DOI: 10.1111/cgf.70297

COMPUTER GRAPHICS *forum*
Volume 0 (2025), number 0, e70297

Robust Differentiable Sketch Rendering for Single-View 3D Reconstruction

Aobo Jin,[1] Qixin Deng,[2] Lei Si[3] and Zhigang Deng[3]

[1]Department of Computer Science, University of Houston-Victoria, Victoria, USA
jina@uhv.edu
[2]Department of Computer Science, Wabash College, Crawfordsville, USA
dengq@wabash.edu
[3]Department of Computer Science, University of Houston, Houston, USA
lsi@uh.edu, zdeng4@central.uh.edu

Abstract

In this paper, we propose a novel end-to-end method to model 3D objects with geometric details from a single-view sketch input. Specifically, a novel deep learning-based differentiable sketch renderer is introduced to establish the relationship between geometric features, represented by normal maps, and 2D sketch strokes. Then, building upon this renderer, we design algorithms to automatically create 3D models with geometric details from a single-view sketch. With the aid of two introduced loss functions: one based on silhouette-derived confidence maps and the other on regression similarities, our framework supports the gradient of loss functions calculated between the rendered sketch and input sketch back-propagating through the whole architecture, thereby enhancing the geometric details on the frontal surface of the generated 3D object. Through comparisons with state-of-the-art sketch-based 3D modelling techniques, our approach demonstrates superior capability in generating plausible geometric shapes and details, without the necessity for semantic annotations within the input sketch.

Keywords: modelling, geometric modelling

CCS Concepts: • Computing methodologies → Computing methodologies; Neural networks;

1. Introduction

Sketch-based modelling has attracted significant attention in recent years due to its potential to simplify the process of 3D modelling through sketching—an inherently intuitive method for conveying shapes. Typically, a 2D sketch encapsulates a variety of geometric features such as silhouettes, occluding contours, suggestive contours, ridges, and valleys. Historically, most sketch-based modelling techniques [ZHH07, GIZ09, XXM*13, FCSF16, HKYM17] have necessitated either multiple-view sketches [RDI10, LGK*17] or semantically annotated strokes [LPL*17, LPL*18] as prerequisites to mitigate issues related to depth ambiguity. Consequently, the requirements for sketch inputs in these methods pose considerable challenges, particularly for novice users.

Single-view sketch-based 3D modelling presents significant technical challenges primarily due to two factors. First, the application of state-of-the-art methods for 3D object modelling from RGB images [WZL*18, WRV20, JRR*19, LLCL19] is not directly transferable to sketch-based modelling. This is because a single-view sketch comprises only sparse 2D strokes and lacks the shading and texture present in RGB images, which contain substantial depth information. Therefore, the differentiable renderers employed in those methods, which are designed to process RGB outputs, are not suitable for our task. Consequently, accurately predicting surface details and modelling fine features, such as ridges, valleys, and structural elements like animal legs, becomes problematic.

Second, the development of supervised machine learning models to establish a mapping between single-view 2D sketches and category-agnostic 3D models (e.g., not limited to a specific category of objects like faces or chairs) often encounters performance limitations due to the inherent ambiguity in 3D shape representation from sparse sketches. This ambiguity complicates the models' ability to accurately reconstruct geometric details, such as wrinkles and other fine features.

In this work, we propose a deep learning based differentiable sketch renderer that can be seamlessly integrated into various deep learning architectures to facilitate 3D modelling with plausible geometric details from a single-view sketch input. Specifically, we first introduce a *differentiable sketch renderer*—a trainable architecture designed to convert normal maps into sketches utilising in-house-collected pairs of sketches and normal maps, by learning the geometric feature relationships between normal maps and 2D semantic strokes. The proposed renderer architecture is the first differentiable sketch renderer, demonstrating that a trainable design can enhance detail modelling and be integrated into various deep learning architectures. Then, we further train our end-to-end deep learning framework using an in-house-constructed dataset that encompasses various 3D objects and their corresponding planar sketches. Utilising a single-view sketch input, our method involves predicting a preliminary 3D model by deforming a predefined sphere coupled with a displacement map, and the front view of the 3D object is rendered to a normal map. After that, leveraging the capabilities of the differentiable sketch renderer and the introduced silhouette-based confidence maps, gradients of loss functions between the rendered sketch and the input sketch are back-propagated throughout the entire architecture. This back-propagation facilitates the update of learnable weights, thereby enhancing the geometric details of the predicted 3D object. It is noteworthy that since the 3D models are generated by deforming a predefined 3D sphere, our methodology is inherently limited to modelling watertight surfaces such as genus zero.

To evaluate the effectiveness of our method, we conducted comparative experiments with various state-of-the-art sketch-based modelling methods. Our experimental results demonstrate that our method can efficiently generate more plausible 3D objects with geometric details at runtime, without requiring semantic annotations in the input sketch.

In sum, the main contributions of our method are as follows:

- We introduce a differentiable sketch renderer capable of converting a normal map into a sketch, thus learning the geometric feature relationships between normal maps and 2D semantic strokes (e.g., suggestive contours, ridges, and valleys). The differentiable nature of the sketch renderer facilitates its seamless integration into deep learning frameworks for comprehensive end-to-end training.
- We introduce an end-to-end architecture for 3D object modelling that effectively captures plausible geometric details from a single-view sketch, significantly reducing the barrier to practical application by eliminating the need for semantic annotations on the input strokes.
- We introduce two innovative loss functions: a silhouette-based confidence maps weighted Intersection Over Union (IOU) loss and a similarly weighted L2 regression loss, which enhance the reconstruction of geometric details and improve the accuracy by clearly distinguishing the contributions of each supervised silhouette.

2. Related Work

Single-view Sketch-based Modelling. Early geometric inference methods [IMT99] employed pre-defined rules to deduce local geometric properties from 2D strokes. Although these methods were limited in their ability to propagate geometric details, thus reducing their ability to generate complex 3D objects, they did not necessitate semantic strokes in the input. Later, various approaches [KH06, NISA07, FWX*13] were developed to create free-form surfaces by leveraging geometric constraints or shadow guidance found within specific types of line drawings. Additionally, they developed a method to search for and combine nearest shapes from a pre-ordered dataset to construct new 3D shapes. Building upon this concept, researchers explored the strategy of retrieving 3D object parts from a dataset and assembling them to create a new shape that aligns with a user-drawn sketch [XXM*13, GLX*16]. To address surface detail inference from sketches, the use of 2D semantically-annotated strokes was employed as a prior to limit depth ambiguity [LPL*17, LPL*18]. Jin et al. [JFD20] simplified the user interface by requiring only 2D occluding contours as input, integrating these contours and 3D objects into a joint space. However, their method does not adequately address surface details.

The recently proposed sketch-based modelling methods can generate 3D shapes based on the sketch input. Methods like Sketch2Model [ZGG21], SENS [BHSH*24], LAS-D [ZPW*23], and Doodle-Your-3D [BKD*24] focus on utilizing sketches to generate 3D shapes that do not accurately align with the input relative sketches. The 3D shapes capture the semantic information from the input sketches but omit the overall accuracy. In contrast, our work aims to generate 3D shapes which can align well with the input 2D sketch strokes. The shape and geometric details of the generated 3D shapes match as much as possible with the input relative sketches. Methods like SketchSampler [GYS*22] and Sketch2Mesh [GRYF21] can generate 3D shapes aligned with the 2D input sketches but they cannot generate 3D meshes directly. They either generate implicit representations or cloud representations. Although the results can be transferred to a mesh, the output quality is often either unsmooth or loses important details. In contrast, our method is an end-to-end approach that directly generates high-quality 3D meshes from input sketches.

Differentiable Rendering for 3D Reconstruction. Recently, advancements have been made to incorporate differentiable capabilities within rendering pipelines. Methods such as OpenDR [LB14] and NMR [KUH18] allow back-propagation during training by computing approximate gradients, while preserving traditional rendering processes during the forward pass. Innovations like SoftRas [LLCL19] and DIB-R [CGL*19] have emerged as probabilistic process methods that depart from conventional rendering pipelines, where the coloration of each pixel is decided by multiple mesh faces. Differentiable rendering has been extensively applied in various research areas, including human shape and pose estimation [BKL*16, KBJM18, OLP*18, KAB20], hand shape and pose estimation [WPVY19, KGK*20], and 3D face reconstruction [SSB*19, WRV20], which benefit from the absence of 3D ground truth supervision. PyTorch3D [JRR*19], based on SoftRas [LLCL19], offers a modular approach to allow users to customise the rendering pipeline using PyTorch shaders. Xiang et al. [XWJ*20] developed a two-stage approach that initially transforms a sketch image into a normal map and further recovers its corresponding 3D shape through a differentiable rendering-based pipeline. However, their method falls short in recovering detailed 3D geometry. Furthermore,

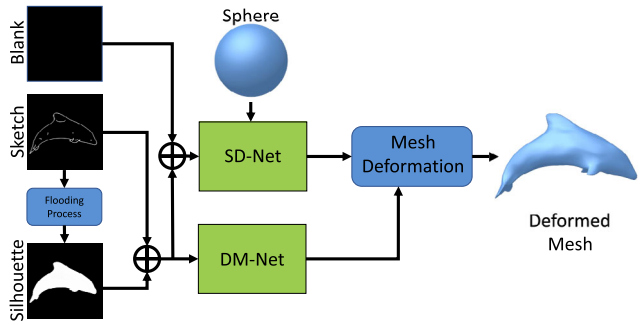

Figure 1: *The inference pipeline of our approach. The silhouette image is computed by a flooding process that starts from background pixels and stops at arbitrary sketch pTixels. The vertex displacements outputted from the SD-Net are used to directly deform the sphere to generate a target shape.*

differentiable ray tracing methods such as Redner [LADL18] and Mitsuba2 [NDVZJ19] have the capability to produce more photorealistic images by significantly increasing computational resources, making them less suitable for integration into machine learning pipelines due to their intensive computational demands. 3Doodle [CLPK24] generates view-consistent sketches from multi-view images by optimising 3D stroke primitives (e.g., Bézier curves, superquadric contours) through a differentiable sketch renderer. However, its differentiable rendering operates on abstract 3D strokes, making it unsuitable for integration into our surface-based 3D reconstruction framework.

3. Our Approach

The inference pipeline of our approach, depicted in Figure 1, works as follows: Initially, from a given input sketch image, we derive its silhouette using a pixel flooding process, which begins at the background pixels and concludes at any encountered sketch pixels. Then, the extracted sketch image, its silhouette, and a blank image are combined into a 3-channel image, which is then inputted into a pre-trained Shape Deformation Network (SD-Net) along with a pre-defined reference sphere mesh. The SD-Net predicts the displacements for all vertices, briefly described below and elaborated in Section §6.1. These predicted offsets are utilised as vertex displacements to morph the reference sphere into a target shape that aligns with the silhouette derived from the input sketch.

To further refine the geometric details of the deformed sphere, we introduce a Displacement Mapping Network (DM-Net), which receives input of a 2-channel image (i.e., sketch image and its silhouette) to create a displacement map. The training process of the DM-Net is described in section §6.2. Ultimately, the displacement map is applied to the front faces of the deformed sphere, ensuring alignment with the internal feature strokes of the sketch, thereby enhancing the visual coherence and detail of the model.

