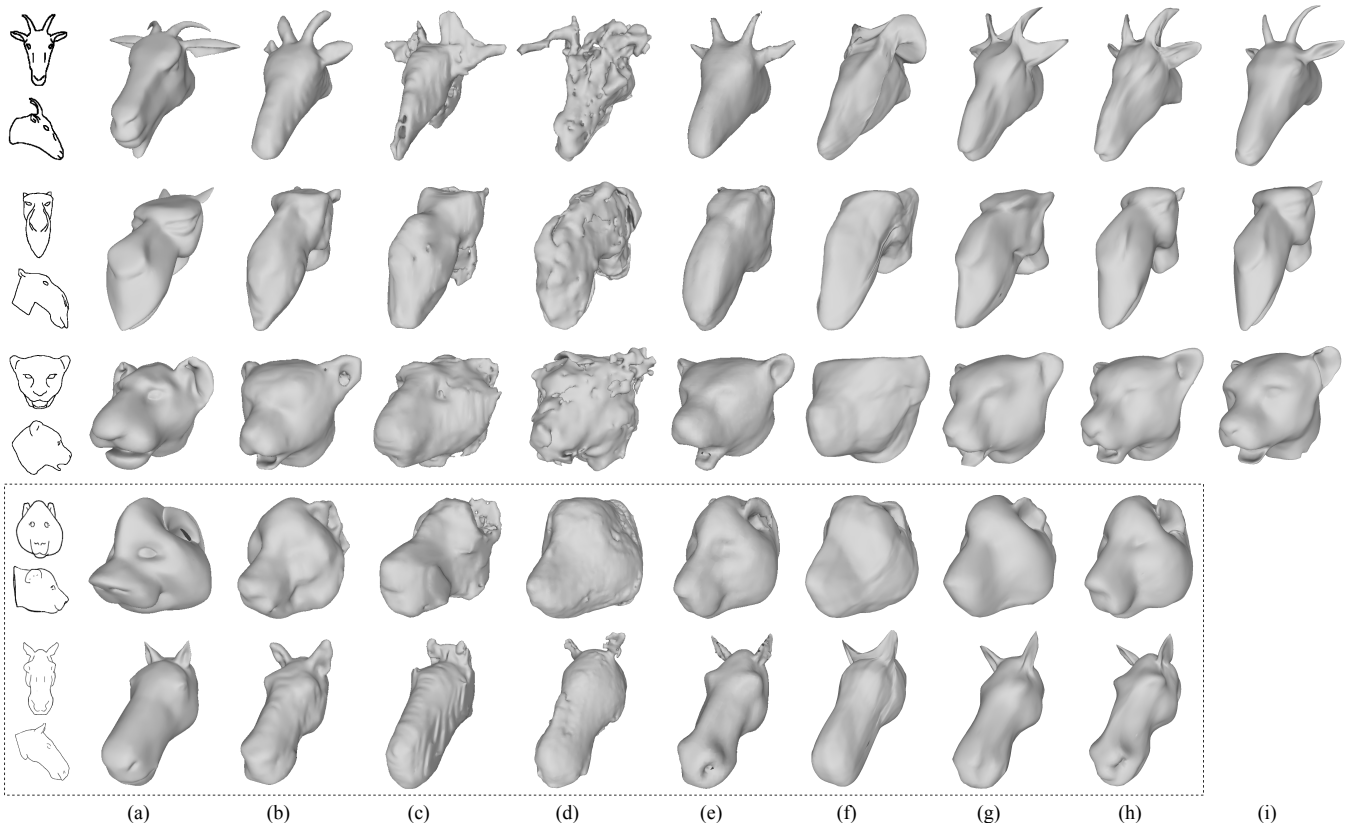


Fig. 14: Qualitative comparisons on our method and existing deepnet-based approaches, given the same dual-view sketches as input. (a) sketch-based shape retrieval [28]; (b) $3dR2N2$ [34] (in 64^3); (c) OGN [37] (in 64^3); (d) $MVRN$ [4]; (e) $DeepSDF$ [58]; (f) $AtlasNet$ [44]; (g) $Pixel2Mesh$ [9]; (h) Ours and (i) Ground truth. For the last two rows, there is no ground truth for the given hand-drawn sketches.

