3D Reconstruction from a Single Sketch via View-dependent Depth Sampling

Chenjian Gao*, Xilin Wang*, Qian Yu†, *Member, IEEE*, Lu Sheng, *Member, IEEE*, Jing Zhang, *Member, IEEE*, Xiaoguang Han, *Member, IEEE*, Yi-Zhe Song, *Senior Member, IEEE*, Dong Xu, *Fellow, IEEE*

***Abstract*—Reconstructing a 3D shape based on a single sketch image is challenging due to the inherent sparsity and ambiguity present in sketches. Existing methods lose fine details when extracting features to predict 3D objects from sketches. Upon analyzing the 3D-to-2D projection process, we observe that the density map, characterizing the distribution of 2D point clouds, can serve as a proxy to facilitate the reconstruction process. In this work, we propose a novel sketch-based 3D reconstruction model named *SketchSampler*. It initiates the process by translating a sketch through an image translation network into a more informative 2D representation, which is then used to generate a density map. Subsequently, a two-stage probabilistic sampling process is employed to reconstruct a 3D point cloud: firstly, recovering the 2D points (i.e., the x and y coordinates) by sampling the density map; and secondly, predicting the depth (i.e., the z coordinate) by sampling the depth values along the ray determined by each 2D point. Additionally, we convert the reconstructed point cloud into a 3D mesh for wider applications. To reduce ambiguity, we incorporate hidden lines in sketches. Experimental results demonstrate that our proposed approach significantly outperforms other baseline methods.**

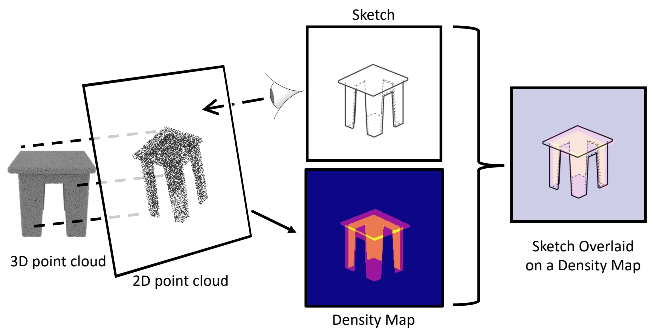


Fig. 1. The motivation of our work. We can see: 1) a 2D point cloud can be generated by projecting a 3D shape onto an image plane. On the projection plane, some locations have more than one 2D point with different depth values. 2) the distribution of 2D points can be characterized by a density map, where the value at each location indicates the probability of points projected at that location. 'Red' color indicates higher density. 3) the density map is spatially rough-aligned with the sketch.

I. Introduction

SKETCH is an intuitive form for individuals to express their ideas and has long been employed in 3D modeling. With the rapid advancements in deep learning and eXtended Reality (XR) techniques, sketch-based 3D modeling has garnered increasing interest as a form of User-Generated Content (UGC). This task holds immense potential in various domains, such as design, animation, and entertainment, captivating both academic researchers and industry professionals [1].

In recent years, significant strides have been made in the field of sketch-based 3D modeling [2]–[5]. Inspired by the success of image-based single-view 3D reconstruction (SVR) [6]–[8], most sketch-based 3D modeling approaches adopt a well-established SVR pipeline [2]–[4]. This pipeline involves encoding a sketch into a feature vector using a convolutional neural network (CNN) and utilizing 3D decoders to generate 3D coordinates that define the 3D shape. Generating fine details of a 3D shape from global features poses a challenge due to the substantial domain gap between a sketch and a 3D shape. To enhance the fine details of the generated 3D shape, some pipelines [5], [9], [10] in SVR and sketch-based 3D reconstruction extract local features to recover the fine details in the generated 3D shape. However, these approaches have not effectively addressed the inherent ambiguity when inferring local shapes from sketches. For instance, blank areas within a sketch may correspond to 1) a simple surface, 2) a surface obscured by objects behind it, or 3) the absence of any specific 3D shape.

Figure 1 illustrates a 2D point cloud projected from a 3D shape. Note that on the projection plane, there could be more than one 2D projected point with different depth values at the same location due to occlusions. The distribution of 2D projected points can be characterized by a density map, indicating the probability of points projected at each location. In other words, with a density map, we can infer the corresponding 2D point cloud. Considering that both the sketch and density map are 2D images, their domain gap is supposed to be smaller than that between sketch and 3D shape. This motivates us to introduce the density map as a proxy to facilitate sketch-to-3D reconstruction. Namely, given an input sketch, the reconstruction model first predicts a density map to recover the 2D projected points and then predicts the depth value for each 2D point. Finally, we employ an inverse-projection function to map the 2D projected points and their depths into a 3D point cloud. The adoption of the density map in our

C. Gao, X. Wang, Q. Yu, L. Sheng and J. Zhang are with School of Software, Beihang University, Beijing, China (email: gaochenjian@buaa.edu.cn, wang_xilin@buaa.edu.cn, qianyu@buaa.edu.cn, lsheng@buaa.edu.cn, zhang_jing@buaa.edu.cn).

X. Han is with School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, China (email: hanxiaoguang@cuhk.edu.cn).

Y.-Z. Song is with SketchX, CVSSP, University of Surrey, UK (e-mail: y.song@surrey.ac.uk).

D. Xu is with Department of Computer Science, The University of Hong Kong, Hong Kong, China. (email: dongxu@cs.hku.hk)

Code is available at https://github.com/cjeen/sketchsampler

**Co-first authors: Chenjian Gao and Xilin Wang*

†*Corresponding author: Qian Yu*

approach offers two significant advantages: 1) It alleviates the ambiguity in understanding blank areas of sketches during 3D reconstruction. These blank areas in sketches often pose challenges for accurate 3D modeling as they may represent varying geometric or spatial information. Utilizing a density map allows for a more effective interpretation of these regions, thereby reducing uncertainties in interpretation. 2) The density map serves as a regular, rasterized representation. Compared to the direct prediction of irregular point cloud representations, the density map offers a more structured and regular data format, which is more conducive for traditional convolutions. Point clouds, being irregular and sparse data representations, pose challenges for traditional convolutions due to difficulties in capturing the spatial relations and structural features of the point cloud data. In contrast, the grid-like regularity of density maps provides a more manageable and analyzable data structure, enhancing prediction accuracy and efficiency.

Although introducing the density map can reduce the domain gap between 2D sketches and 3D shapes, the ambiguity nature of sketches still poses a challenge. We observe that artists often utilize hidden lines to imply the structure of objects [11], [12]. The presence of hidden lines in sketches can suggest the contours and geometric features of objects through extensions, variations, and partial occlusions. These lines provide indirect but valuable clues for understanding sketches, helping us infer the structure of occluded regions. To differentiate the information expressed by hidden lines, we adopt a tri-channel sketch representation for the input to our 3D reconstruction network. Furthermore, for sketches that do not include hidden lines, we additionally train a neural network to predict a tri-channel sketch representation that incorporates hidden lines.