The training pipeline is illustrated in Figure 2. The silhouettes of the deformed mesh are rendered from six different views using a renderer based on the predicted mesh, the renderer is differentiable to allow gradient back-propagation through the entire pipeline, and silhouette losses are computed with corresponding confidence maps. These confidence maps for the six views are generated by an encoder–decoder architecture, with the sketch serving as the input. Furthermore, a differentiable sketch renderer (SR-Net)

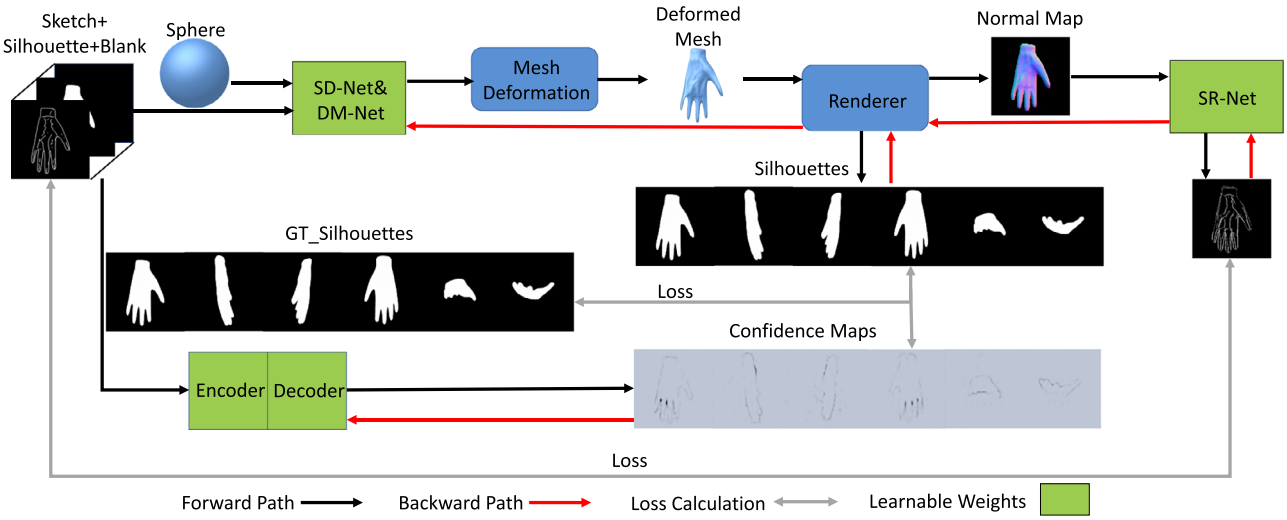

Figure 2: *Pipeline of the our training process. Black, red, and grey arrows indicate forward pass, back-propagation, and loss computation, respectively. The input includes a binary sketch, silhouette mask, and a blank image (added to meet ResNet-50's 3-channel input). A mesh is deformed from a sphere via the Shape Deformation Network (SD-Net) and Displacement Mapping Network (DM-Net). Silhouettes from six views are rendered and compared against ground truth using losses weighted by confidence maps, which are predicted from the sketch using an encoder-decoder. The Sketch Rendering Network (SR-Net) renders a sketch from the predicted normal map, and the sketch loss is computed against the input.*

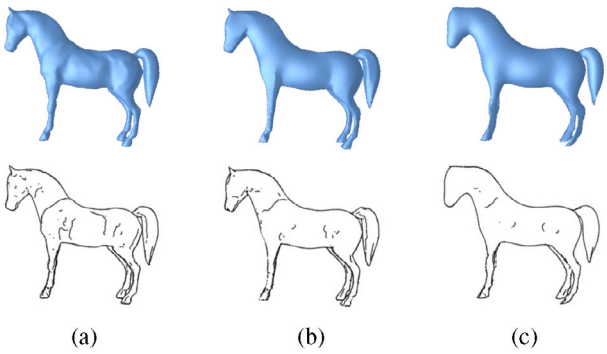

Figure 3: *An example of a 3D object from [CGF09] is depicted along with its planar sketch (a) and its results after undergoing two levels of smoothing ((b) and (c)). The variances between consecutive levels of both the 3D models and the 2D sketches are critical for the training of the Sketch Renderer Network (SR-Net) and the Shape Deformation Network (SD-Net). These differences facilitate the learning process by helping the networks understand the relationship between the 2D inner strokes and the 3D geometric details.*

is employed to convert the normal map—produced from the predicted mesh—into a sketch image. The sketch loss is then calculated based on the discrepancy between this rendered sketch image and the original input sketch image. All losses are subsequently back-propagated through the networks via the backward path (red arrows) to update the weights within the networks, optimising the training outcomes. The SD-Net and DM-Net are optimised jointly within an end-to-end training framework.

4. Data Preprocessing

In our approach, due to the absence of existing pairs of 2D sketches and 3D models (or normal maps), it becomes necessary to pre-process some data for model training, briefly described below. More details of data pre-processing are described in the supplemental document. Our pre-processing consists of the following main steps.

3D model selection. We used a dataset of 3D objects [CGF09] that comprises 19 categories, totaling 380 objects. To accommodate our method's requirement for both sketches and silhouettes from six selected views, we generated such data from these 3D objects. Each 3D object was initially smoothed to create two additional levels of detail. For this smoothing process, we experimentally determined two sets of parameters using bilateral normal filters [ZFAT11]. All 3D objects were zero-centred and normalised to fit within a unit bounding sphere. The objects in each category was randomly divided into training (80%), validation (10%), and testing (10%) subsets. Figure 3 shows an example of a 3D object along with its two-level smoothed results.

Data augmentation. Data augmentation through the selection of additional primary viewpoints was employed to enrich the training data. It is assumed that users typically create sketches from viewpoints that encapsulate the majority of geometric features, termed '*primary viewpoints*.' For each 3D object, primary viewpoints were automatically identified by initially sampling 50 viewpoints uniformly over its unit bounding sphere. The angle between the viewing direction of each sampled viewpoint and the 3D object's Principal Component Analysis (PCA) axis, corresponding to the smallest eigenvalue, was calculated. These angles were then sorted in the ascending order, and the top three viewpoints were selected. After that, for each of these top three viewpoints, four upward directions perpendicular to the viewpoint were generated. Each 3D object was also uniformly scaled using three different scaling factors: 0.8, 1.0, and 1.2. Consequently, for each 3D object, a total of 36 primary viewpoints (3 viewpoints $\times$ 4 upward directions $\times$ 3 scaling factors) were determined. These 36 primary viewpoints for each object were considered as '*front-views*' to significantly augment the training dataset.

Sketch data preparation. Our approach used the well-known non-photorealistic rendering technique developed by DeCarlo et al. [DFRS03] to generate sketches from 3D models. To maintain simplicity in the sketch-based modelling interface, all rendered sketch lines are uniform in width, thereby avoiding the introduction of potential semantic features that could complicate the interface.

Normal maps and silhouette rendering. In our approach, we generated and paired one normal map with six silhouettes from different views, along with a rendered sketch. Each image was produced at a resolution of 128 $\times$ 128, and the pixel values were normalised within the [0, 1] range. The RGB values of the normal map were modified to represent a non-negative transformation of the xyz components of vertex normals. Specifically, the transformation was defined as $X' = (X_{norm} + 1)/2$, and Y' and Z' were derived in a similar manner. The chosen views to capture the six silhouettes include the front, back, left, right, top, and bottom perspectives. Silhouette images were rendered as binary masks where 1 indicates the foreground object and 0 indicates the background.

5. Sketch Renderer

We introduce a novel differentiable sketch renderer, termed SR-Net, which is capable of rendering a normal map into its corresponding sketch while also supporting back-propagation within deep learning frameworks. We aimed to design a lightweight network to address the differentiable sketch rendering problem by taking computational efficiency into consideration. The SR-Net is structured as an encoder–decoder based on the U-Net [RFB15] architecture and features four levels of image resolutions. It accepts a 128 $\times$ 128 $\times$ 3 normal map, I_{norm}, as its input, and produces a 128 $\times$ 128 $\times$ 1 binary sketch image, $\hat{I}_s$, which is combined with the binary silhouette image of the input, $\hat{S}$. Each down-sampling stage in the network comprises a double convolution module paired with a max pooling layer. The double convolution module includes two convolutional operations, each with a kernel size of 3 and padding of 1, followed by a batch normalisation layer and ReLU activation. Each up-sampling stage incorporates a bilinear up-sampling operation with a scale factor of 2, complemented by a double convolution module. Additionally, two double convolution modules are placed at the beginning and end of the architecture to adapt the input and output channels effectively. Figure 4 shows the architecture of SR-Net. Table 1 shows the lightweight encoder-decoder architecture designed specifically to generate a confidence map from a sketch image input.

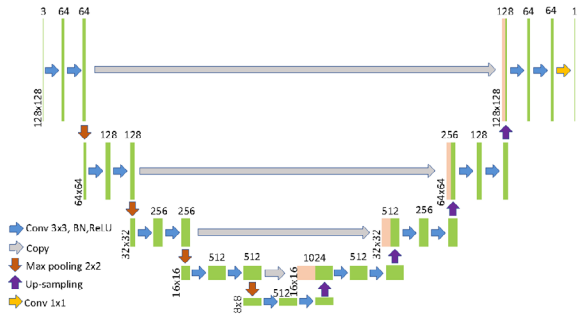

Figure 4: *The architecture of the Sketch Renderer Network (SR-Net) is designed to process a normal map as input and produce a binary sketch image as output. The network architecture integrates a batch normalisation layer and a ReLU activation layer in sequence with each convolutional layer. In the decoder, bilinear up-sampling operations are employed. Additionally, the spatial dimensions and the number of channels for each feature map within the network are annotated alongside in the schematic representation.*

Table 1: *The encoder–decoder architecture utilised for generating confidence maps is detailed as follows: The initial three convolutional layers, as well as all deconvolutional layers, incorporate group normalisation and are succeeded by Leaky ReLU activation, with the exception of the final layer in the decoder.*

Layer	Type	Kernel	Stride	Output
enc.	conv2d(+GN+LReLU)	4×4	2×2	64×64×64
enc.	conv2d(+GN+LReLU)	4×4	2×2	32×32×128
enc.	conv2d(+GN+LReLU)	4×4	2×2	16×16×256
enc.	conv2d(+LReLU)	4×4	2×2	8×8×512
enc.	conv2d(+ReLU)	4×4	1×1	4×4×128
dec.	deconv2d(+ReLU)	4×4	1×1	8×8×512
dec.	deconv2d(+GN+ReLU)	4×4	2×2	16×16×256
dec.	deconv2d(+GN+ReLU)	4×4	2×2	32×32×128
dec.	deconv2d(+GN+ReLU)	4×4	2×2	64×64×64
dec.	deconv2d(+GN+ReLU)	4×4	2×2	128×128×64
dec.	conv2d(+Sigmoid)	5×5	1×1	128×128×6

Note: The fourth convolutional layer is also followed by Leaky ReLU activation. Conversely, the final convolutional layer employs ReLU activation, and the concluding deconvolutional layer adopts Sigmoid activation to finalise the output.

To train the SR-Net, a loss function $\mathcal{L}_{SR}$ is introduced to optimise the weights of the SR-Net as follows:

$$\mathcal{L}_{SR} = \mathcal{L}^{SR}_{L_1\_norm} + \lambda^{SR}_{perc}\mathcal{L}^{SR}_{perc}. \tag{1}$$

$$\mathcal{L}^{SR}_{L_1\_norm} = \frac{1}{n}\sum_{i=1}^{n}|\hat{I}^s_i - I^s_i| + \frac{1}{n}\sum_{i=1}^{n}|\hat{S}^s_i - S^s_i|. \tag{2}$$

$$\mathcal{L}^{SR}_{perc} = \sum_{k=1}^{4}\mathcal{L}^{SR}_{perc(k)}. \tag{3}$$

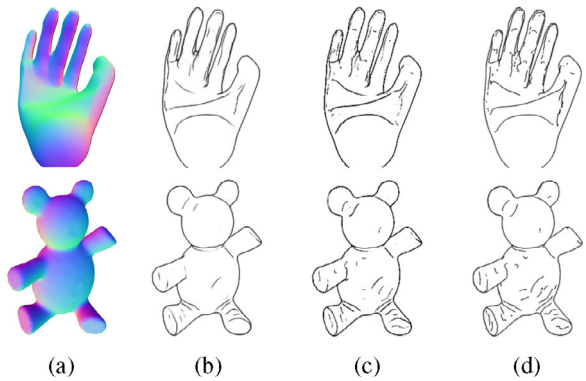

Figure 5: *An example of SR-Net output is used to show the effect of using the L_1 loss alone (b) versus using a combination of the masked perceptual loss and the L_1 loss (c). The input normal map is displayed in (a), and the ground truth sketch image (d) is generated through non-photorealistic rendering as described in [DFRS03].*

$$\mathcal{L}^{SR}_{perc(k)} = \frac{1}{n}\sum_{i=1}^{n}(e^{(k)}_i(\hat{I}^s) - e^{(k)}_i(I^s))^2. \tag{4}$$

In the above Equations (1)–(4), $\mathcal{L}^{SR}_{L_1\_norm}$ is the mean of L_1 distance between the rendered sketch image pixels $\hat{I}^s_i$ and the ground truth sketch image pixels I^s_i plus the mean of L_1 distance between the rendered silhouette image pixels $\hat{S}^s_i$ and the ground truth silhouette image pixels S^s_i, $\mathcal{L}^{SR}_{perc}$ is the masked perceptual loss, and n is the total number of pixels (=128 × 128). The sketch image pixels outside the sketch lines are black (value 0). By accurately predicting silhouettes, the network enhances its capacity to differentiate between 0-valued pixels associated with the object and those parts of the background, thus improving the prediction of internal sketches.