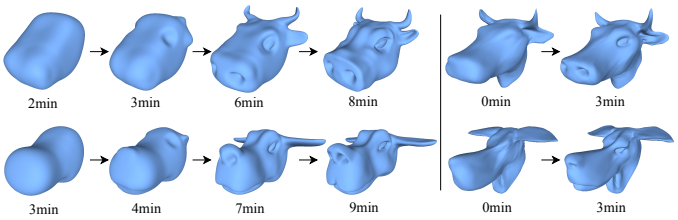


Fig. 15: The models (l) and (m) in Fig. 9 were shown to a skilled ZBrush artist, who was invited to create similar shapes from a sphere using ZBrush. The whole process was recorded. Four pivotal snapshots are shown in the left column with time consumption. We further asked him to reproduce them by adding geometry details to the results generated by *Pixel2Mesh* (shown in the right column) to compare with the synthesis details of our network (shown in Fig. 9).

computer that had been set up ahead. The participant should first randomly choose a system, and then received a tutorial about how to use the chosen system. This took around 20 minutes. Afterward, we randomly provided the participant with an image of an animal head for reference and asked him/her to create a similar 3D model. When the participant completed the modeling, we invited him/her to finish a brief questionnaire to get their feedback on the user experience and modeling difficulty. After a short break, the participant went through the evaluation of the other system with the same procedure except that the reference image was the one given in the first system. This modeling session for each system was terminated after 20 minutes or earlier if the participant was satisfied with the results. The created models using the two different systems are shown in Fig. 13. The input sketches of our system were all dual-view in this evaluation. It can be seen obviously that the models created using our system are visually better and exhibit more animal-style details. All of the participants gave positive feedback on our system, and they had a better user experience for our system than FiberMesh. They commented that our system was smarter by quickly generating an animal-like shape that better resembled the drawn sketches. On average it took only 5 minutes (on average, 4 minutes for sketching on two views and 1 minute for sketching on the surface) for them to obtain a satisfying result using our system, while they often ran out of the time when using FiberMesh. We found the novices preferred to draw 2D sketches than to edit 3D surfaces. Half of them even did not utilize the tools to revise the models. This discovery also confirms that directly 3D modeling is not easy to grasp for amateurs.

Comparison with ZBrush. A skilled artist was recruited, who had eight years of drawing experience and around five years of modeling experience using ZBrush. He was given two reference models and asked to use ZBrush to model similar shapes. The modeling procedure was recorded. Specifically, the models (l) and (m) in Fig. 9 were used as

the reference shapes, both of which were obtained within 12 minutes (around 4 minutes for sketching on the views and 8 minutes for sketching on the surface) with our system by an amateur user with little training in modeling. We provided the front and side rendering images of the models to the artist. It took above 8 minutes for him to achieve a satisfying result on average. The left column of Fig. 15 shows the representative results during the process. Then, we invited him to reproduce the semblable shapes from the output of *Pixel2Mesh* by adding geometry details, which costed him around 3 minutes. Note that ZBrush is designed for professionals with 3D modeling skills or graphics knowledge, while our system can be easily used by novices. Our system makes amateurs create an impressive model using almost the same time compared with artists.

Method	*Retrieval*	*3dR2N2*	*OGN*	*MVRN*	*DeepSDF*	*AtlasNet*	*Pixel2Mesh*	*Ours*
CD($\times 10^{-3}$)	3.114	2.857	7.262	3.102	2.403	1.533	0.9602	**0.8173**
Users' Eva	27.47%	3.33%	0.40%	0.27%	6.00%	0.67%	1.73%	**60.13%**

TABLE 1: Quantitative comparisons on sketch-based mesh inference. The first row reports the CD metric evaluation (the lower the better) while the second row shows the voting percentage from 50 users for the best results.

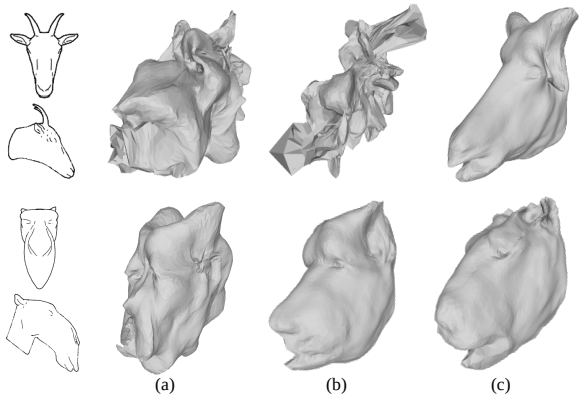


Fig. 16: Qualitative comparisons between our approach and a parametric-based method, which first uses PCA to build a parametric space and then uses a trained deep regression model to map input sketches to a set of parameters. (a), (b), and (c) are the results by using the parametric spaces with dimensions = 100, 200, and 300, respectively. Please refer to the corresponding results by our method in Fig. 14. A consistent parametric space is hard to establish and regress since the rich data diversity.