In this work, we present a new method for sketch-based SVR. We employ a sketch translator and a point cloud generator to produce point clouds. The sketch translator adopts a CNN-based encoder-decoder network, where the encoder network extracts features from the input sketch and the decoder network infers target 3D information, outputting a more informative 2D representation. Based on the output of the sketch translator, our point cloud generator aims to reconstruct a point cloud of the corresponding 3D shape. It first predicts the density map, which can be used as guidance to recover 2D point clouds, and then samples along a ray determined by each 2D projected point to predict depth values, where the point with farther depth values means it is occluded by the point with nearer depth values.

Considering that mesh representation has a broader range of practical applications, such as in physics simulation and ray tracing calculations, we utilize Poisson surface reconstruction to recover meshes from point clouds. To achieve improved mesh reconstruction results, we further enhance the point cloud. An additional sketch translator and point cloud enhancement network collaborate to upsample the point cloud and predict normals for each point. This process results in an enhanced point cloud with normals. Subsequently, this enhanced point cloud is utilized as the initial value in the Poisson equation solver to reconstruct the surface and generate the 3D mesh representation.

To demonstrate the effectiveness of our proposed model, we train and test it on a newly rendered dataset, *Synthetic-LineDrawing*. It is worth noting that a sketch may exhibit different levels of deformation and abstraction. Here we focus on sketches with reliable shape and fine-grained details, i.e., sketches with significant deformation or only expressing conceptual ideas are not considered in this work.

The contributions of this work can be summarized as follows:

- First, we present a novel method for sketch-based single-view 3D reconstruction, in which a 3D shape is recovered in two easier but indispensable steps, sketch translation and point cloud generation. A sketch-aware mesh generator is further introduced to convert a 3D point cloud into a mesh.
- Second, we formulate the point generation process as a two-stage probabilistic sampling process, where the density map is introduced as guidance.
- Third, we explore the potentials of hidden lines in reducing the ambiguity of sketches. To the best of our knowledge, this is the first time hidden lines are utilized for sketch-based 3D reconstruction.
- Fourth, we conduct extensive experiments to demonstrate the effectiveness of the proposed model on both synthetic and hand-drawn sketch datasets.

A preliminary version of this work has been published in ECCV 2022 [13]. Compared to the conference version, we have made several improvements. *First*, we incorporate hidden lines into sketches, commonly used in industrial scenarios but not yet fully explored in academia, to address sketch sparsity and ambiguity issues. We will show that utilizing hidden lines can significantly improve the reconstruction quality. *Second*, we extend the ability of SketchSampler to generate a 3D mesh, allowing for a broader range of practical usage such as physics simulation and ray tracing calculations. A novel sketch-aware point cloud enhancement network is proposed for mesh generation. *Third*, we compared more baselines, including [5], [10], and conducted additional experiments concerning robustness to provide more discussions and insights about this task.

II. Related Works

A. Single-view 3D Reconstruction (SVR)

3D reconstruction is a problem that has been widely studied in computer vision. Reconstructing a 3D shape from a single image is an ill-posed problem that requires strong prior knowledge. In recent years, with the development of deep learning, neural networks can be used to extract useful features for 3D reconstruction [14]–[17]. The early works focus on reconstructing 3D shapes represented by regular voxels [18]–[20]. This regular grid allows for a direct transposition of 2D convolution operations into 3D voxel space, making it cost-effective in terms of network design. However, the computational and memory cost of 3D voxels scales cubically, restricting their resolution and limiting their ability to capture fine-grained 3D structures and need further detailization [21]. Sparse 3D voxels were introduced to overcome the limitations of regular voxel grids [22]–[24]. Approaches like octree [22] and hash representations [25] use specialized data structures to improve storage and computational efficiency, enabling higher resolutions.

MarrNet [19], 3DensiNet [20] and ZeroShape [26] resorted to intermediate representations (i.e., 2.5D sketches, density heat-map and projection map) to facilitate reconstruction. In this work, we introduce the density map as a proxy, which reflects the probability of points projected at each location of the image plane. Unlike MarrNet and 3DensiNet, which directly reconstruct 3D shapes from intermediate representations, and ZeroShape, which employs only the visible surface as its intermediate representation, our approach utilizes a density map to guide the sampling of 2D points. Subsequently, both visible and invisible depth values are predicted from these sampled points.

Point clouds are another common representation for 3D shape generation [17], [27], [28]. They offer spatially adaptive representation without requiring additional data structures. However, point clouds do not inherently capture the surface-based representation of 3D shapes, which limits their applicability in scenarios such as physics simulations and ray tracing. Mesh is the most common 3D representation in computer graphics and is also used for 3D generation [29]–[31]. However, mesh structures are inherently graph-based, and deep neural networks still face various limitations when it comes to generating complex graph structures. These methods are only capable of generating 3D objects with a genus of 0, restricting their topological complexity.

Recently, implicit 3D representations utilize coordinate conditioned Multi-Layer Perceptrons (MLPs) to describe 3D shapes [32]–[34]. They define implicit 3D fields that correspond to the 3D shape, mapping 3D coordinates to values. For instance, OccNet [33] employs MLPs to map 3D coordinates to occupancy values, while DeepSDF [34] maps 3D coordinates to signed distance function (SDF) values. [9], [33], [35] explore 3D reconstruction based on implicit surface learning. These methods employ the marching cubes algorithm to extract isosurfaces from the given field during the inference. Implicit representations offer enhanced 3D representation capabilities but often require carefully designed regularization losses to avoid artifacts.

While SVR based on natural images has been extensively explored and has recently achieved remarkable results [36], [37], 3D reconstruction based on sketches has been explored to a lesser extent. In comparison to natural images, sketches are colorless and lack details, and most clues for depth prediction in photos are not available in sketches. Therefore, directly adapting SVR from natural images to sketches does not yield good results.