The detailed fidelity in the rendered sketch image would diminish if solely L_1 loss were utilised. To address this, a perceptual loss is integrated to deepen the connection between 2D sketch features and geometric attributes. This integration involves extracting four levels of encoded features from both the rendered sketch image and the ground truth sketch using a pre-trained VGG16 [SZ15]. The perceptual loss $\mathcal{L}^{SR}_{perc(k)}$ at each kth level is calculated by determining the pixel-wise L_1 distance between the encoded features $e^{(k)}_i(\hat{I}^s)$ and $e^{(k)}_i(I^s)$. During training, the weight $\lambda^{SR}_{perc} = 3$ is experimentally set for this loss. As depicted in Figure 5, combining $\mathcal{L}^{SR}_{perc}$ and $\mathcal{L}^{SR}_{L_1\_norm}$ enables the recovery of more geometric details, compared to using the $\mathcal{L}^{SR}_{L_1\_norm}$ loss alone.

6. Mesh Generation Networks

The mesh generation network operates in an end-to-end manner, accepting a 3-channel image—comprising a binary sketch image, a binary mask image and a blank image—and a pre-defined reference sphere as inputs. The output from this network is a detailed mesh, which is essentially a deformation of the sphere influenced by predicted per-vertex offsets and a displacement map. The architecture of the mesh generation network is divided into two components: the SD-Net and the DM-Net.

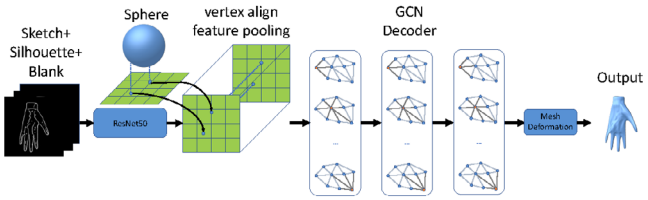

Figure 6: *The architecture of the Shape Deformation Network (SD-Net) incorporates a three-channel input that is processed through a pre-trained ResNet50 to extract a feature map. Vertex features are subsequently derived from this feature map utilising a combination of projection techniques and bilinear interpolation. These extracted features are then fed into a Graph Convolutional Network (GCN) decoder, which is tasked with predicting the deformation offsets for each vertex.*

6.1. Shape Deformation Network Architecture

The introduced SD-Net is designed to deform a reference sphere—comprised of 2562 vertices and 5120 faces and centred at the origin—into a 3D shape guided by a sketch. The input to the SD-Net is a 3-channel image that includes a binary sketch image, a binary mask image and a blank image. The binary sketch image is sparse, making it challenging to determine whether 0-valued pixels are parts of the foreground object (inside the object) or the background, solely based on their neighbouring pixels. This complexity renders it difficult for convolutional layers to directly extract effective features from the binary sketch image alone. By stacking a binary mask image alongside the binary sketch image, it becomes possible to differentiate foreground 0-valued pixels from those in the background. ResNet50 processes these inputs to generate a feature map $f_{map} \in \mathbb{R}^{C\times W\times H}$ of dimensions $2048 \times 5 \times 5$. A blank image is incorporated to satisfy the input constraints of the pre-trained ResNet50 model.To extract vertex features from f_{map}, our network employs a vertex align feature pooling approach similar to that described in [WZL*18]. This method pools per-vertex features from f_{map} indexed by their 2D projection coordinates determined using the perspective projection matrix $\mathcal{P}$ utilised in the rendering process. Bilinear interpolation is used to pool perceptual features from f_{map}, resulting in 2048-dimensional features for each vertex, which are then available for subsequent processing by graph convolutional layers. The architecture of the SD-Net is depicted in Figure 6.

The reference sphere mesh can be represented by a graph $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$, where $\mathcal{V} \in \mathbb{R}^{V\times 3}$ stores vertex positions; $\mathcal{E} \in \mathbb{R}^{E\times 2}$ stores the indices of the vertices that form the edges, and $\mathcal{F} \in \mathbb{R}^{V\times f}$ stores vertex features.

Then, we define a graph convolutional layer on an irregular graph as follows:

$$f_p^{l+1} = w_0 f_p^l + \sum_{q\in\mathcal{N}(p)} w_1 f_q^l, \tag{5}$$

where $f_p^l \in \mathbb{R}^{d_l}$ and $f_p^{l+1} \in \mathbb{R}^{d_{l+1}}$ are the input and output feature vectors on vertex p of the graph convolutional layer, respectively; $\mathcal{N}(p)$ is the set of the neighbouring vertices, each of which shares an edge with p; and w_0 and w_1 are two learnable matrices of $d_l \times d_{l+1}$ that are applied to all the vertices. Three graph convolutional layers are employed, each followed by a LeakyReLU activation function. The input feature vector has a dimension $d_l = 2051$, which includes 2048 dimensions from the vertex features combined with the *XYZ* coordinates of each vertex on the sphere mesh. The dimension of the output feature vector is reduced to $d_{l+1} = 2048$. At the end of the process, a fully connected layer with a *tanh* activation function is utilized to predict the *XYZ* offsets for all vertices. We generate the front-view silhouette $\hat{S}^1$ by positioning a virtual camera aimed at the deformed mesh. The silhouettes from the other five views (i.e., left, right, back, top, and bottom), represented by $\hat{S}^2$, $\hat{S}^3$, $\hat{S}^4$, $\hat{S}^5$, and $\hat{S}^6$, respectively, are similarly rendered by properly positioning the virtual camera.

6.2. Displacement Mapping Network Architecture

The U-Net architecture, identical to that used in the sketch renderer, is employed to generate a displacement map from a 2-channel input (i.e., sketch image and its silhouette). This displacement map is applied to the vertices of the front faces (those faces oriented towards the front view), causing an offset in the direction of their current normal. Then, a normal map $\hat{I}_{norm}$ is created by orienting a virtual camera towards the front view of the mesh. This normal map $\hat{I}_{norm}$ is then inputted into our pre-trained SR-Net, which processes it to produce the sketch image $\hat{I}_s$.

6.3. Objective Function

To train the SD-Net and DM-Net, we minimise the following loss function:

$$\begin{aligned}\mathcal{L}_{total} = \mathcal{L}_{S\_L_2} + \mathcal{L}_{IOU} + \lambda_{L_1\_norm}\mathcal{L}_{L_1\_norm} + \lambda_{perc}\mathcal{L}_{perc} \\ + \lambda_{lap}\mathcal{L}_{lap} + \lambda_{edge}\mathcal{L}_{edge} + \lambda_m\mathcal{L}_m,\end{aligned} \tag{6}$$

where $\mathcal{L}_{S\_L_2}$ is the L_2 silhouette regression loss, $\mathcal{L}_{IOU}$ is the intersection over union regression loss, $\mathcal{L}_{L_1\_norm}$ is the L_1 loss, $\mathcal{L}_{perc}$ is the perceptual loss, $\mathcal{L}_{lap}$, $\mathcal{L}_{edge}$, and $\mathcal{L}_m$ are the Laplacian regulariser, the edge length regulariser, and the normal consistency regulariser, respectively. We describe these loss terms below.

L_2 silhouette regression loss ($\mathcal{L}_{S\_L_2}$) This term measures the squared distance between the rendered silhouettes $\{\hat{S}^j\}(j = 1 \sim 6)$ and the ground truth silhouette $\{S^j\}(j = 1 \sim 6)$ as follows:

$$\mathcal{L}_{\mathcal{S}}\_\mathcal{L}_{\epsilon} = \sum_{j=1}^{6}\frac{1}{n}\sum_{i=1}^{n}(w_i^j)^2(\hat{S}_i^j - S_i^j)^2 + (1 - w_i^j)^2, \tag{7}$$

where n denotes the total number of pixels, and $\left\{w_i^j\right\}$ $(0 \le w_i^j \le 1)$ denotes confidence maps.

In this work, we introduce a encoder–decoder architecture designed specifically to generate a confidence map from a sketch image input as shown in Table 1. Conceptually, the confidence map quantifies the uncertainty associated with the regression of each silhouette, where a higher uncertainty indicates a more complex shape for regression, and conversely, lower uncertainty indicates simpler shapes. This architecture is trained in an unsupervised fashion, thereby the ground truth confidence maps are not required. Our

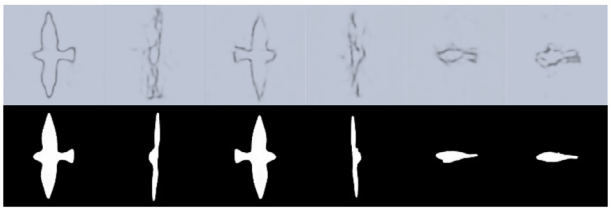

Figure 7: *Confidence maps for 6 views (top) and the corresponding predicted silhouettes (bottom) for a bird sample.*

confidence maps conceptually resemble those in previous studies [LPL*18, WRV20], which assess uncertainty arising from the transformation from 2D space to 3D space. Examples of the confidence maps and their corresponding predicted silhouettes are depicted in Figure 7.

The confidence maps serve an important function in weighting the losses from different views in the Intersection Over Union (IOU) calculations. Given that the complexity of silhouettes varies across different views, it is essential to differentiate among them. This differentiation allows for greater emphasis to be placed on more complex silhouettes during the training process.

Intersection over union (IOU) regression loss ($\mathcal{L}_{IOU}$) Since $\hat{S}^j (j = 1 \sim 6)$ and $S^j (j = 1 \sim 6)$ are binary images, it is difficult for the L_2 loss to converge when $\hat{S}^j$ is significantly different from S^j. The following IOU regression loss $\mathcal{L}_{IOU}$ can provide a direction of convergence at the beginning of training:

$$\mathcal{L}_{IOU} = \sum_{j=1}^{6}\left(1 - \frac{\sum_{i=1}^{n}(1 - w_i^j)2\hat{S}_i^j \cdot S_i^j}{\sum_{i=1}^{n}(1 - w_i^j)(\hat{S}_i^j + S_i^j)}\right), \tag{8}$$

where $\left\{w_i^j\right\}$ $(0 \le w_i^j \le 1)$ denotes the confidence maps. Since $\hat{S}^j$ and S^j are binary image representations, $\sum_{i=1}^{n}(1 - w_i^j)2\hat{S}_i^j \cdot S_i^j$ denotes their weighted intersection and $\sum_{i=1}^{n}(1 - w_i^j)(\hat{S}_i^j + S_i^j)$ denotes their weighted union.

It is noteworthy that the confidence maps utilised here are separated from the encoder–decoder network; therefore, they are not updated during the back-propagation of this specific loss term. In Equation (8), the role of the confidence maps is to weight the silhouettes differently due to their varying shape complexities and to prioritise the recovery of boundary details in silhouettes. The predicted confidence map w from the encoder-decoder network quantifies the uncertainty between the predicted silhouette $\hat{S}$ and the ground truth silhouette S, whereby $1 - w$ assigns higher values to less certain areas and lower values to more certain areas. Consequently, the weighted IOU loss focuses on regions with larger values, typically the boundaries and sharp features of the silhouettes. Figure 8 shows comparisons between the traditional IOU loss and the weighted loss, demonstrating how the confidence map weighted loss significantly improves the delineation of boundary and sharp features, such as the legs of the octopus examples shown in Figure 8b,c.

Geometric detail regression losses ($\mathcal{L}_{L_1\_norm}$ **and** $\mathcal{L}_{perc}$) The above silhouette related losses $\mathcal{L}_{S\_L_2}$ and $\mathcal{L}_{IOU}$ can improve the regression on the shape of the 3D object, yet lack of geometric details.