6.2.2 Comparisons on Mesh Inference

In this section, we evaluate and compare our method with the existing approaches that can generate a mesh from an input sketch.

Comparisons with Deepnet-based Methods. Many existing deep network designs could be utilized to build the mapping from 2D sketches to the corresponding 3D shapes of animal heads. We chose the methods of [34] (denoted as $3dR2N2$), [37] (denoted as OGN), [4] (denoted as $MVRN$), [58] (denoted as $DeepSDF$), [44] (denoted as $AtlasNet$) and [9] ($Pixel2Mesh$) for comparisons, because of their publicly available implementations and representativeness. We further adopted a sketch-based shape retrieval method [28] to search for the closest shapes from the training data as a reference. For a fair comparison, the outputs of these methods were all converted to a mesh representation. The Marching Cube technique [59] was used with a threshold of 0.4 provided in [34] for mesh conversion from the output occupancy probabilities of $3dR2N2$ and OGN, whose resolutions were both in 64^3. The zero-isosurfaces of $DeepSDF$ generation in 64^3 were also extracted with the Marching Cube technique. All the networks were trained using dual-view inputs (i.e., the front view and left-side view sketches), by taking the same training and testing data. We evaluate the averaged CD metric for these methods and report the results in Table 1. The visual comparisons are shown in Fig. 14. Although the retrieval-based method [28] searches similar shapes for the input sketches, their morphologies are always different from the ground truth. It also fails when the underlying models of the input sketches are far from any model in the database. More visual results are illustrated in Fig. 17. OGN and $MVRN$ are pretty difficult to train, leading to the unsatisfactory generation. $3dR2N2$, $AtlasNet$, and $Pixel2Mesh$ can produce generally satisfactory results but fail to capture geometry details of animal heads. Even the recent implicit representation learning method, such as $DeepSDF$, still produces kind of smooth shapes and suffers from topological artifacts when the structure is thin (e.g., in the horn regions), due to the fact that extracting global features from the input sketches would lose detail information for reconstruction. Note that the mesh results of $3dR2N2$, OGN, and $DeepSDF$ contain more than 10,000 vertices in our experiments, while ours are with only 4,962 vertices. Our method led to much better results, both qualitatively and quantitatively. The CD metric is usually not sufficient to evaluate the output quality, especially for synthesized details. We thus conducted an extra user study (please refer to supplemental material for details). We randomly selected 15 examples and invited 50

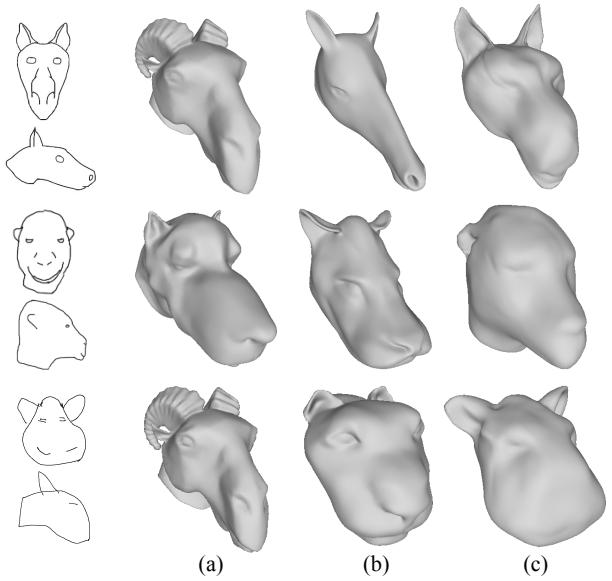


Fig. 17: Comparison between our method (c) with sketch-based shape retrieval [28] for novel sketch inputs. Given dual-view sketches, top-2 retrieved results are shown in (a) and (b).