B. Sketch-based Shape Reconstruction and Generation

Sketch-based modeling is a problem that has been studied for a long time. The earlier methods extracted local geometric properties from hand-crafted rules and then inferred the 3D shape from the geometric properties [38], [39]. In recent years, some deep learning based methods have been proposed for sketch-based 3D modeling. Wang *et al* introduced a method to reconstruct 3D shapes based on retrieval [40]. The work [2] proposed to generate point clouds from a single hand-drawn image. They enhanced the PSGN [6] method with a viewpoint estimation module. To alleviate the deformation of sketches, they correspondingly proposed a sketch standardization module. The work [39] discussed the additional challenges of line drawings in comparison with images in 3D reconstruction. In [38], sketches from two viewpoints were used as the input to perform 3D reconstruction. Sketch2model [3] alleviated the ambiguity in sketch modeling by decoupling view code and shape code. Sketch2mesh [4] used an encoder/decoder architecture to learn a latent representation of an input sketch and refined it by matching the external contours of the reconstructed 3D mesh to the sketch during the inference process. While achieving good performance, this approach is time-consuming. Most deep sketch modeling methods encode a sketch as a latent code and then apply a decoder to convert the latent code to a 3D shape. However, these approaches fail to preserve spatial details in a sketch. LAS-Diffusion [5] utilized a conditional diffusion model to generate a signed distance field from the sketch. While this approach employs local features of the sketch as controlling conditions, it overlooks the issue of ambiguity caused by blank areas in the sketch and relies solely on the priors of 3D model distribution to address occlusion problems. Sketch-A-Shape [41] employs large pre-trained models to extract features from sketches, achieving zero-shot sketch-to-3D shape generation. However, the level of detail in the generated 3D objects remains inadequate.

In the field of sketch-based 3D reconstruction, the exploration of hidden lines has been limited. In CAD modeling [42], [43], sketches with hidden lines are often used to convey the shape of occluded parts, thereby reducing ambiguity. However, these works do not specifically differentiate between hidden and visible lines. To address this, we have adopted a tri-channel sketch representation, which aids neural networks in distinguishing between occluded shapes and visible shapes. Our approach effectively mitigates the ambiguity of occluded areas in sketches.

III. Methodology

As shown in Fig. 2, our sketch-based modeling framework mainly consists of two stages: the point cloud generation stage (Sec. III-A) and the mesh generation stage (Sec. III-B). During the point cloud generation stage, we first predict a 2D point cloud based on the sketch and then perform depth sampling. In the mesh generation stage, we employ a point cloud enhancement network to upscale and predict normals for the point cloud, leveraging sketch features and the enhanced point cloud is utilized for the Poisson surface reconstruction. In the end of this section (Sec. III-C), we propose to utilize hidden lines to address the sparsity and ambiguity problems of sketches.

A. Point Cloud Generation

Given an input sketch I, the sketch translator T first translates it to a feature map F. Next, the point cloud generator produces a point cloud $\mathcal{S}$ based on the given feature map F.

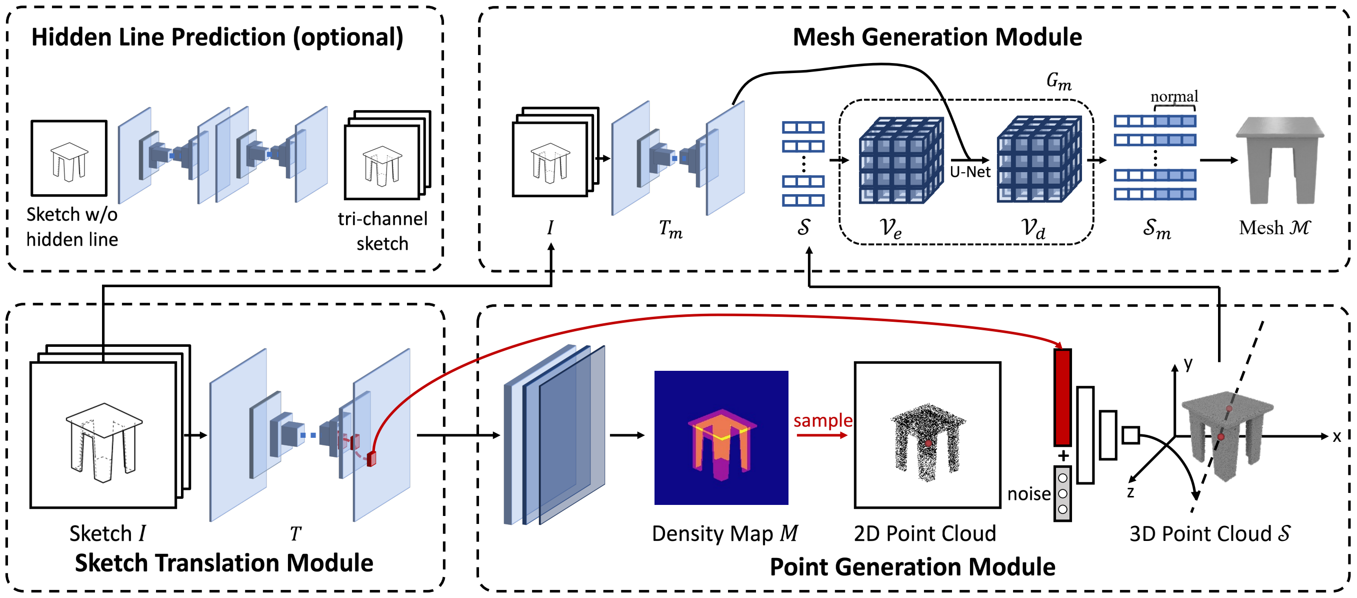


Fig. 2. Our method consists of two stages: the **point cloud generation stage** and the **mesh generation stage**. 1) In the point cloud generation stage, the input sketch is first translated to feature maps. Subsequently, a density map is predicted, from which projection points are sampled. These projection points are then used to sample corresponding features from the feature maps, enabling the prediction of depth values. 2) In the mesh generation stage, we upsample the point cloud and predict the corresponding normals. The sketch features are incorporated into the intermediate features of the 3D Unet as additional information.

When recovering the point cloud, a 2D density map is first predicted, from which 2D points are sampled; then the depth of each 2D point is predicted by using the proposed conditional depth generator. Note that in line with [4], [44], we adopt the commonly used "viewer-centered" setting [45], in which we assume the image space and the 3D space are aligned.

1) Sketch Translation: The goal of the sketch translator T is to fully exploit the spatial information in a sketch and generate suitable features for 3D shape prediction. It is non-trivial because there is a large information discrepancy between a sketch and a 3D shape: 1) a sketch is sparse and mainly preserves structural framework of a corresponding 3D shape. The object surface, occluded object surface, and vacancy between surfaces can all be shown as blank in a sketch. 2) most depth information in a 3D shape is also lost in a sketch image. Therefore, the sketch translator aims to complement the missing information. For example, inferring whether a blank area belongs to an object, or which pixels are on the same surface.