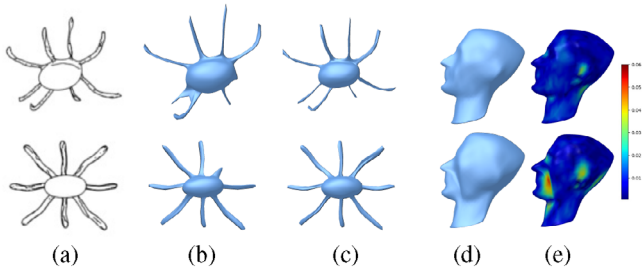

Figure 8: *The result comparison between using the traditional IOU loss (b) and the confidence map weighted IOU loss proposed in our work (c). The input sketches are shown in (a). The comparison of the reconstructed 3D object is presented alongside a heatmap visualisation of per-vertex errors relative to the ground truth. The top row of (d) and (e) displays the outcome when the SR-Net, inclusive of geometric detail regression losses, is incorporated during training. Conversely, the bottom row of (d) and (e) illustrates the results obtained without the inclusion of the SR-Net in the training process.*

Based on the pre-trained SR-Net that takes a rendered normal map as the input and outputs its corresponding sketch image I^s, two loss terms are introduced to optimise the reconstructed geometric details. The first L_1 loss term $\mathcal{L}_{L_1\_norm}$ can be formulated as:

$$\mathcal{L}_{L_1\_norm} = \frac{1}{\sum m_i'}\sum_{i=1}^{n} m_i' |\hat{I}_i^s - I_i^s|, \tag{9}$$

where $\left\{m_i'\right\}$ is the binary inner mask inside the occluding contours of the input sketch. This term focuses on the regression of geometric details only to eliminate the unnecessary influence of the occluding contours on the relevant vertices.

The second term $\mathcal{L}_{perc}$ is the perceptual loss. Similar to the multi-level masked perceptual loss for $\mathcal{L}_{perc}^{SR}$ (refer to Equation (3)), multi-level perceptual losses are summed up (Equation (10)). The perceptual loss at the kth level $\mathcal{L}_{perc^{(k)}}$ is defined as follows:

$$\mathcal{L}_{perc} = \sum_{k=1}^{4} \mathcal{L}_{perc^{(k)}}, \tag{10}$$

$$\mathcal{L}_{perc^{(k)}} = \frac{1}{\sum m_i}\sum_{i=1}^{n} m_i (e_i^{(k)}(\hat{I}_i^s) - e_i^{(k)}(I_i^s))^2, \tag{11}$$

where $\{m_i\}$ is the binary mask that contains all sketch lines. This term enhances the regression of both geometric details and the front-view silhouette. Since the SR-Net is differentiable, $\mathcal{L}_{L_1\_norm}$ and $\mathcal{L}_{perc}$ can be back-propagated during training to regress the geometric details. Figure 8d,e presents a comparative analysis of the results with and without the geometric detail regression losses. The comparison result clearly demonstrates that the integration of the SR-Net with the geometric detail regression losses significantly enhances the capability of the SD-Net to accurately recover geometric details. To thoroughly evaluate the effectiveness of SR-Net and corresponding geometric detail regression losses, we conduct an additional comparison against a baseline configuration that uses only

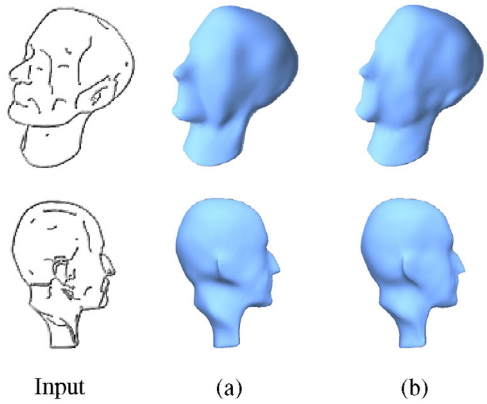

Figure 9: *Comparison between (a) using ground-truth depth and normal maps with L2 norm loss functions (without SR-Net and relative loss functions) and (b) using SR-Net with relative loss functions. The SR-Net variant (b) yields improved alignment of sketch strokes with recovered geometry, while (a) produces less accurate reconstruction of fine details.*

L2 norm loss with ground-truth depth and normal maps, as shown in Figure 9. While the L2-based baseline benefits from strong supervision on depths and normals, it lacks explicit mechanisms for enhancing fine-scale geometry. In contrast, SR-Net, coupled with the geometric detail regression losses, not only enforces consistency in depth and normal predictions but also promotes the recovery of intricate local structures and subtle surface variations. This results in more faithful reconstruction of fine details, such as sharper object boundaries and more accurate alignment between sketch strokes and recovered geometry, which are less evident in the L2-only approach.

Smoothness regularisers ($\mathcal{L}_{lap}$, $\mathcal{L}_{edge}$, **and** $\mathcal{L}_m$) To enforce the smoothness of the created 3D object, we also employ the following three regularisers. A Laplacian regularisation, $\mathcal{L}_{lap}$, is added to constrain the displacements of the vertices, which is defined below.

$$\mathcal{L}_{lap} = ||L \cdot v||_1, \tag{12}$$

where L is the Laplacian matrix ($V \times V$) of the 3D object, and v is the $V \times 3$ vector formed by concatenating all the vertex positions of the 3D object. Notice that L is fixed during training. An edge length regulariser, $\mathcal{L}_{edge}$, is added to minimise the edge lengths of the 3D object by penalising the flying vertices that often cause long edges, and it is defined as follows:

$$\mathcal{L}_{edge} = \sum_{p} \sum_{q \in \mathcal{N}(p)} ||p - q||_2^2, \tag{13}$$

where p is a vertex on the mesh, and $\mathcal{N}(p)$ denotes the set of the neighbouring vertices of p. This helps avoid stretched triangles that can distort surface details, improves numerical stability during optimisation, and preserves high-frequency geometry. Figure 15 illustrates that, with this term, the reconstructed mesh exhibits smoother surfaces and more consistent triangle shapes compared to the case without this regularisation. We carefully tune its weight to prevent the vertex collapse. The last normal consistency loss, $\mathcal{L}_m$, computes the normal consistency for all the pairs of neighbouring faces as follows:

$$\mathcal{L}_m = \sum_{(n_i, n_j) \in F} (1 - cos < n_i, n_j >), \tag{14}$$

where n_i and n_j are the normals of two adjacent faces, $< n_i, n_j >$ is the dihedral angle between two adjacent faces, and F denotes the set of all adjacent face pairs.

During training, we experimentally set $\lambda_{L_1\_norm} = 10, \lambda_{perc} = 0.1$, $\lambda_{lap} = 5, \lambda_{edge} = 0.2$, and $\lambda_m = 0.3$. The training process of the SD-Net is end-to-end and thus no manual fine-tuning is required.

7. Training Details

In our training data, one training sample consists of a sketch, its mask, normal map, and six silhouettes from different views of the corresponding 3D shape.

We describe the implementation details of our model training below. First, the SR-Net was trained using the Adam optimiser, processing batches of 16 input normal maps, each resized to 128×128 pixels. The generated sketch images were also sized at 128×128. We trained the network for 400 epochs with a learning rate of 1×10^{-4}. Next, the SD-Net and DM-Net were trained with the Adam optimiser on batches of 12 input sketch images, each resized to 128×128 pixels. Integrated within the SD-Net, the encoder-decoder architecture for generating confidence maps was trained in an unsupervised manner. This component outputs a 6-channel 128×128 confidence map, where each channel corresponds to the silhouette from one of the six selected views. Note that the learnable parameters of the SR-Net were not updated during the training phase of SD-Net and DM-Net. The SD-Net and DM-Net training proceeded for 800 epochs with a learning rate of 1×10^{-4}.

8. Experiment Results and Evaluations

This section provides the 3D modelling outcomes produced by our method and comparisons between our method and some state-of-the-art methods. After that, an ablation study is presented to evaluate the significance of some key modules in our approach. Notice that all sketches used during evaluation were hand-drawn by humans and were not seen during training. To ensure consistency in sketching style and to maintain a fair evaluation setting, the human sketches were drawn based on 2D shapes from the same dataset used for training. Finally, a user study and runtime statistics are discussed to provide insight into the computational effectiveness and efficiency of our method.

Local geometry detail generation. Our method is capable of automatically processing input sketches that contain varying levels of detail. As demonstrated in Figure 10, our method effectively generates 3D models that exhibit enhanced geometric details when the provided sketches include a greater number of inner strokes. This enhancement is more prominently observable in the zoomed sections of the figure.

Local geometry detail editing. To test the local editing ability of our model, we asked users to add sketch lines without adhering to the style present in the dataset. We provide the user with some

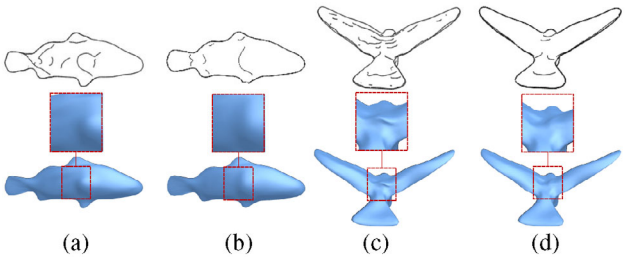

Figure 10: *Some results generated by our approach based on input sketches that vary in the level of contained details. (a) and (c) display the input sketches that feature more inner details at the top, and the corresponding 3D models generated by our method are shown at the bottom. Conversely, (b) and (d) illustrate input sketches with fewer inner details at the top, with the corresponding 3D models produced by our approach depicted at the bottom.*

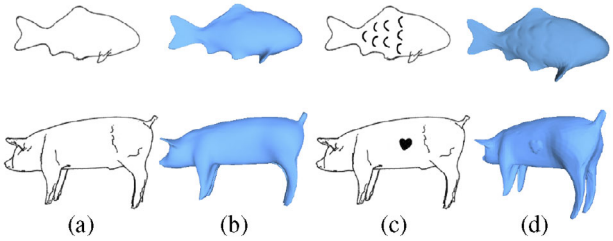

Figure 11: *Examples of the users adding customised details. (a) shows the original input sketches; (b) shows the corresponding 3D models generated from these sketches; (c) shows various users' customised sketch lines; and (d) shows the 3D models that have been modified to include these user-specified details.*

original sketches, ask users to add more sketches, and utilise our method to generate the relative 3D models. Figure 11 illustrates the inputs and the resulting outputs from this experiment. The models generated from users' sketches preserve global geometric properties while accurately reflecting local details based on the users' input.

Results on hand-drawn vs. synthesised sketches. Figure 12 shows the experiment results validate the novice-friendly nature of our method, highlighting that users can effortlessly draw sketch lines while still achieving a high degree of generalisation with the model. The results also demonstrate that precise line accuracy is not necessary for the generation of high-quality 3D models using our method. To evaluate the generalisability of our method across varying sketch styles, we conducted experiments on diverse inputs, including (a) cat, (b) big-ear head, (c) bird, and (d) insect. These sketches vary in both abstraction levels and artistic styles. As shown in Figure 13, our approach demonstrates robustness to stylistic variations, maintaining both faithfulness to the input and precise alignment between sketch strokes and the reconstructed geometry. Notably, without the semantically annotated strokes, our method can interpret a wide range of stroke types, including structural connections between different parts of the model (e.g., body and arms). Such understanding is implicitly learned from the training data, enabling the reconstruction of coherent and structurally consistent 3D models across diverse sketching styles.

To further examine the choice of initialisation, we conducted an experiment replacing the predefined sphere with alternative shapes (cube and heart). As shown in Figure 14, the results are randomly deformed and fail to recover meaningful geometry, since the model is trained with a sphere template and all predicted deformations are implicitly conditioned on its topology. The sphere proves to be the most effective initialisation because it offers isotropic deformation, smooth and simple topology, and a uniform mesh distribution, which together ensure stable optimisation and coherent reconstructions. In contrast, cube- and heart-based templates introduce anisotropy, sharp features, and irregular mesh structures, leading to unstable deformations and degraded reconstruction quality.

Evaluation of Mesh Quality. We evaluated the quality of the generated meshes by performing a self-intersection test, as illustrated in Figure 15. The smoothness regularisers incorporated during training effectively suppress the vast majority of self-intersection cases, ensuring clean and watertight surfaces. Only rare instances involving extremely sharp geometric features, such as the tail of the dolphin or the slender legs of the cow and dog, have minor self-intersections, as highlighted in red in Figure 15. These occurrences are infrequent and localised, having negligible impact on the overall geometry. Overall, our results demonstrate that the generated meshes are of consistently high quality and well-suited for practical use.