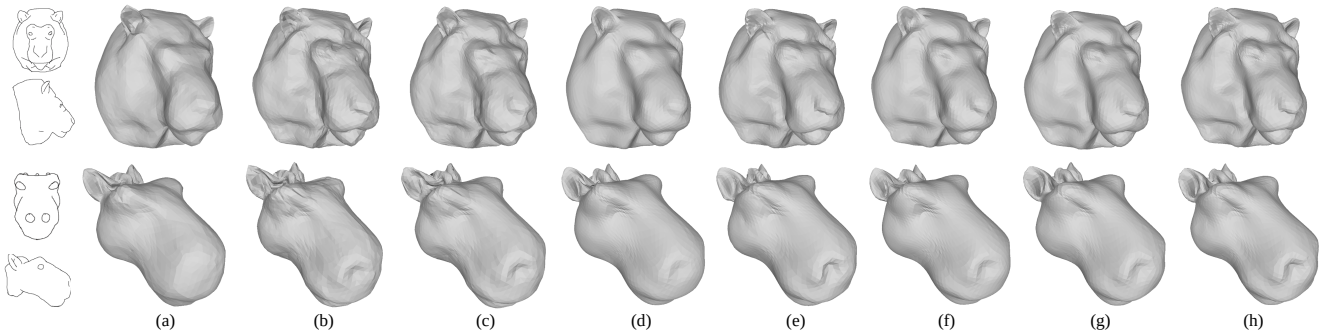


Fig. 18: Qualitative ablation analysis on our network design. (a)-(h) correspond to Table 2.

subjects to evaluate the visual quality of the results by the compared methods. Specifically, for each example, given input sketches and the corresponding results synthesized with different methods, each participant was asked to pick a result that best resembles the input sketches. The voting percentages are reported in Table 1, from which we can see that the results by our method were highly preferred by the participants.

Comparison with Parametric-based Method. Another straightforward solution is to firstly build a parametric space for animal-like head models and then train a deep-regression model to map sketches into a set of parameters, as similarly done in [8]. We thus firstly built the dense correspondences across all the models, using the approach mentioned in the part of "model interpolation" in Section 4.2. After that, the method of [60] was exploited to construct the parametric space. Three spaces with different dimensions, i.e., 100-d, 200-d, and 300-d, were finally built. We separately trained the deep regression models for them, which are denoted as $DeepReg_{100}$, $DeepReg_{200}$, and $DeepReg_{300}$. The evaluated CD metric for these inference models are 63.2, 62.4, and 61.0 respectively (all with 10^{-3}) (Please refer to Table 1 for the CD value of our method), while the visual comparisons of two sampled examples are shown in Fig. 16. As seen, the results of parametric-based methods are far from satisfactory. The reasons are two-fold: 1) due to the complexity and diversity of the studied shape space, it is difficult to build a low-dimensional parametric space; 2) regressing many parameters is inherently a very challenging task.

Comparison with Sketch-based Retrieval. We have compared our approach with the method of [28] in the previous subsection. Fig. 17 shows the visual comparisons for the inputs of more novel dual-view sketches, where the top-2 retrieved models are shown. As seen, the results by our approach better resembles the input sketches due to its better generalization ability.

6.3 Ablation Study on Network Architecture

Our network design has many options. In this section, we conducted an ablation study to evaluate their effects. Specifically, 7 alternative architectures were evaluated: (a) w/o view-surface alternating mesh enhancing, which was our baseline, i.e., only with *Pixel2Mesh*; (b) w/o surface-based detail refinement, i.e., only applying *Pixel2Mesh+VDS-Net*; (c) w/o end-to-end training, i.e., simply concatenating the pre-trained sub-networks together; (d) w/o sketch guidance in *VDS-Net*; (e) w/o sketch guidance in *SDR-Net*; (f) w/o the cascaded mechanism for displacement inference; (g) w/o the cascaded mechanism for *VDS-Net+SDR-Net*. Table 2 reports the CD evaluation for these alternative designs. Fig. 14 shows the qualitative results of two sampled examples by the alternative methods. As seen, our final design produces the best results.

w/o phases	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)
CD($\times 10^{-3}$)	0.9602	0.8331	0.8330	0.9245	0.8362	0.8371	0.8302	**0.8173**

TABLE 2: The evaluated CD metric on our testing dataset for 7 alternative network designs. The smaller the better. (a) w/o the stage of view-surface alternative mesh enhancing (i.e., using *Pixel2Mesh* only); (b) w/o surface-based detail refinement; (c) w/o end-to-end training; (d) w/o sketch guidance for in *VDS-Net*; (e) w/o sketch guidance in *SDR-Net*; (f) w/o cascaded mechanism for displacement inference; (g) w/o cascaded mechanism for *VDS-Net+SDR-Net*; (h) our whole architecture.