Specifically, we adopt an encoder-decoder based CNN network for sketch translation. Firstly, an encoder network is used to extract features from the input sketch with multiple down-sampling blocks. This is to increase the receptive field of the neurons to acquire an overview of the input sketch. A decoder network consisting of multiple upsampling blocks is then used to gradually infer the information of the 3D shape with increased spatial resolution. Instead of using the last feature map F^n for point cloud generation, we use a similar idea [46] that leverages the feature maps at all scales by upsampling the feature map at each individual scale F^i to the size of F^n and concatenating them together to produce the final feature F.

After sketch translation, the response of the feature map F is much denser than the input sketch while the spatial alignment and resolution are roughly maintained. It will facilitate the prediction of point clouds of with fine details, which will be explained below.

The point cloud generator aims to recover the point cloud of the corresponding 3D shape $\mathcal{S}$ from the translated feature maps F. To utilize the spatial information of the input sketch, we decompose the point could generation process into two steps: 1) predicting the 2D point cloud which is the projection of the 3D point cloud into the image plane; 2) inferring the depth of each 2D point.

For generating a 2D point cloud, the point cloud generator predicts the joint distribution of the coordinates from the projected points $p(X, Y|I)$, where X, Y are random variables corresponding to the x, y axis respectively. Sampling from $P(X, Y|I)$ will generate the 2D point cloud. Figure 1 shows an example of projecting a 3D shape into the image plane and its corresponding density map. The probabilistic density at each location varies because it depends on how many surfaces are being passed by the ray centered at this location.

After a 2D point cloud is generated, the point cloud generator predicts the depth distribution of each point $p(Z_i|x_i, y_i, I)$, where x_i, y_i is the location of the i-th point in the image plane. Sampling from $P(Z_i|x_i, y_i, I)$ gives the depth of each point. Combining x, y coordinates from density map sampling and z coordinate from depth sampling, the overall 3D point cloud can be generated.

From a probabilistic view, this process actually models the shape of a 3D object as a joint distribution of x, y, z coordinates. Our generation process assumes a factorization process over projection and conditional independency between different locations for depth prediction, i.e., $P(X, Y, Z|I) = P(X, Y|I)P(Z|X, Y, I)$. The first term and the second respectively correspond to the process of generating a 2D point cloud and the process of predicting depth given a 2D point cloud and a sketch.

2) 2D Point Cloud Generation.: As all valid locations must lay inside the image, we firstly model the distribution of projected points in pixel coordinates. The image coordinates can be seen as quantizing the x, y location into $\mathcal{W} \times \mathcal{H}$ bins, where $P_{u,v} = P(X = u, Y = v)$ is the probability of a projected point inside the (u, v)-th bin.

We use a mask prediction head to directly predict the density map $M \in \mathcal{R}^{\mathcal{W}\times\mathcal{H}}$, where $M_{v,u} = P_{u,v}$. It takes the translated sketch feature F as the input, and resizes the feature map to the size of $\mathcal{W} \times \mathcal{H}$ by using bilinear interpolation. The interpolated feature map is then passed to three convolutional layers for density prediction. The hyper parameters $\mathcal{W}$ and $\mathcal{H}$ control the resolution of the point clouds.

To generate a 2D point cloud, we can see $P(X, Y)$ as a multinomial distribution over $\mathcal{W} \times \mathcal{H}$ locations. We firstly sample a specific number of locations with the probabilities defined by the density map M, and then use the column and row indices u, v as x, y coordinates. We normalize the coordinate to the range of $[-1, 1]$ [1] to produce the coordinate in the image plane (x^I, y^I). It then converts the points to world coordinates by using the camera parameters. As we use the orthogonal projection model to produce the rendered sketches, the x, y coordinates are linearly mapped from that of the 3D point clouds. That is the x, y coordinates of a point in the raw 3D point cloud, and (x, y) can be computed as $(x^I/s, y^I/s)$, where s is a preset parameter of the projection model. Similar mapping functions can be drawn for other projection models.

3) Conditional Depth Estimation.: After producing the 2D coordinates (x, y) of the 3D points, the next step is to predict their z coordinates. Given its x, y location, we assume estimating the depth for each individual point to be independent, so we predict the conditional depth distribution $P(Z_i|x_i, y_i)$ separately for each 2D location. Given a x, y location, the depth distribution $P(Z_i|x_i, y_i)$ can be multimodal and the number of modes tends to be varied, as there may be one or multiple points from different surfaces sharing the same 2D location. It is hard to explicitly define the probabilistic function of $P(Z_i|x_i, y_i)$.

Inspired by the Generative Adversarial Networks [47]–[49], we use an implicit approach and adopt the generator network design to model $P(Z_i|x_i, y_i)$. It takes a noise variable $N \in R^d$ and the local feature $f_{x,y}$ as input, and predict a scalar of depth z, where N is sampled from the uniform distribution $\mathcal{U}(0, 1)$ and $f_{x,y}$ is obtained by extracting from the feature map F at the corresponding location. Note that the depth generator can output different depth values given the same feature and different noise variables. It takes a multi-layer perceptron (MLP) as the backbone and its parameters are shared at all (x_i, y_i) locations.

For inference, we randomly sample a noise vector $\mathbf{n}_i$ by following the uniform distribution for each point (x_i, y_i) in the predicted 2D point cloud, and then predict the corresponding z_i. Putting the 2D location (x_i, y_i) and depth prediction z_i together will generate the final point cloud $\mathcal{S}$. Note that the sampled random noise $\mathbf{n}_i$ controls which mode the predicted depth z_i falls in if the corresponding $P(Z_i|x_i, y_i, I)$ is multimodal. Together with 2D point cloud sampling, the two-stage process can be seen as sampling from the joint distribution that defines the coordinates of a 3D shape. The detailed process of point cloud generation is listed in Alg. 1. Note that our proposed method is compatible with both orthogonal projection and perspective projection. It can be controlled by the 'invproj' function in Alg. 1.