Comparison with occluding contour-based methods. We evaluated our method against occluding contour-based 3D modelling approach [JFD20]. This method primarily focuses on the occluding contours of an input sketch, whereas our method incorporates both the contours and the inner feature strokes of the input sketch. To ensure a fair comparison, we adhered strictly to the training parameters outlined in [JFD20] for its implementation, including the recommended post-processing technique described therein to generate smooth 3D meshes from the voxel representations produced. Figure 16 presents a comparison involving two examples: one is a man-made object (an airplane) and the other is a natural object (an octopus). The results from our approach successfully retain the essential features of the input sketches and demonstrate greater accuracy compared to the results from [JFD20].

Result gallery. Our method was applied to generate various 3D models from single sketch inputs, with the modelling results displayed in Figure 17. The examples shown in this figure illustrate the robustness of our method in handling various categories of input sketches. For a comprehensive view of these results, please refer to the accompanying demo video, which offers 360-degree visualisations. Notably, all outcomes depicted in Figure 17 were produced directly by the mesh generation network, without any subsequent post-processing. In contrast, many existing sketch-based modelling techniques (e.g., [JFD20, LPL*18, LGK*17]) typically require post-processing of the predicted output.

8.1. Comparisons with State of the Art Methods

To evaluate the effectiveness of our approach, we compared it with some state of the art, sketch-based 3D modelling methods, including deep learning based methods [LGK*17, LPL*18, ZGG21, GYS*22, BKD*24], geometric inference based methods [NISA07],

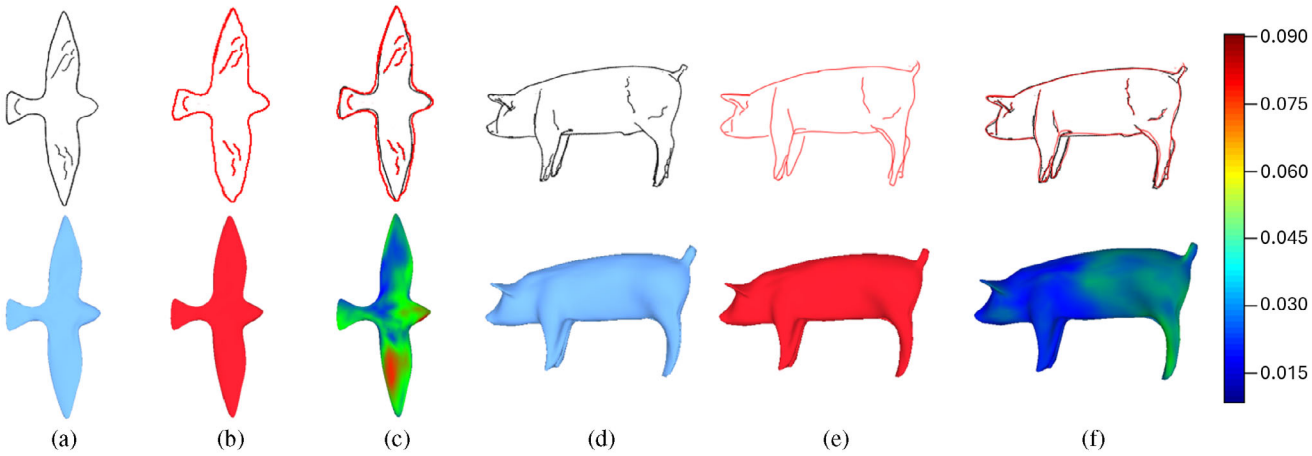

Figure 12: *(a) and (d) are sketch images in the test set and corresponding 3D models generated by our method; (b) and (e) are user-drawn sketches and corresponding 3D models generated by our method; and (c) and (f) show the heat maps that illustrate the per-vertex differences between the models in (a)-(b) or in (d)-(e). The heatbar shows the measurement for column (c) and (f). Results show that our method enables high-quality 3D reconstruction from freehand sketches, demonstrating strong generalisation and robustness to input strokes.*

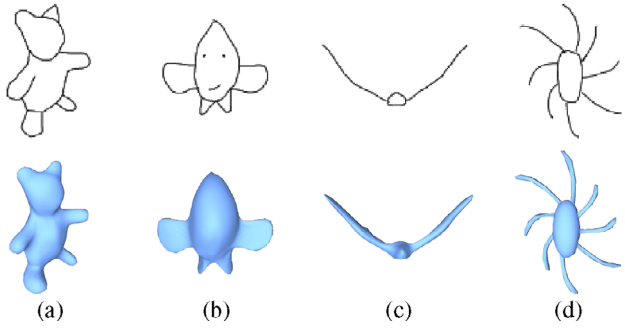

Figure 13: *User-provided sketches in various styles—(a) cat, (b) big-ear head, (c) bird, and (d) insect—alongside their corresponding reconstructed results. Rather than strictly replicating the input sketches, our method prioritises producing faithful 3D models while maintaining close alignments with the input sketches.*

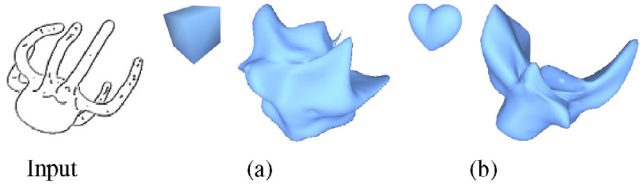

Figure 14: *Results when replacing the initial sphere with other templates: (a) input sketch, (b) cube mesh, and (c) heart mesh. Both alternatives lead to unstable deformations and fail to capture meaningful geometry.*

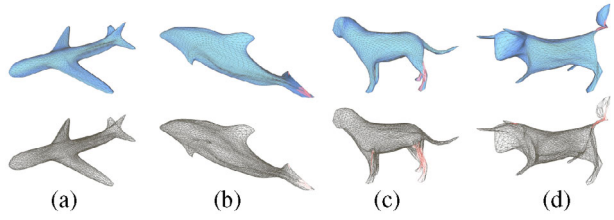

Figure 15: *Mesh quality evaluation via self-intersection tests for (a) airplane, (b) dolphin, (c) dog, and (d) cow. Smoothness regularisers prevent most intersections, with rare cases in sharp features (highlighted in red) having minimal impact.*

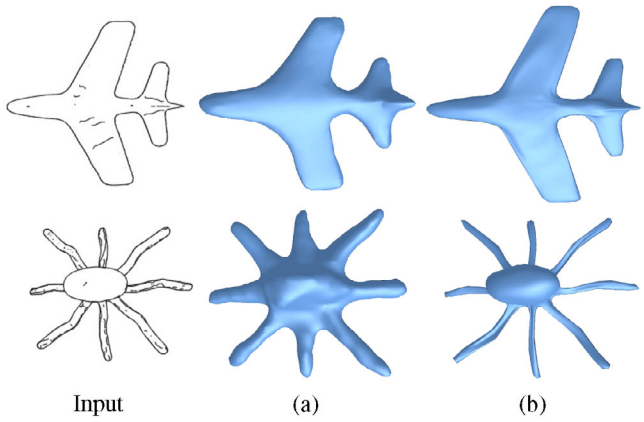

Figure 16: *Two comparison examples between [JFD20] (a) and our approach (b).*

semantically-annotated sketch based methods [LPL*17, LPL*18], and occluding contour-based methods [JFD20].

Comparison with deep learning based methods. For the evaluation of our method, we selected two recent deep learning-based approaches [LGK*17, LPL*18] for comparison. Specifically, Lun et al. [LGK*17] developed a deep encoder–decoder network that generates depth and normal maps from pre-defined views, which are

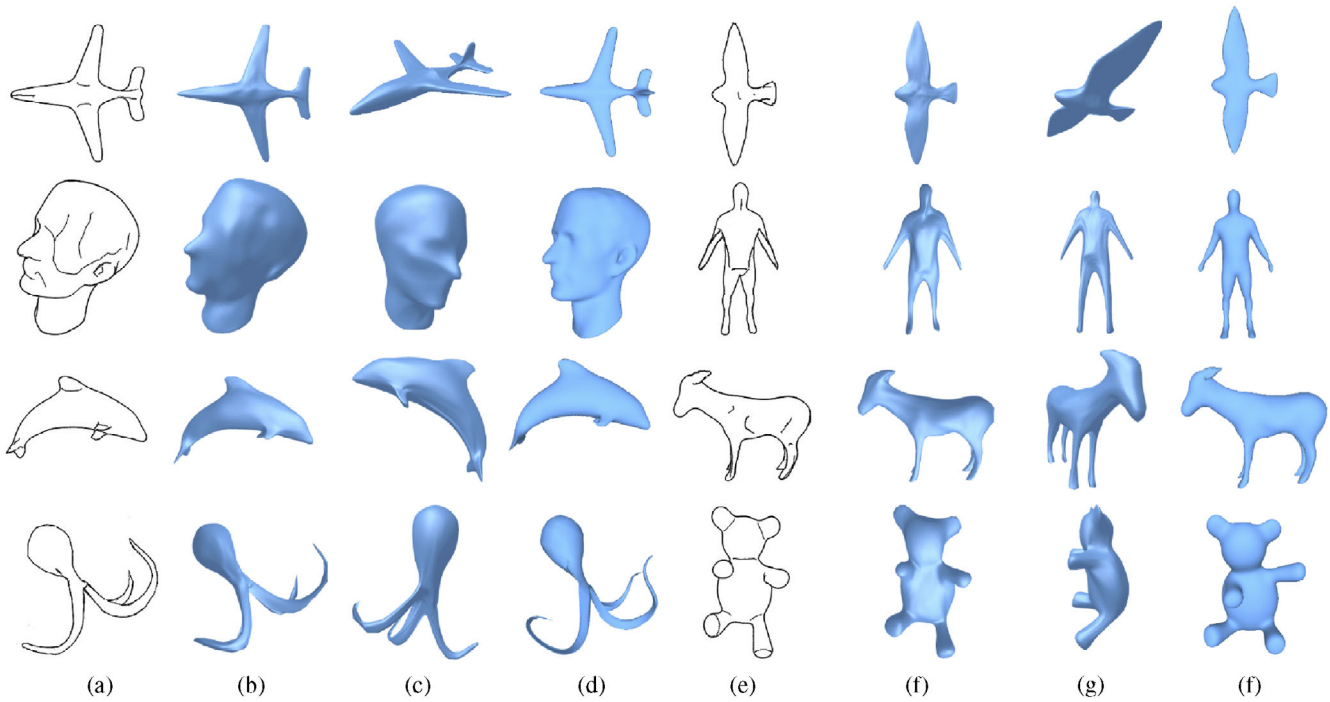

Figure 17: *This gallery presents the outcomes of sketch-based modelling achieved through our method. For illustration, (a) and (e) display input sketches from various categories, specifically jet, head, dolphin, and octopus from top to bottom in (a), and bird, human, animal, and teddy from top to bottom in (e). For each example, we not only present the 3D model from the same viewpoint as the input sketch in (b) and (f), but we also provide a visualization of the 3D model from another randomly selected viewpoint in (c) and (g). To further demonstrate generalisation of the trained network, we put the closest training shapes alongside these examples shown in (d) and (f).*

further fused to create a 3D model. Li et al. [LPL*18] introduced a CNN-based method designed to predict the front view surface of a 3D model from a single-view sketch input. Unlike the approach in [LGK*17], which requires category-specific shapes during training, our method does not have such limitations. To ensure a fair comparison, all three methods ([LGK*17], [LPL*18], and ours) were trained and evaluated using the character dataset published by Lun et al. [LGK*17], noted for its high-quality geometric details. The same front view sketches were used across all methods. Additionally, the deep learning networks for each method were trained for 10 epochs with 10,000 samples, adhering to the protocol suggested by Li et al. [LPL*18]. Figure 18 presents comparisons of the results by the three methods alongside the ground truth. The results reveal that our method produces 3D models with superior geometric details that align better with the ground truth, particularly in areas with sharp features, as highlighted in the zoomed sections of Figure 18. Note that the 3D models shown in Figure 18b (by [LGK*17]) and Figure 18c (by [LPL*18]) are originally reported in the work of [LPL*18].