7 Conclusion and Limitations

In this paper, we have presented an easy-to-use sketching system for modeling 3D animal-like character heads. Our user interface allows users to sketch on either single-view (front-view or side-view) or dual-view canvas for initial modeling and sketch on the surface for more detailed manipulation in real-time. Built upon $Pixel2Mesh$, an end-to-end mesh inference network is trained to map input sketches to a clean mesh. Our method based on a novel view-surface alternating mesh enhancing network synthesizes geometry details very well. Both qualitative and quantitative comparisons show the superiority of our method over state-of-the-art techniques and alternative solutions. The user evaluations also verify the usability of our interface.

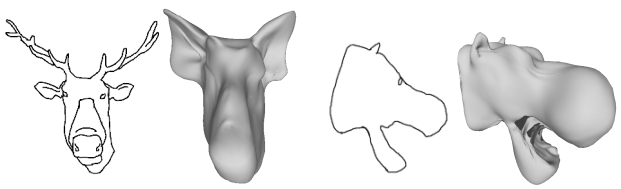


Fig. 19: The limitation of our system to model complex shapes. As shown, the horns of the deer and the open mouth of the hippo cannot be reconstructed correctly.

Limitations and Further Works. Our system still suffers from several limitations, which are left as future works. First, the output meshes still lack fancy geometric details, compared with the ground truth. This might be improved by designing a more efficient feature extraction layer on a mesh, rather than graph convolutions. More effective surface metrics for details, e.g., [61], are also worth studying in the future. Second, our current system lacks details controllability in the mode of "Sketching on View". For example, it is not allowed to do asymmetric details enhancement. If the users are unsatisfied with the automatically inferred geometric details, they are only allowed to do the tuning in the mode of "Sketching on Surface". Third, this work has focused on head modeling only. It might be interesting to make an extension to full-body animals, where the pose modeling needs to be considered. Fourth, as illustrated in Fig. 19, using our system 1) is very hard to create shapes with thin structures (the left example) due to the modeling capability of deformation-based methods; 2) tends to produce artifacts when the shape is far from the models in our dataset (the right example) due to the limited generalization ability of our mesh inference algorithm.

ACKNOWLEDGEMENT

The authors would like to thank the reviewers for their constructive comments, and the participants of our user study for their precious time. The work was supported in part by the National Key R&D Program of China with grant No. 2018YFB1800800, by the Key Area R&D Program of Guangdong Province with grant No. 2018B030338001, by Shenzhen Outstanding Talents Training Fund, by Guangdong Research Project (No. 2017ZT07X152), by Zhejiang Lab (No. 2019NB0AB03), by the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CityU 11212119), by the City University of Hong Kong (Project No. 7005176), by Hong Kong Research Grants Council under General Research Funds (HKU17206218), and by the National Natural Science Foundation of China (61902334, 61629101, 61672482).

REFERENCES

[1] T. Igarashi, S. Matsuoka, and H. Tanaka, "Teddy: A sketching interface for 3d freeform design," in *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '99. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1999, pp. 409–416. [Online]. Available: http://dx.doi.org/10.1145/311535.311602

[2] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa, "Fibermesh: designing freeform surfaces with 3d curves," *ACM transactions on graphics (TOG)*, vol. 26, no. 3, p. 41, 2007.

[3] J. Delanoy, M. Aubry, P. Isola, A. Efros, and A. Bousseau, "3d sketching using multi-view deep volumetric prediction," *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, vol. 1, no. 21, 2018.

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

[5] W. Su, D. Du, X. Yang, S. Zhou, and H. Fu, "Interactive sketch-based normal map generation with deep neural networks," *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, vol. 1, no. 1, pp. 1–17, 2018.

[6] C. Li, H. Pan, Y. Liu, X. Tong, A. Sheffer, and W. Wang, "Robust flow-guided neural prediction for sketch-based freeform surface modeling," in *SIGGRAPH Asia 2018 Technical Papers*. ACM, 2018, p. 238.

[7] G. Nishida, I. Garcia-Dorado, D. G. Aliaga, B. Benes, and A. Bousseau, "Interactive sketching of urban procedural models," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, p. 130, 2016.