Algorithm 1 Point Cloud Generation Process

```
Input: total number of points $N$, the predicted density map $M$, feature map $F$, and depth generator $T_d$.
Output: the reconstructed point clouds of the sketch $\mathcal{S}$.
1:  Let $\mathcal{S} = \emptyset$
2:  while $|\mathcal{S}| \leq N$ do
3:      sample a location from the multinomial distribution defined by $M$, i.e. $(u, v) \sim Mult(x, y; M)$.
4:      convert $u, v$ to the image plane coordinate $x^I, y^I$
5:      sample the noise vector $\mathbf{n} \sim \mathcal{U}(0, 1)$.
6:      inference the depth at $u, v$: $z_c = T_d(\mathbf{n}, F_{uv})$
7:      convert $(x^I, y^I, z_c)$ to the world coordinate: $(x, y, z) = \text{invproj}(x^I, y^I, z_c)$
8:      $\mathcal{S} = \mathcal{S} \cup \{(x, y, z)\}$
9:  end while
10: return $\mathcal{S}$
```

4) Loss Function: A key role in our proposed approach is the density map. Fortunately, we can freely produce the ground-truth density map from a 3D shape by a customized renderer, i.e., counting the number of points that occurred when projecting a ray from a 3D point onto an image plane followed by normalization. To provide supervision information for the learning process of the density map, we use the L1 loss as a constraint, as shown in Eq. (1).

$$L_D = \sum_{x_i, y_i} \|\hat{p}(x_i, y_i) - p(x_i, y_i)\|_1. \tag{1}$$

To provide supervision information for the learning process of the conditional generator, we constrain the distance between the output point cloud and the ground-truth point cloud. We use the Chamfer distance as the loss function during the training process. Given two point clouds $\mathcal{S}, \hat{\mathcal{S}} \subseteq \mathcal{R}^3$, the Chamfer distance is defined as Eq. (2). The final loss function is shown in Eq. (3), and λ_1 and λ_2 are the weights of L_{CD} and L_D, respectively.

$$L_{CD} = \frac{1}{|\mathcal{S}|} \sum_{p \in \mathcal{S}} \min_{q \in \hat{\mathcal{S}}} \|p - q\|_2^2 + \frac{1}{|\hat{\mathcal{S}}|} \sum_{q \in \hat{\mathcal{S}}} \min_{p \in \mathcal{S}} \|q - p\|_2^2 \tag{2}$$

$$L = \lambda_1 L_{CD} + \lambda_2 L_D, \tag{3}$$

As shown in Fig. 2, during the training process, the feature maps from the encoder-decoder network are fed into two paths: 1) the convolutional layers to predict the density map; 2) the fully-connected layers to predict the depth value. Correspondingly, the L1 loss in Eq. (1) and the Chamfer loss in Eq. (2)

[1] $x = \frac{2u}{\mathcal{W}-1} - 1$, $y = \frac{2v}{\mathcal{W}-1} - 1$, where $u = 0, 1, ..., \mathcal{W} - 1$, $v = 0, 1, ..., \mathcal{H} - 1$

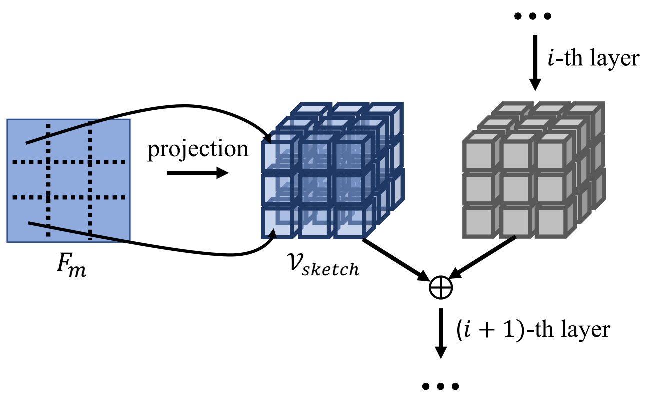


Fig. 3. We incorporate sketch features into the middle features of the 3D UNet decoder. According to the projection function, we initially transform the sketch feature F_m into voxel feature $\mathcal{V}_{sketch}$. Subsequently, we concatenate it with the feature output from the i-th layer of the UNet. The resulting fused features are then passed as input to the (i+1)-th layer of the UNet.

are computed, and the gradients from these two losses will be separately backpropagated along two different paths back to the encoder-decoder network.

B. Mesh Generation

The goal of mesh generation is to generate a mesh $\mathcal{M}$ that corresponds to the sketch I and the predicted point cloud S. In our method, we adopt the paradigm of Poisson surface reconstruction, which is a classic method for reconstructing surfaces from point clouds. The basic idea of the Poisson surface reconstruction algorithm is to compute surface normals based on the sampling information of the discrete point cloud and use these normals for surface reconstruction. Our surface reconstruction method consists of three steps:

Step-1: Translate the input sketch to the feature map F_m using T_m, which has a similar structure with T.

Step-2: Predict an upsampled point cloud $\mathcal{S}_m$ with normals based on the feature map F_m and the point cloud $\mathcal{S}$. This step is to enhance the point cloud $\mathcal{S}$ obtained in Sec. III-A.

Step-3: Solve the Poisson equation with $\mathcal{S}_m$ as the initial value to reconstruct the mesh surface $\mathcal{M}$.

1) Point Cloud Enhancement: To predict the upsampled point cloud $\mathcal{S}_m$ with normals, the point cloud enhancement network G_m adopts an encoder-decoder network similar to [50], with the difference that we incorporate the sketch feature F_m.

The encoder E_m of the point cloud enhancement network takes F_m and $\mathcal{S}$ as inputs and outputs a voxelized feature $\mathcal{V}_e$. The encoder first extracts features from the point cloud $\mathcal{S}$ using a per-point multi-layer perceptron (MLP) [51]. To incorporate information exchange between points, the encoder employs local point set feature pooling between layers [50]. Then, the encoder voxelizes the feature-extracted point cloud $\mathcal{S}$ to obtain a voxelized feature representation, where each voxel block encapsulates the features of a local point cloud. Next, the encoder applies a 3D convolutional U-Net network [52] to obtain the final voxelized feature $\mathcal{V}_d$. The network consists of a series of down-sampling and up-sampling convolutions as well as skip connections to integrate local and global information. To insert the information from F_m into the encoded features, we voxelize the sketch features. The voxelized sketch features $\mathcal{V}_s$ at the position (x, y, z) are defined as $F_m^{(u,v)}$, where $(u, v) = \pi(x, y, z; c)$ represents the 2D coordinates of the 3D coordinates (x, y, z) projected onto the sketch image under the camera pose c, and $F_m^{(u,v)}$ is the feature obtained by bilinearly interpolating the feature map F_m at (u, v). This process is illustrated in Fig. 3. Then, we concatenate the voxelized sketch features with the intermediate features of the 3D U-Net network in the feature dimension. Since the intermediate features in the U-Net encoder are connected to the U-Net decoder through skip connections, it is sufficient to fuse the voxelized sketch features into the voxelized features on the decoder side.

The decoder D_m of the point cloud enhancement network takes $\mathcal{S}$ and the voxelized features $\mathcal{V}_d$ as inputs, performs upsampling for $\mathcal{S}$, and predicts the normals to obtain $\mathcal{S}_m$. First, the decoder uses bilinear interpolation on $\mathcal{V}_e$ with the coordinates of each point in $\mathcal{S}$ to obtain the features of each point. Then, it alternates between applying multi-layer perceptrons (MLPs) and average pooling on the feature-enriched point cloud to process the features of the point cloud. The final output point cloud has feature dimensions of $N \times (3+3)$, which are interpreted as the offsets and normal directions for N (upsampling rate) points.