We also employed two quantitative metrics for comparison among the three methods: (i) surface normal distance and (ii) depth map error, as these metrics were utilised in [LPL*18]. Specifically, surface normal distance is calculated by first determining the angle between the normal at each surface point and the normal at its closest surface point, followed by averaging these angles. The depth map error is assessed by calculating the average absolute pixel-depth difference between the predicted depth map and the ground-truth depth map, ensuring both maps are normalised to the same mean value before comparison.

Table 2: *Quantitative comparison of the three methods ([LGK*17], [LPL*18], and our approach).*

Method	Depth map error	Surface normal distance
[LGK*17]	0.023	22.4°
[LPL*18]	0.033	18.6°
our approach	0.010	18.3°

Table 2 presents the performance metrics for all three methods using the specified character dataset. Note that the values for these metrics for [LGK*17] and [LPL*18] were directly extracted from the paper of [LPL*18]. As indicated in the table, our method outperforms the other two in terms of both surface normal distance and depth map error. It is noteworthy that our approach achieves these results without utilising normal maps and depth maps of 3D objects for model training; in contrast, these maps are employed in the training processes of [LGK*17, LPL*18].

We also utilised an example provided in the work of Doodle-Your-3D [BKD*24] to conduct a visual comparison with [ZGG21, GYS*22, BKD*24] in Figure 19. We chose the airplane for this

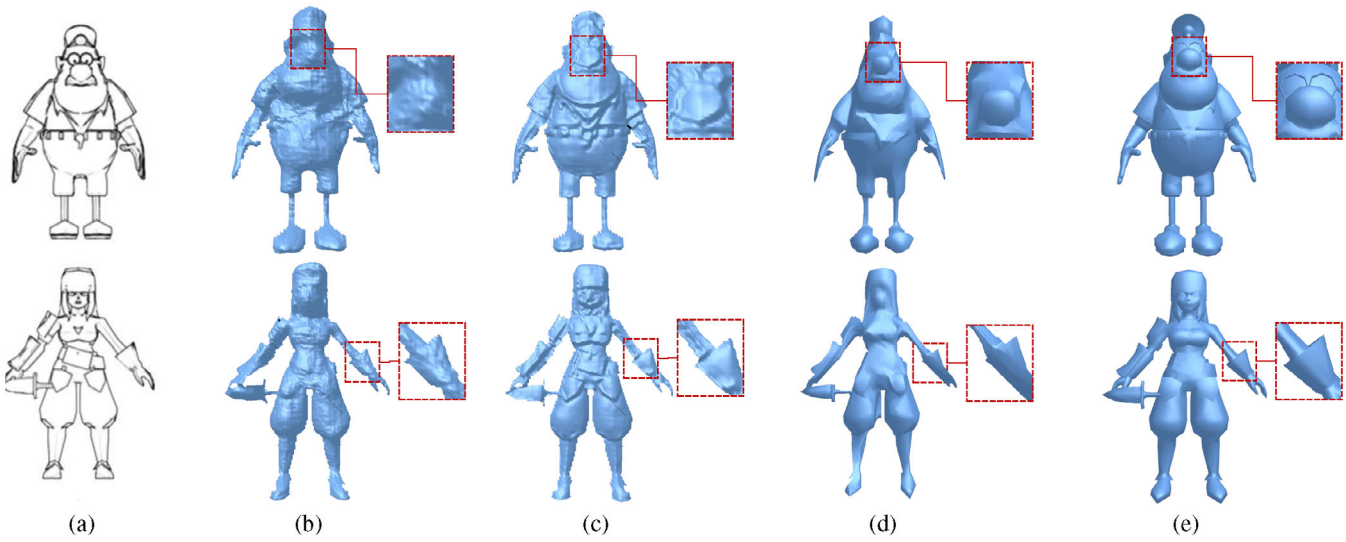

Figure 18: *Result comparison among [LGK*17] (b), [LPL*18] (c), our method (d), and the ground-truth 3D models (e), given a single front-view sketch as the input (a). All the methods were trained and tested on the same character dataset published by Lun et al. [LGK*17].*

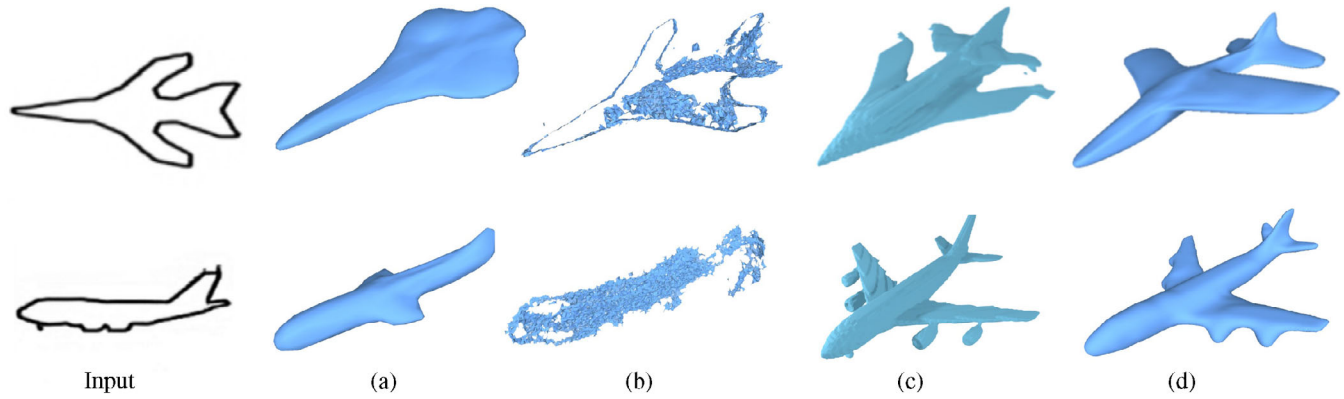

Figure 19: *Comparison among [ZGG21] (a), [GYS*22] (b), [BKD*24] (c) and our approach (d). Our approach can preserve the faithfulness of the generated 3D models and align with the input sketch.*

comparison. All methods were tested with the same sketch input provided in [BKD*24]. We utilised the ball-pivoting algorithm [BMR*99] to convert the point cloud result from sketchsampler [GYS*22] to mesh. In contrast to other methods, the 3D result generated by our approach is much more faithful to the input sketch. Instead of simply follow the input sketch, our method tends to keep the faithfulness of the generated 3D models as a priority and align the 3D models with the input sketches as closely as possible.

As shown in Figure 20, we compared our result with two recent image-to-3D reconstruction methods, TRELLIS [XLX*25] and SPARC3D [LWZ*25]. When provided with only the sketch input, both TRELLIS and SPARC3D struggled to produce coherent shapes, often resulting in fragmented or incomplete meshes. Even when augmented with normal maps generated via OpenAI ChatGPT to align with the input sketch, their outputs failed to form an integrated 3D structure and exhibited poor alignment with the input strokes. (ChatGPT prompt: "help me generate the normal maps aligned with the input sketch"). In contrast, our method consistently produced a unified, well-structured mesh that remained faithful to the input sketch, demonstrating superior robustness and alignment capabilities.

Comparison with geometric inference-based methods. We also evaluated our approach against classical geometric inference-based 3D modeling methods [NISA07]. These traditional methods typically employ a predefined geometric smoothness prior, such as biharmonic equations, to guide the modelling process. In interactive modelling, each input stroke is used to define a new boundary or feature curve in 3D, which subsequently alters the current base shape.

In contrast, our method treats 3D modelling as an end-to-end process, offering a user-friendly interface for modelling. Additionally, the geometric priors and multi-view priors learned within our

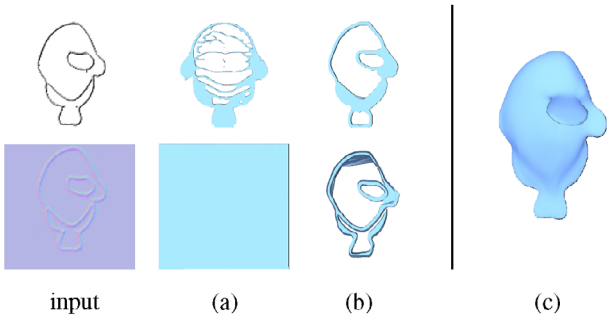

Figure 20: *Comparison with recent image-to-3D methods: (a) TRELLIS [XLX*25] (a), SPARC3D [LWZ*25] (b), and (c) ours. Top row: results using the sketch as the sole input. Bottom row: results using the sketch augmented with normal maps generated via ChatGPT. Trellis and SPARC3D struggle to produce a single, integrated mesh and to stay well aligned with the inputs.*

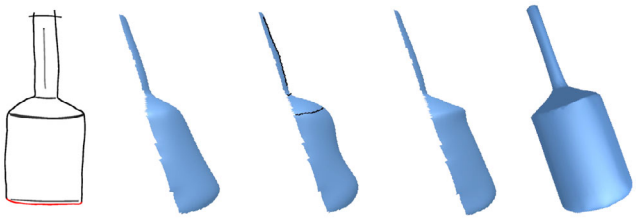

Figure 21: *Result comparison among biharmonic surfaces [NISA07] (b and c), [LPL*18] (d), and our approach (e). Given the input sketch of a bottle shape (a), the biharmonic surface (b) is obtained using the boundary positions and normal constraints that are computed by [LPL*18]. Lifted spatial strokes (bold black curves) computed from [LPL*18] are added to (b) as additional constraints to generate the result in (c). The 3D surface in (d) is obtained through post-processing (i.e., lift the bottom of the bottle, refer to the red part of the sketch) the prediction by [LPL*18]. Our method generates a whole 3D model (e), instead of a surface, based on the black strokes alone. Note that the 3D models in (b)–(d) are directly taken from the work of [LPL*18].*

framework tend to be more accurate than those derived from hand-crafted rules. Figure 21 illustrates a comparison of one bottle sketch example across various methods. The results clearly demonstrate that our approach more effectively retains the essential characteristics of the input sketch and constructs a complete 3D shape, unlike the partial 3D surface generated by [LPL*18] (as seen in Figure 21d). Moreover, the biharmonic surface shown in Figure 21b, constrained by boundary positions and normals, loses key features and appears overly rounded, even with the inclusion of additional inner curve constraints (Figure 21c).

Comparison with semantically annotated sketch-based methods. We evaluated our method by comparing it with two semantically-annotated sketch-based modelling approaches [LPL*17, LPL*18]. Both methods require the use of colour-coded, semantically-annotated strokes (identifying features such as ridges, valleys, curvature lines, and sharp features) within the

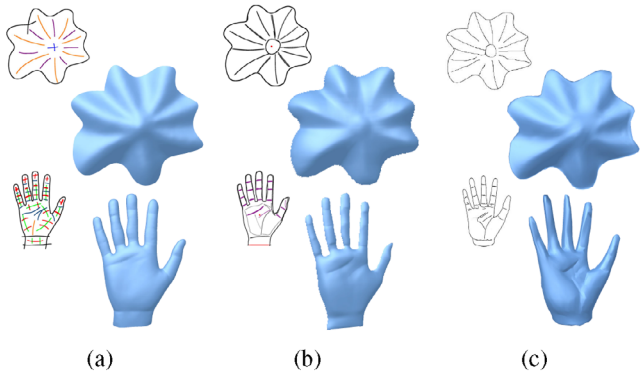

Figure 22: *Comparison among [LPL*17] (a), [LPL*18] (b), and our approach (c). Despite producing similar 3D models with geometric details, our method does not need semantically annotated strokes (for ridges, valleys, curvature lines, etc.) in the input sketch. Note that the 3D models in (a) and (b) are directly taken from the work of [LPL*18].*

input sketch to deduce depth and normal information for the 3D shape. In contrast, our approach does not require semantically annotated strokes in the input sketch. Two examples in Figure 22 show that similar surface details can be achieved by the methods in [LPL*17, LPL*18], and our approach using different sketch inputs. Compared to [LPL*17, LPL*18], our method requires simpler sketch inputs (i.e., without semantic annotations) to generate comparable 3D models with detailed geometric features. Additionally, in our method, the 3D mesh results are directly outputted from the SD-Net without any need for post-processing, whereas the method in [LPL*18] involves some post-processing steps on the predicted output.