[8] X. Han, C. Gao, and Y. Yu, "Deepsketch2face: a deep learning based sketching system for 3d face and caricature modeling," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 126, 2017.

[9] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang, "Pixel2mesh: Generating 3d mesh models from single rgb images," in *ECCV*, 2018.

[10] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 295–307, 2015.

[11] J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1646–1654.

[12] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, "Enhanced deep residual networks for single image super-resolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017, pp. 136–144.

[13] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4681–4690.

[14] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *CVPR*, 2017.

[15] L. Olsen, F. F. Samavati, M. C. Sousa, and J. A. Jorge, "Sketch-based modeling: A survey," *Computers & Graphics*, vol. 33, no. 1, pp. 85–103, 2009.

[16] C. Ding and L. Liu, "A survey of sketch based modeling systems," *Frontiers of Computer Science*, vol. 10, no. 6, pp. 985–999, 2016.

[17] K. Singh and E. Fiume, "Wires: a geometric deformation technique," in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. ACM, 1998, pp. 405–414.

[18] A. Nealen, O. Sorkine, M. Alexa, and D. Cohen-Or, "A sketch-based interface for detail-preserving mesh editing," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 1142–1147, Jul. 2005. [Online]. Available: http://doi.acm.org/10.1145/1073204.1073324

[19] R. Schmidt, B. Wyvill, M. C. Sousa, and J. A. Jorge, "Shapeshop: Sketch-based solid modeling with blobtrees," in *ACM SIGGRAPH 2006 Courses*. ACM, 2006, p. 14.

[20] O. A. Karpenko and J. F. Hughes, "Smoothsketch: 3d free-form shapes from complex sketches," in *ACM Transactions on Graphics (TOG)*, vol. 25, no. 3. ACM, 2006, pp. 589–598.

[21] Y. Gingold, T. Igarashi, and D. Zorin, "Structured annotations for 2d-to-3d modeling," in *ACM Transactions on Graphics (TOG)*, vol. 28, no. 5. ACM, 2009, p. 148.

[22] F. Cordier, H. Seo, J. Park, and J. Y. Noh, "Sketching of mirror-symmetric shapes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 11, pp. 1650–1662, 2011.

[23] C. Shao, A. Bousseau, A. Sheffer, and K. Singh, "Crossshade: Shading concept sketches using cross-section curves," *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2012)*, vol. 31, no. 4, 2012. [Online]. Available: http://www.crossshade.com

[24] B. Xu, W. Chang, A. Sheffer, A. Bousseau, J. McCrae, and K. Singh, "True2form: 3d curve networks from 2d sketches via selective regularization," *Transactions on Graphics (Proc. SIGGRAPH 2014)*, vol. 33, no. 4, 2014.

[25] H. Pan, Y. Liu, A. Sheffer, N. Vining, C.-J. Li, and W. Wang, "Flow aligned surfacing of curve networks," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, p. 127, 2015.

[26] C. Li, H. Pan, Y. Liu, A. Sheffer, and W. Wang, "Bendsketch: Modeling freeform surfaces through 2d sketching," *ACM Trans. Graph. (SIGGRAPH)*, vol. 36, no. 4, pp. 125:1–125:14, 2017.

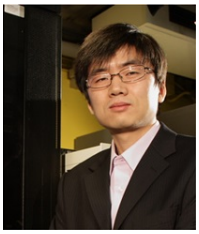
[27] M. Eitz, R. Richter, T. Boubekeur, K. Hildebrand, and M. Alexa, "Sketch-based shape retrieval." *ACM Trans. Graph.*, vol. 31, no. 4, pp. 31–1, 2012.

[28] F. Wang, L. Kang, and Y. Li, "Sketch-based 3d shape retrieval using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1875–1883.