2) Loss Function: To supervise the point cloud enhancement network, we adopt the differentiable Poisson surface reconstruction (DPSR) [53]. Poisson surface reconstruction is achieved by solving the Poisson equation, and DPSR is a differentiable algorithm for solving the Poisson equation. The Poisson equation arises from the observation that an implicit indicator function χ can describe the geometry of a solid, and the zero-level set of χ corresponds to the surface of the solid. A set composed of point coordinates and normal vectors can be regarded as a sampling of the gradient of χ. The Poisson equation belongs to the partial differential equations (PDEs) and is defined as Eq. (4):

$$\nabla^2 \chi = \nabla \cdot \nabla \chi = \nabla \cdot \mathbf{v}. \quad (4)$$

Here, $\mathbf{v}$ is a vector field defined by the normals of the point cloud. Given the predicted point cloud $\mathcal{S}_m$ with normals, we can use DPSR to obtain the predicted implicit indicator function $\hat{\chi}$, and then compute the mean square error between χ and the ground truth χ as the loss function, as shown in Eq. (5):

$$L_{DPSR} = \|\chi - \hat{\chi}\|_2. \quad (5)$$

During model inference, we utilize the marching cubes algorithm [54] to extract the isosurface with a value of 0 from $\hat{\chi}$ as the predicted mesh $\mathcal{M}$.

C. Enhanced Sketch Input with Hidden Lines

Hidden lines refer to the lines indicating the parts of an object that are obscured from the given view. Hidden lines are commonly employed in industrial design scenarios, such as

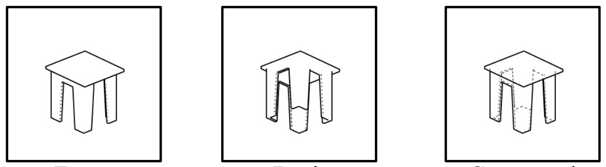


Fig. 4. Our used tri-channel sketch representation. Solid lines are used in the front sketch and back sketch, while in the composed sketch, the back view is represented using thin dashed lines.

CAD drafting [42], [43]. We assume that the sketch is made with hidden lines distinctly marked.

Hidden lines can be used to eliminate ambiguity in cases where parts of a 3D object are obscured when viewed from a specific angle. As shown in Fig. 4, if only the front view sketch is available, the table could either have three legs or four legs, and the style of the table legs is also uncertain. In contrast, in the *composed* sketch where hidden lines are included, as shown in Fig. 4, such ambiguity can be significantly reduced.

We use a tri-channel representation to depict sketches with hidden lines, as illustrated in Fig. 4. In addition to the front view, we include another two channels: the back view and the composed sketch. For the user input sketch, we represent the visible lines as the front view. The back view is a combination of the hidden lines and the outer contours of the visible lines. The composed sketch contains both the visible lines (represented as solid lines) and the hidden lines (represented as fine dashed lines). In the experimental section, we compared the performance of different representations.

To address the situation in which only the front view sketch is available, we propose an adaptation strategy. We utilize the network [49] to generate a tri-channel representation (front view + back view + composed) based on the front view sketch. In our experiments, we observed that using either L2 or L1 loss alone to guide the network's training led to blurry lines and the omission of back view lines. Consequently, we adopted a two-stage approach to accomplish this task.

In the first stage, the input is the front view sketch, and the training supervision is applying L2 loss to the predicted tri-channel sketch. In the second stage, the input is the prediction obtained from the first stage, and the training supervision is guided by L1 loss for the predicted tri-channel sketch. Through our experiments, we found that this two-stage prediction method effectively generates a tri-channel representation from the front view sketch. We present these results in detail in the experimental section, showcasing the effectiveness of our approach.

IV. Experiment

A. Implementation Details

We use a CNN-based encoder-decoder network [49] as our sketch translator. The encoder and decoder of T_{stage1} consist of three downsampling layers and three upsampling layers. There are nine residual blocks between the encoder and decoder. The network outputs a total of four feature maps with dimensions $(channels \times height \times width)$ as follows: $\{(64 \times 256 \times 256), (128 \times 128 \times 128), (256 \times 64 \times 64), (512 \times 32 \times 32)\}$.

The encoder and decoder of T_m consist of four downsampling layers and two upsampling layers. There are nine residual blocks between the encoder and decoder. The network outputs three feature maps with dimensions $(channels \times height \times width)$ as follows: $(128 \times 64 \times 64), (256 \times 32 \times 32), (512 \times 16 \times 16)$.

The density map prediction head contains three convolutional layers with 256, 64, and 1 filters, respectively. The kernel sizes are 1, 3, and 1. The depth sampler uses a 4-layer Multi-Layer Perceptron (MLP) with residual connections. The channel sizes are 64, 32, 16, and 1. During training, we used real density maps instead of predicted density maps to sample projection points.

The values of λ_1 and λ_2 are set to 1 and 10^4, respectively. The point cloud refinement network G_m has five basic blocks. The basic block in the encoder of G_m includes max pooling of local voxel blocks, while the basic block in the decoder does not. The resolution of voxels used for feature aggregation in the encoder of G_m is 32^3, and the voxelized point cloud features in G_m have a resolution of 64^3.

B. Datasets

There is a scarcity of publicly available large-scale paired sketch-3D datasets. Therefore, we utilized Blender's freestyle rendering feature [55] to generate sketch images from 3D models of 13 categories selected from the ShapeNet dataset [18]. This contributed to the creation of a new dataset called *Synthetic-LineDrawing*.

Previous benchmark methods were trained using different 3D representations such as point clouds, meshes, SDF, *etc.*, and employed varying object size and position normalization strategies. To establish consistency, we adopted the preprocessing approach of DISN [9] for 3D objects, aligning the coordinate system's origin with the object's centroid and scaling the object's size to fit within a unit sphere. DISN provided both the preprocessed SDF representation and the mesh representation of the 3D objects. Additionally, we sampled point clouds from the mesh surface.

For each object, we randomly sampled five viewpoints, following the viewpoint sampling strategy of 3D-R2N2 [18]. The azimuth angle was uniformly sampled from 0 to 360 degrees, while the elevation angle was uniformly sampled from 25 to 30 degrees. For each viewpoint, we rendered both a hand-drawn sketch and a projection density image. When rendering the projection density image, we utilized the 'DepthPeeler' provided by nvdiffrast [56], repeatedly rasterizing the 3D mesh and incrementing the pixel count for foreground pixels by 1 each time. Finally, we normalized the entire image so that the sum of pixel values equaled 1, obtaining the projection density image. Synthetic-LineDrawing comprises a total of $218,915$ sketch images and their corresponding $43,783$ 3D objects, spanning 13 categories. We followed the conventional training/test split as indicated in [18], with a ratio of $4/5$ for the training set and $1/5$ for the test set.