8.2. Ablation Study

To assess the effectiveness of some key components in our architecture, we conducted an ablation study as described below. Essentially, we developed three reduced versions of our method, in addition to utilising the term 'full architecture' to denote the complete version of our approach.

- (*baseline*) A baseline architecture using only the traditional IOU loss to regress the silhouettes from six different views.
- (*baseline + confidence map weighted losses*) Both the confidence map weighted IOU loss (Equation 8) and the confidence map weighted L_2 regression loss (Equation 7) are used to calculate the difference between S and $\hat{S}$.
- (*baseline + SR-Net*) The SR-Net is added to the baseline architecture, and only the traditional IOU loss is used to regress the silhouettes from 6 different views (i.e., confidence maps are not used).
- (*baseline + ND*) The ground-truth depth and normal maps are incorporated into the baseline architecture via L2 norm loss functions, while the silhouettes from six different views are regressed using only the traditional IoU loss (i.e., without confidence maps).

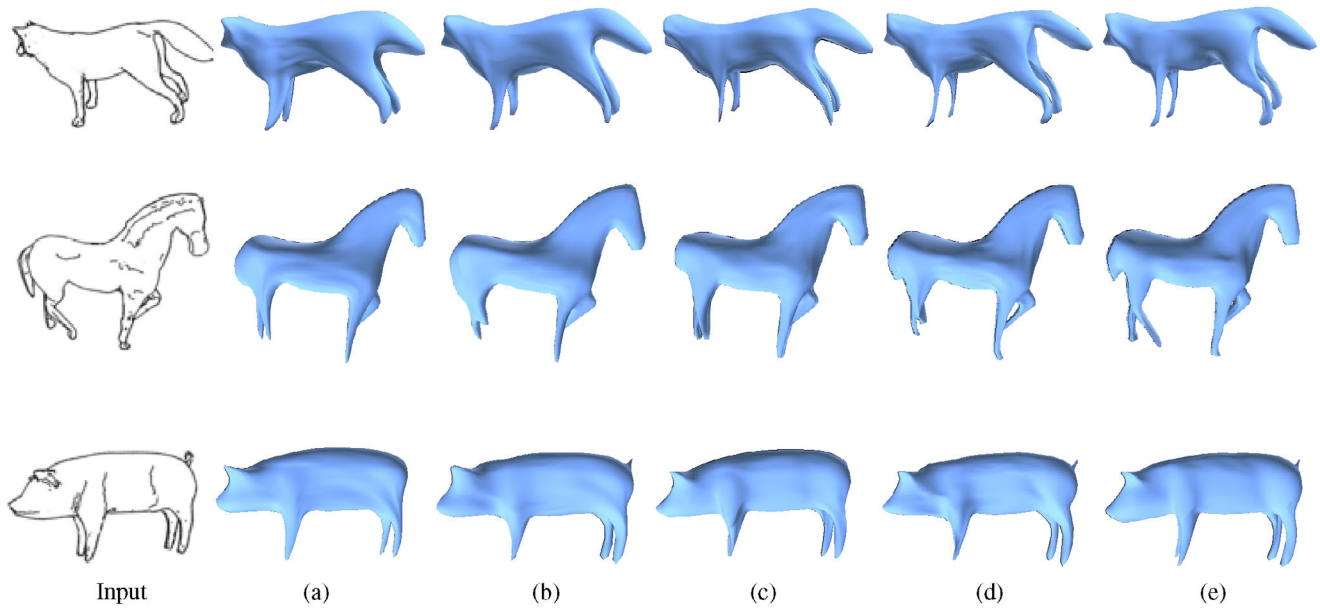

Figure 23: *Comparison of the results by the three reduced versions and the full architecture version in our ablation study. (a) shows the results by the baseline version, (b) shows the results by the baseline + confidence map weighted losses version, (c) shows the results by the baseline + SR-Net version, (d) shows the result by the baseline + ND version, and (e) shows the results by the full architecture version.*

Table 3: *Average test errors of different versions in our ablation study.*

Version	Depth map error	Normal distance
baseline	0.015	10.5°
baseline + confidence map weighted losses	0.017	11.0°
baseline + SR-Net	0.018	10.4°
baseline + ND	0.013	9.2°
full architecture	0.014	8.7°

Specifically, the networks in the above four versions maintain the same structural design as the SD-Net and DM-Net. These versions were trained using the same 'four-legs' dataset, which consists of 1824 samples for training, 240 samples for validation, and 240 samples for testing, over a span of 800 epochs. Figure 23 presents a visual comparison of three test examples derived from these versions. Additionally, the average test errors are also reported in Table 3, which includes the mean depth map error and the mean surface normal distance calculated across all 240 test samples.

As shown in Figure 23 and Table 3, the *baseline* version of our model fails to accurately generate surface details and sharp features, such as failing to recover the tails of both the pig and horse models. The *baseline + confidence map weighted losses* version enhances the visual representation of sharp features; however, both the depth map error and the surface normal distance error are higher due to the absence of depth constraints for sharp feature recovery. The addition of the SR-Net to the baseline (i.e., the *baseline + SR-Net* version) provides some depth constraints and better handles geometric feature strokes, leading to a reduction in the surface normal distance error. Nevertheless, due to the influence of the geometric detail regression loss on the IOU loss, the *baseline + SR-Net* version's ability to recover sharp features is diminished compared to the baseline, resulting in an increased depth map error, as evidenced by the incomplete recovery of the tails in both the horse and pig models. While the *baseline + ND* variant shows slightly improved depth and normal recovery due to stronger geometric constraints, the depth differences among all ablation variants remain minimal. Moreover, it fails to reconstruct certain global and local structures (e.g., legs and body geometry in Figure 23d). We contend that qualitative appearance alone is insufficient; alignment between sketch strokes and reconstructed geometry is more critical than accurately reproducing less relevant regions. The *full architecture* version of our approach results in the lowest depth map error and surface normal distance error. This improvement is attributed to the combination of confidence map weighted losses and the SR-Net, which together provide robust depth constraints and effectively restore sharp features.

8.3. User Study

We conducted a user study to evaluate the effectiveness of our method, which was structured into two distinct phases: 'Draw Sketches' and 'Complete Post-study Questionnaires.' The study involved a total of 16 participants (12 males and 4 females) from 26 to 37 years old who were recruited from a university campus.

In Phase 1, participants were provided with five objects from different categories to draw: pig, bird, airplane, fish, and octopus. The five objects were randomly selected from our dataset. For each object, its three different perspectives were available, from which participants selected one randomly as a reference to draw. Specifically, each participant selected one image rendered with a randomly

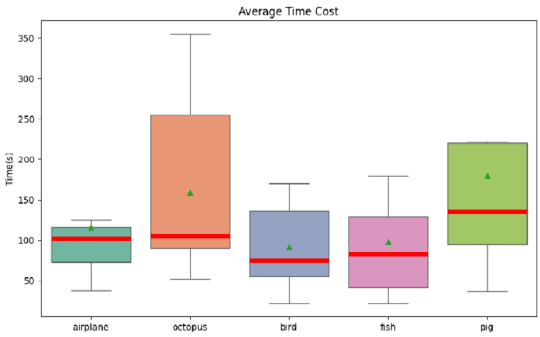

Figure 24: *The average sketching time spent for each object in our user study. The red line in each box denotes the median time, and the green triangle in each box denotes the average time.*

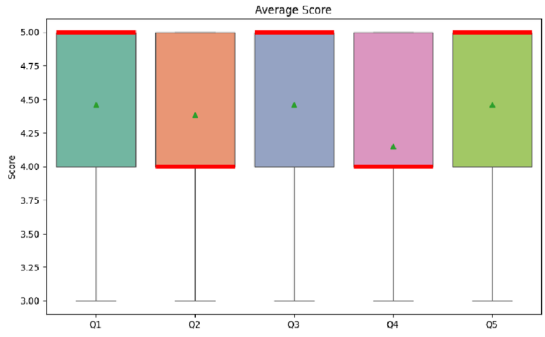

Figure 25: *The obtained average scores of the questions in our user study. The red line in each box denotes the median value, and the green triangle in each box denotes the average value.*

Table 4: *The average sketching time spent for each object in our user study (minute as unit).*

Statistic	Airplane	Octopus	Bird	Fish	Pig
Average	1.865	2.540	1.487	1.627	3.019
Median	1.658	1.750	1.250	1.417	2.092
Variance	1.176	2.470	0.658	1.753	5.688

Table 5: *The obtained average scores of the questions in our user study.*

Statistic	Q1	Q2	Q3	Q4	Q5
Average	4.4375	4.375	4.4375	4.18	4.4375
Median	5.00	4.00	5.00	4.00	5.00
Variance	0.529	0.383	0.529	0.429	0.529

perspective view as a reference to start drawing. Sketching time was recorded to calculate the average time spent, as depicted in Figure 24 and Table 4. The sketches drawn by participants were then used to generate 3D models by our method, which were subsequently displayed in a 3D viewer for phase 2 usage.

Phase 2 involved a post-study questionnaire where participants assessed their study experience based on both the drawing process and the resultant 3D models. Participants rated each question on a likert scale from 1 to 5, where a higher score indicates a more positive response. The used questions were: (Q1) Is it difficult to draw the sketch lines? (Q2) Do the generated 3D models adequately convey the intended information in your sketch line pictures? (Q3) Please rate the overall contours of the reconstructed models. (Q4) Please rate the overall details of the reconstructed 3D models. (Q5) Please rate the rationality of the models from multiple perspectives.

The aggregated results of the obtained user responses are presented in Figure 25 and Table 5.

Analysis of the obtained user study data revealed that the time taken to complete the sketches was generally short. As shown in Figure 24 and Table 4, most participants completed the task within several minutes. Notably, the complexity of the objects varied; for example, the sketching of an airplane was typically completed within 2 min, while more complex objects such as octopuses showed greater variability in completion time, ranging from 2 to 5+ min. Despite these variations, no significant difficulty in completing the sketches was reported. Furthermore, the obtained user response results, summarised in Figure 25 and Table 5, indicate that the average scores on all questions were above 4, suggesting a high level of participant satisfaction with our method.

8.4. Runtime Statistics

The SR-Net was trained for approximately 6 h, while the training duration for the SD-Net was extended to about 4 days, utilising a computing setup equipped with eight NVIDIA V-100 GPUs and an Intel Xeon 2.50GHz CPU. Post-training, the trained networks were deployed on a desktop computer featuring an NVIDIA P-100 GPU and an Intel Xeon 2.50GHz CPU. The runtime statistics for our method, as applied to all test cases shown in Figure 17, are detailed in Table 6. As indicated in this table, our approach demonstrates significantly higher runtime efficiency–approximately two orders of magnitude better–compared to the state-of-the-art method reported in [LPL*18], which took about 13 s to complete a sketch-based 3D modeling task on a system with an Intel Xeon 2.0GHz CPU and NVIDIA GeForce GTX 980 GPU. It is important to note that we did not conduct the tests with identical sketch inputs on the same hardware for the two methods.

Table 6: *Runtime statistics of our approach for all the test cases in Figure 17. s = second.*

Test sketch	Jet	Head	Dolphin	Octopus	Bird	Human	Animal	Teddy
run time	0.1966s	0.1968s	0.2248s	0.1971s	0.1997s	0.1968s	0.1967s	0.1945s

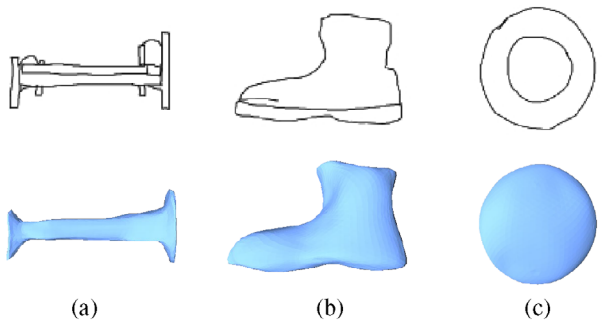

Figure 26: *Example failure cases where input sketches either do not correspond to any training shape category—(a) bed and (b) shoe—or represent non–genus-zero topology—(c) torus. In these cases, the reconstructed geometry deviates from the intended shape due to the lack of prior examples in the training data or topological constraints of the method.*

9. Conclusion and Limitations

In this paper we present a novel end-to-end approach to generate 3D models that exhibit plausible geometric details from a single-view sketch input, which does not require semantic annotations on input strokes.