[29] L. Fan, R. Wang, L. Xu, J. Deng, and L. Liu, "Modeling by drawing with shadow guidance," *Computer Graphics Forum (Proc. of Pacific Graphics 2013)*, vol. 23, no. 7, pp. 157–166, 2013.
[30] X. Xie, K. Xu, N. J. Mitra, D. Cohen-Or, W. Gong, Q. Su, and B. Chen, "Sketch-to-design: Context-based part assembly," in *Computer Graphics Forum*, vol. 32, no. 8. Wiley Online Library, 2013, pp. 233–245.
[31] K. Xu, K. Chen, H. Fu, W.-L. Sun, and S.-M. Hu, "Sketch2scene: sketch-based co-retrieval and co-placement of 3d models," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, pp. 1–15, 2013.
[32] X. Guo, J. Lin, K. Xu, S. Chaudhuri, and X. Jin, "Customcut: On-demand extraction of customized 3d parts with 2d sketches," in *Computer Graphics Forum*, vol. 35, no. 5. Wiley Online Library, 2016, pp. 89–100.
[33] L. Li, H. Fu, and C.-L. Tai, "Fast sketch segmentation and labeling with deep learning," *IEEE computer graphics and applications*, vol. 39, no. 2, pp. 38–51, 2018.
[34] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, "3d-r2n2: A unified approach for single and multi-view 3d object reconstruction," in *European conference on computer vision*. Springer, 2016, pp. 628–644.
[35] R. Girdhar, D. F. Fouhey, M. Rodriguez, and A. Gupta, "Learning a predictable and generative vector representation for objects," in *European Conference on Computer Vision*. Springer, 2016, pp. 484–499.
[36] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling," in *Advances in Neural Information Processing Systems*, 2016, pp. 82–90.
[37] M. Tatarchenko, A. Dosovitskiy, and T. Brox, "Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs," in *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, vol. 2, 2017, p. 8.
[38] P.-S. Wang, C.-Y. Sun, Y. Liu, and X. Tong, "Adaptive o-cnn: a patch-based deep representation of 3d shapes," in *SIGGRAPH Asia 2018 Technical Papers*. ACM, 2018, p. 217.
[39] H. Fan, H. Su, and L. J. Guibas, "A point set generation network for 3d object reconstruction from a single image," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 605–613.
[40] C.-H. Lin, C. Kong, and S. Lucey, "Learning efficient point cloud generation for dense 3d object reconstruction," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
[41] A. Arsalan Soltani, H. Huang, J. Wu, T. D. Kulkarni, and J. B. Tenenbaum, "Synthesizing 3d shapes via modeling multi-view depth maps and silhouettes with deep generative networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1511–1519.
[42] X. Zhang, Z. Zhang, C. Zhang, J. Tenenbaum, B. Freeman, and J. Wu, "Learning to reconstruct shapes from unseen classes," in *Advances in Neural Information Processing Systems*, 2018, pp. 2263–2274.
[43] H. Kato, Y. Ushiku, and T. Harada, "Neural 3d mesh renderer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3907–3916.
[44] T. Groueix, M. Fisher, V. G. Kim, B. Russell, and M. Aubry, "AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation," in *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.
[45] H. Huang, E. Kalogerakis, E. Yumer, and R. Mech, "Shape synthesis from sketches via procedural models and convolutional networks," *IEEE transactions on visualization and computer graphics*, vol. 23, no. 8, 2017.
[46] L. Wang, C. Qian, J. Wang, and Y. Fang, "Unsupervised learning of 3d model reconstruction from hand-drawn sketches," in *2018 ACM Multimedia Conference on Multimedia Conference*. ACM, 2018, pp. 1820–1828.
[47] D. DeCarlo, A. Finkelstein, S. Rusinkiewicz, and A. Santella, "Suggestive contours for conveying shape," *ACM Transactions on Graphics (TOG)*, vol. 22, no. 3, pp. 848–855, 2003.
[48] Q. Chen and V. Koltun, "Photographic image synthesis with cascaded refinement networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1511–1520.
[49] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Transactions on Graphics (ToG)*, vol. 32, no. 3, p. 29, 2013.
[50] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa, "Laplacian mesh optimization," in *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*. ACM, 2006, pp. 381–389.
[51] K. G. Kobayashi and K. Ootsubo, "t-ffd: free-form deformation by using triangular mesh," in *Proceedings of the eighth ACM symposium on Solid modeling and applications*. ACM, 2003, pp. 226–234.
[52] Y. Yang, X.-M. Fu, S. Chai, S.-W. Xiao, and L. Liu, "Volume-enhanced compatible remeshing of 3d models," *IEEE transactions on visualization and computer graphics*, vol. 25, no. 10, pp. 2999–3010, 2018.
[53] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.
[54] P. Sangkloy, N. Burnell, C. Ham, and J. Hays, "The sketchy database: learning to retrieve badly drawn bunnies," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, pp. 1–12, 2016.
[55] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel, "Laplacian surface editing," in *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. ACM, 2004, pp. 175–184.
[56] M. Peng, J. Xing, and L.-Y. Wei, "Autocomplete 3d sculpting," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, p. 132, 2018.
[57] C. Loop, "Smooth subdivision surfaces based on triangles," *Master's thesis, University of Utah, Department of Mathematics*, 1987.
[58] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 165–174.
[59] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," in *ACM siggraph computer graphics*, vol. 21, no. 4. ACM, 1987, pp. 163–169.
[60] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3d faces," in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, 1999, pp. 187–194.
[61] D. Smirnov, M. Fisher, V. G. Kim, R. Zhang, and J. Solomon, "Deep parametric shape predictions using distance fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 561–570.