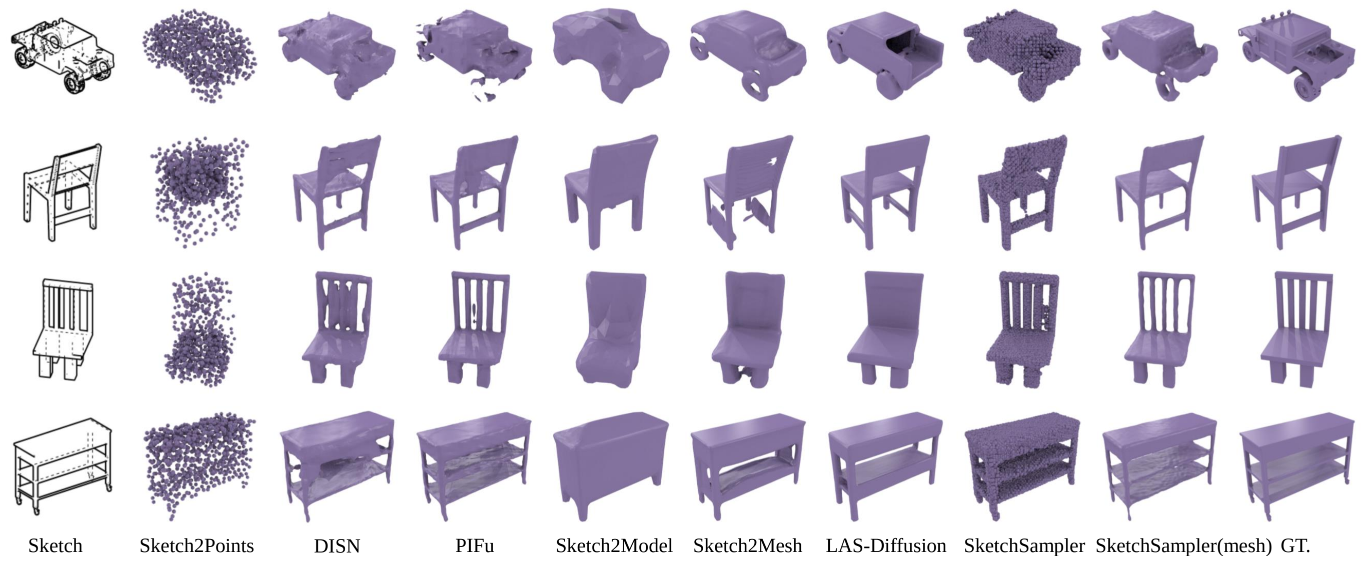


Fig. 5. Comparison between our method and baseline methods. **SketchSampler** represents the point cloud results generated in the first stage, and **SketchSampler(mesh)** represents the mesh results generated throughout the entire stage.

C. Training Details

Both stages of the model were trained for 30 epochs with an initial learning rate of 10^{-3} using a linear learning rate decay schedule, where the learning rate decreased gradually and reached 0 at the end of training. We employed the Adam optimizer [57] for optimization. The same training/test split was used for both the point cloud generation stage and the mesh generation stage of the network.

D. Evaluation Metrics

To evaluate the quality of 3D object generation, we sample point clouds from the surfaces of 3D objects and compare the distances between the ground truth and predicted point clouds. For two point clouds, denoted as $\mathcal{X}$ and $\mathcal{Y}$, we employ the following metrics:

Chamfer Distance (CD): The Chamfer Distance is commonly used to measure the distance between two point clouds and is widely employed in shape comparison. It measures the distance between each point in one point cloud and the nearest point in another point cloud. The formula for Chamfer Distance is defined as follows:

$$d_{\mathrm{CD}}(\mathcal{X},\mathcal{Y}) = \frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}}\min_{y\in\mathcal{Y}}\|x-y\|_2^2 + \frac{1}{|\mathcal{Y}|}\sum_{y\in\mathcal{Y}}\min_{x\in\mathcal{X}}\|y-x\|_2^2$$

Earth Mover's Distance (EMD): The Earth Mover's Distance is also used to evaluate the similarity between two point clouds. But it is more sensitive to the local details and density distribution. The EMD is defined as:

$$d_{\mathrm{EMD}}(\mathcal{X},\mathcal{Y}) = \frac{1}{|\mathcal{X}|}\min_{\phi:\mathcal{X}\to\mathcal{Y}}\sum_{x\in\mathcal{X}}\|x-\phi(x)\|_2$$

Voxel-IoU: Voxel-IoU applies the concept of Intersection over Union (IoU) to 3D space to evaluate the overlap between predicted voxel models and ground truth voxel models. It is computed as the ratio of the intersection volume to the union volume. For our case, we voxelized the point clouds and compared the Voxel-IoU between the two voxelized point clouds. A higher Voxel-IoU indicates a greater similarity between the two point clouds. The formula for Voxel-IoU is defined as follows:

$$\text{Voxel-IoU}(\mathcal{X},\mathcal{Y}) = \frac{\mathrm{intersection}(V(\mathcal{X}),V(\mathcal{Y}))}{\mathrm{union}(V(\mathcal{X}),V(\mathcal{Y}))}$$

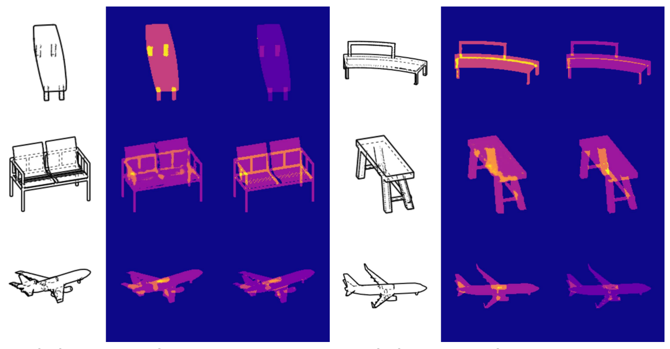


Fig. 6. Density maps generated by our method. Red indicates higher density, while blue indicates lower density.

E. Comparison with Existing Methods

We first compare our method with four state-of-the-art approaches for sketch-based single-view 3D reconstruction (SVR):

Sketch2Mesh [4]: This method takes a sketch as the input and predicts a 3D mesh using an encoder-decoder network [8]. It learns a compact feature representation and recovers the 3D shape by minimizing the 2D Chamfer distance between the projected contour of the 3D shape and the input sketch.