Figure 26 illustrates several example cases where our method struggles due to limitations in the training data and topology constraints. Specifically, when the input sketches represent object categories not present in the training dataset—such as a bed or a shoe—the reconstruction fails to capture the intended structure, as the model lacks prior shape knowledge for these categories. Similarly, when the target shape has non–genus-zero topology, such as a torus, the reconstruction deviates from the desired geometry because the current framework assumes genus-zero surfaces. Also the resolution of the results is bounded by the reference sphere mesh. To overcome these limitations, future work should explore (1) integrating 3D Gaussian Splatting to initialise flexible point- or mesh-based templates for a wider range of topologies, followed by SR-Net refinement, and (2) adopting implicit surface representations to expand adaptability to varying genera while preserving fine surface details.

Acknowledgements

This research has been in part supported by US NSF IIS-2005430.

Ethics Statement

This research involved a user study with human participants. Participation was voluntary, and informed consent was obtained from all participants before the study. No personally identifiable information was collected beyond what was necessary for the study, and all data were stored and analyzed in anonymized form. The study design, data collection, and analysis followed the relevant ethical guidelines and regulations of the hosting institution. The authors are not aware of any conflicts of interest that could have influenced the results reported in this paper.

References

[BHSH*24] Binninger A., Hertz A., Sorkine-Hornung O., Cohen-Or D., Giryes R.: Sens: Part-aware sketch-based implicit neural shape modeling. *Computer Graphics Forum 43*, 2 (2024), e15015. https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12966

[BKD*24] Bandyopadhyay H., Koley S., Das A., Bhunia A. K., Sain A., Chowdhury P. N., Xiang T., Song Y.-Z.: Doodle your 3d: From abstract freehand sketches to precise 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2024), pp. 9795–9805.

[BKL*16] Bogo F., Kanazawa A., Lassner C., Gehler P., Romero J., Black M. J.: Keep it smpl: Automatic estimation of 3d human pose and shape from a single image. In *Computer Vision – ECCV 2016* (Cham, 2016), Leibe B., Matas J., Sebe N., Welling M., (Eds.), Springer International Publishing, pp. 561–578.

[BMR*99] Bernardini F., Mittleman J., Rushmeier H., Silva C., Taubin G.: The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics 5*, 4 (1999), 349–359. https://doi.org/10.1109/2945.817351.

[CGF09] Chen X., Golovinskiy A., Funkhouser T.: A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics (Proc. SIGGRAPH) 28*, 3 (2009), 73:1–73:12.

[CGL*19] Chen W., Gao J., Ling H., Smith E., Lehtinen J., Jacobson A., Fidler S.: Learning to predict 3d objects with an interpolation-based differentiable renderer. In *Advances In Neural Information Processing Systems* (2019).

[CLPK24] Choi C., Lee J., Park J., Kim Y. M.: 3doodle: Compact abstraction of objects with 3d strokes. *ACM Trans. Graph. 43*, 4 (2024), 107:1–107:13. https://doi.org/10.1145/3658156

[DFRS03] DeCarlo D., Finkelstein A., Rusinkiewicz S., Santella A.: Suggestive contours for conveying shape. *ACM Transactions on Graphics (TOG) 22*, 3 (2003), 848–855.

[FCSF16] Fu Q., Chen X., Su X., Fu H.: Natural lines inspired 3d shape re-design. *Graphical Models 85*, (2016), 1–10.

[FWX*13] Fan L., Wang R., Xu L., Deng J., Liu L.: Modeling by drawing with shadow guidance. *Computer Graphics Forum 32*, 7 (2013), 157–166.

[GIZ09] Gingold Y., Igarashi T., Zorin D.: Structured annotations for 2d-to-3d modeling. *ACM Transactions on Graphics (TOG) 28*, 5 (2009), 148.

[GLX*16] Guo X., Lin J., Xu K., Chaudhuri S., Jin X.: Customcut: On-demand extraction of customized 3d parts with 2D sketches. *Computer Graphics Forum 35*, 5 (2016), 89–100. https://doi.org/10.1111/cgf.12966

[GRYF21] Guillard B., Remelli E., Yvernay P., Fua P.: Sketch2mesh: Reconstructing and editing 3D shapes from sketches. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (October 2021), pp. 13023–13032.

[GYS*22] Gao C., Yu Q., Sheng L., Song Y.-Z., Xu D.: Sketchsampler: Sketch-based 3D reconstruction via view-dependent depth sampling. In *Computer Vision – ECCV 2022* (Cham, 2022), Avidan S., Brostow G., Cissé M., Farinella G. M., Hassner T., (Eds.), Springer Nature Switzerland, pp. 464–479.

[HKYM17] Huang H., Kalogerakis E., Yumer E., Mech R.: Shape synthesis from sketches via procedural models and convolutional networks. *IEEE Transactions on Visualization and Computer Graphics 23*, 8 (2017), 2003–2013.

[IMT99] Igarashi T., Matsuoka S., Tanaka H.: Teddy: A sketching interface for 3D freeform design. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques* (USA, 1999), SIGGRAPH '99, ACM Press/Addison-Wesley Publishing Co., p. 409–416. https://doi.org/10.1145/311535.311602

[JFD20] Jin A., Fu Q., Deng Z.: Contour-based 3D modeling through joint embedding of shapes and contours. In *Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2020), I3D '20, Association for Computing Machinery. https://doi.org/10.1145/3384382.3384518

[JRR*19] Johnson J., Ravi N., Reizenstein J., Novotny D., Tulsiani S., Lassner C., Branson S.: Accelerating 3D deep learning with pytorch3d. In *SIGGRAPH Asia 2020 Courses* (New York, NY, USA, 2019), SA '20, Association for Computing Machinery. https://doi.org/10.1145/3415263.3419160

[KAB20] Kocabas M., Athanasiou N., Black M. J.: Vibe: Video inference for human body pose and shape estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020).

[KBJM18] Kanazawa A., Black M. J., Jacobs D. W., Malik J.: End-to-end recovery of human shape and pose. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).

[KGK*20] Kulon D., Guler R. A., Kokkinos I., Bronstein M. M., Zafeiriou S.: Weakly-supervised mesh-convolutional hand reconstruction in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2020).

[KH06] Karpenko O. A., Hughes J. F.: Smoothsketch: 3d free-form shapes from complex sketches. *ACM Transactions on Graphics (TOG) 25*, 3 (2006), 589–598.

[KUH18] Kato H., Ushiku Y., Harada T.: Neural 3d mesh renderer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).

[LADL18] Li T.-M., Aittala M., Durand F., Lehtinen J.: Differentiable monte carlo ray tracing through edge sampling. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 37*, 6 (2018), 222:1–222:11.

[LB14] Loper M. M., Black M. J.: Opendr: An approximate differentiable renderer. In *Computer Vision – ECCV 2014* (Cham, 2014), Fleet D., Pajdla T., Schiele B., Tuytelaars T., (Eds.), Springer International Publishing, pp. 154–169.

[LGK*17] Lun Z., Gadelha M., Kalogerakis E., Maji S., Wang R.: 3d shape reconstruction from sketches via multi-view convolutional networks. In *2017 International Conference on 3D Vision (3DV)* (2017), IEEE, pp. 67–77.

[LLCL19] Liu S., Li T., Chen W., Li H.: Soft rasterizer: A differentiable renderer for image-based 3D reasoning. *The IEEE International Conference on Computer Vision (ICCV)* (Oct 2019).

[LPL*17] Li C., Pan H., Liu Y., Tong X., Sheffer A., Wang W.: Bendsketch: Modeling freeform surfaces through 2D sketching. *ACM Transactions on Graphics (TOG) 36*, 4 (2017), 125.

[LPL*18] Li C., Pan H., Liu Y., Tong X., Sheffer A., Wang W.: Robust flow-guided neural prediction for sketch-based freeform surface modeling. *ACM Transactions on Graphics 37*, 6 (2018), 238:1–238:12.

[LWZ*25] Li Z., Wang Y., Zheng H., Luo Y., Wen B.: Sparc3d: Sparse representation and construction for high-resolution 3D shapes modeling. *arXiv preprint arXiv:2505.14521* (2025), 1–12.

[NDVZJ19] Nimier- David M., Vicini D., Zeltner T., Jakob W.: Mitsuba 2: A retargetable forward and inverse renderer. *ACM Transactions on Graphics 38*, 6 (2019), 203:1–203:17. https://doi.org/10.1145/3355089.3356498

[NISA07] Nealen A., Igarashi T., Sorkine O., Alexa M.: Fibermesh: designing freeform surfaces with 3d curves. *ACM Transactions on Graphics (TOG) 26*, 3 (2007), 41.

[OLP*18] Omran M., Lassner C., Pons-Moll G., Gehler P. V., Schiele B.: Neural body fitting: Unifying deep learning and model based human pose and shape estimation. In *2018 International Conference on 3D Vision, 3DV 2018, Verona, Italy, September 5-8, 2018* (2018), IEEE Computer Society, pp. 484–494. https://doi.org/10.1109/3DV.2018.00062

[RDI10] Rivers A., Durand F., Igarashi T.: 3d modeling with silhouettes. *ACM Transactions on Graphics 29*, 4 (2010), 109:1–109:8.

[RFB15] Ronneberger O., Fischer P., Brox T.: U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention* (2015), Springer, pp. 234–241.

[SSB*19] Sahasrabudhe M., Shu Z., Bartrum E., Güler R. A., Samaras D., Kokkinos I.: Lifting autoencoders: Unsupervised learning of a fully-disentangled 3d morphable model

using deep non-rigid structure from motion. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)* (2019), pp. 4054–4064. https://doi.org/10.1109/ICCVW.2019.00500.

[SZ15] Simonyan K., Zisserman A.: Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (2015), Bengio Y., LeCun Y., (Eds.). http://arxiv.org/abs/1409.1556.

[WPVY19] Wan C., Probst T., Van Gool L., Yao A.: Self-supervised 3d hand pose estimation through training by fitting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 10845–10854. https://doi.org/10.1109/CVPR.2019.01111

[WRV20] Wu S., Rupprecht C., Vedaldi A.: Unsupervised learning of probably symmetric deformable 3d objects from images in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020).

[WZL*18] Wang N., Zhang Y., Li Z., Fu Y., Liu W., Jiang Y.-G.: Pixel2mesh: Generating 3D mesh models from single rgb images. In *Computer Vision – ECCV 2018* (2018).

[XLX*25] Xiang J., Lv Z., Xu S., Deng Y., Wang R., Zhang B., Chen D., Tong X., Yang J.: Structured 3D latents for scalable and versatile 3D generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference* (2025), pp. 21469–21480.

[XWJ*20] Xiang N., Wang R., Jiang T., Wang L., Li Y., Yang X., Zhang J.: Sketch-based modeling with a differentiable renderer. *Computer Animation and Virtual Worlds 31*, 4-5 (2020), e1939.

[XXM*13] Xie X., Xu K., Mitra N. J., Cohen-Or D., Gong W., Su Q., Chen B.: Sketch-to-design: Context-based part assembly. *Computer Graphics Forum 32*, 8 (2013), 233–245.

[ZFAT11] Zheng Y., Fu H., Au O. K., Tai C.: Bilateral normal filtering for mesh denoising. *IEEE Transactions on Visualization and Computer Graphics 17*, 10 (2011), 1521–1530. https://doi.org/10.1109/TVCG.2010.264

[ZGG21] Zhang S.-H., Guo Y.-C., Gu Q.-W.: Sketch2model: View-aware 3d modeling from single free-hand sketches. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2021), pp. 6012–6021.

[ZHH07] Zeleznik R. C., Herndon K. P., Hughes J. F.: Sketch: An interface for sketching 3D scenes. In *ACM SIGGRAPH 2007 courses* (2007), ACM, p. 19.

[ZPW*23] Zheng X.-Y., Pan H., Wang P.-S., Tong X., Liu Y., Shum H.-Y.: Locally attentional sdf diffusion for controllable 3d shape generation. *ACM Transactions on Graphics 42*, 4 (jul 2023). https://doi.org/10.1145/3592103

Supporting Information

Additional supporting information may be found online in the Supporting Information section at the end of the article.

Supplemental Video 1