Dong Du is a fifth-year PhD student of University of Science and Technology of China, and a visiting student of Shenzhen Research Institute of Big Data. He received his BSc in applied mathematics in 2014 from Nanjing University of Science and Technology, and visited the School of Creative Media, City University of Hong Kong as a research associate in 2017. His research mainly focuses on computer graphics and 3D deep learning.

Xiaoguang Han received his BSc in mathematics in 2009 from NUAA and his MSc in applied mathematics in 2011 from Zhejiang University. He obtained his PhD degree in 2017 from HKU. He is currently a Research Assistant Professor at Shenzhen Research Institute of Big Data, the Chinese University of Hong Kong(Shenzhen). His research mainly focuses on computer vision, computer graphics and 3D deep learning.



Hongbo Fu is a full professor in the School of Creative Media, City University of Hong Kong. Before joining CityU, he had postdoctoral research trainings at the Imager Lab, University of British Columbia, Canada and the Department of Computer Graphics, Max-Planck-Institut Informatik, Germany. He received the PhD degree in computer science from the Hong Kong University of Science and Technology in 2007 and the BS degree in information sciences from Peking University, China, in 2002. His primary research interests fall in the fields of computer graphics and human computer interaction. He has served as an associate editor of The Visual Computer, Computers&Graphics, and Computer Graphics Forum.

Feiyang Wu is currently an senior undergraduate student at the CUHK Shenzhen. He majors in Computer Science and Engineering. After finishing his internship at Didi Chuxing Co., Wu continued his academic pursuit. He is looking for a PhD position in computer vision area.

Yizhou Yu is a full professor in the Department of Computer Science at the University of Hong Kong. He was first a tenure-track and then a tenured professor at University of Illinois, Urbana-Champaign (UIUC) for 12 years. He has also collaborated with eBay Research, Google Brain and Microsoft Research in the past. He received his PhD degree in computer science from the computer vision group at University of California, Berkeley. He also holds a MS degree in applied mathematics and a BE degree in computer science and engineering from Zhejiang University. His current research interests include deep learning methods for machine intelligence, computational visual media, geometric computing, intelligent video surveillance, and biomedical data analysis.

Shuguang Cui received his Ph.D in Electrical Engineering from Stanford University, California, USA, in 2005. Afterwards, he has been working as assistant, associate, full, Chair Professor in Electrical and Computer Engineering at the Univ. of Arizona, Texas A&M University, and UC Davis, respectively. He is currently a Chair Professor at CUHK Shenzhen and the Vice Director at Shenzhen Research Institute of Big Data. His current research interests focus on data driven large-scale system control and resource management, large data set analysis, IoT system design, energy harvesting based communication system design, and cognitive network optimization.

Ligang Liu is a professor at the School of Mathematical Sciences, University of Science and Technology of China. He received his B.Sc. (1996) and his Ph.D. (2001) from Zhejiang University, China. Between 2001 and 2004, he worked at Microsoft Research Asia. Then he worked at Zhejiang University during 2004 and 2012. He paid an academic visit to Harvard University during 2009 and 2011. His research interests include digital geometric processing, computer graphics, and image processing. He serves as the associated editors for journals of IEEE Transactions on Visualization and Computer Graphics, IEEE Computer Graphics and Applications, Computer Graphics Forum, Computer Aided Geometric Design, Computers&Graphics, and The Visual Computer. He served as the conference co-chair of GMP 2017 and the program co-chairs of SIAM GD 2019, GMP 2018, CAD/Graphics 2017, CVM 2016, SGP 2015, and SPM 2014. His research works could be found at his research website: http://staff.ustc.edu.cn/~lgliu