Sketch2Model [3]: This method aims to reconstruct 3D meshes from a single sketch. It utilizes an encoder-decoder

TABLE I
COMPARISON RESULTS ON THE SYNTHETIC-LINEDRAWING DATASET.

	Categories													mean
	airplane	bench	cabinet	car	chair	display	lamp	speaker	rifle	sofa	table	phone	boat	
Chamfer Distance(↓) $\times 10^{-3}$														
Sketch2Mesh [4]	3.785	17.741	8.191	3.029	10.554	9.540	56.554	19.223	2.145	6.139	8.931	7.208	9.621	12.512
Sketch2Model [3]	7.994	15.282	12.081	11.514	15.735	13.006	29.348	22.888	5.422	11.850	21.953	9.814	11.710	14.507
Sketch2Points [2]	18.250	86.600	16.486	9.816	22.276	42.373	30.079	34.625	11.853	42.044	22.067	23.648	6.287	28.185
LAS-Diffusion [5]	3.796	-	-	3.235	7.066	-	-	-	3.162	-	6.353	-	-	-
PIFu [10]	2.849	4.947	3.817	2.226	5.922	4.378	19.563	10.372	2.820	3.352	4.875	5.625	4.293	5.772
DISN [9]	3.359	9.100	4.838	2.417	8.326	5.832	25.688	12.291	3.243	4.330	8.267	5.507	5.945	7.626
Ours (SketchSampler, 5class)	1.997	-	-	1.839	3.324	-	-	-	1.436	-	**2.267**	-	-	-
Ours (SketchSampler(mesh), 5class)	1.977	-	-	2.218	4.678	-	-	-	1.544	-	3.647	-	-	-
Ours (SketchSampler)	1.848	**2.257**	**2.450**	**1.719**	**2.966**	**2.864**	**8.593**	**4.916**	**1.226**	**2.285**	2.460	**3.379**	**3.189**	**3.089**
Ours (SketchSampler(mesh))	**1.762**	3.559	3.521	2.295	4.656	3.352	11.725	8.312	1.366	2.921	3.945	3.392	3.862	4.205
Earth Mover's Distance(↓) $\times 10^{-2}$														
Sketch2Mesh [4]	6.988	11.161	8.935	6.074	10.502	9.235	23.819	12.116	5.566	7.298	9.094	6.773	8.083	9.665
Sketch2Model [3]	11.879	11.683	10.300	9.563	13.344	10.490	20.353	13.338	9.026	9.953	14.063	8.131	10.590	11.747
Sketch2Points [2]	16.084	30.889	20.044	13.065	18.682	25.555	28.954	23.178	15.613	21.871	18.087	21.458	11.718	20.400
LAS-Diffusion [5]	9.343	-	-	7.036	9.583	-	-	-	7.566	-	8.569	-	-	-
PIFu [10]	6.733	**7.969**	**6.810**	**5.362**	8.677	6.837	16.644	**10.174**	5.156	5.897	8.047	5.495	6.503	7.716
DISN [9]	6.813	10.208	7.371	5.668	10.700	8.059	19.356	10.820	5.678	6.544	10.528	5.631	7.588	8.843
Ours (SketchSampler, 5class)	8.093	-	-	8.621	10.610	-	-	-	5.063	-	8.580	-	-	-
Ours (SketchSampler(mesh), 5class)	5.793	-	-	5.613	8.035	-	-	-	4.867	-	6.911	-	-	-
Ours (SketchSampler)	7.717	8.283	8.489	7.953	10.607	8.651	17.574	11.356	4.655	10.018	8.385	6.134	9.346	9.167
Ours (SketchSampler(mesh))	**5.953**	8.064	7.124	5.823	**8.225**	**6.500**	**15.698**	10.229	**4.355**	**5.780**	**6.825**	**4.724**	**6.351**	**7.358**
Voxel-IOU(↑)														
Sketch2Mesh [4]	0.640	0.413	0.421	0.550	0.414	0.404	0.294	0.306	0.702	0.422	0.517	0.529	0.457	0.467
Sketch2Model [3]	0.310	0.191	0.199	0.224	0.210	0.228	0.220	0.161	0.376	0.211	0.151	0.302	0.264	0.234
Sketch2Points [2]	0.250	0.101	0.099	0.188	0.121	0.096	0.159	0.071	0.321	0.092	0.156	0.142	0.359	0.166
LAS-Diffusion [5]	0.533	-	-	0.475	0.413	-	-	-	0.612	-	0.496	-	-	-
PIFu [10]	0.679	0.561	0.559	0.606	0.522	0.497	0.469	0.428	0.741	0.532	0.600	0.581	0.548	0.563
DISN [9]	0.667	0.470	0.501	0.598	0.457	0.463	0.389	0.383	0.752	0.489	0.524	0.638	0.524	0.527
Ours (SketchSampler, 5class)	0.651	-	-	0.600	0.529	-	-	-	0.703	-	0.611	-	-	-
Ours (SketchSampler(mesh), 5class)	0.702	-	-	0.625	0.531	-	-	-	0.731	-	0.626	-	-	-
Ours (SketchSampler)	0.660	**0.595**	0.571	0.614	**0.549**	0.551	**0.485**	**0.462**	0.738	0.538	0.631	0.627	0.552	0.583
Ours (SketchSampler(mesh))	**0.706**	0.578	**0.587**	**0.630**	0.545	**0.556**	0.459	0.438	**0.768**	**0.553**	**0.643**	**0.675**	**0.562**	**0.592**

network [7] for mesh reconstruction and introduces an additional encoder-decoder to decompose the sketch features into viewpoint and shape spaces. During inference, each 3D shape is reconstructed based on the input sketch and given viewpoint.

Sketch2Points [2]: This method focuses on reconstructing 3D point clouds from sketches. It is based on PSGN [6]. It can only generate a very limited number of point clouds.

LAS-Diffusion [5]: This method propose a two-stage diffusion model including occupancy-diffusion stage and SDF-diffusion stage. This method use a novel view-aware local attention mechanism to leverage the local information in the sketch.

We also compare our method with two image-based SVR methods:

DISN [9]: This method proposes a signed distance field (SDF) predictor that utilizes both global and local features for prediction. It can generate detailed 3D shapes by leveraging local features sampled from the image feature map.

PIFu [10]: PIFu introduces pixel-aligned implicit functions, which can generate high-resolution surfaces including unobserved regions. The generated surfaces from PIFu are spatially aligned with the input image.

We follow their original works of Sketch2Model and Sketch2Mesh to train an individual model for each category. For LAS-Diffusion, we use their original settings, training a 5-class model for airplane, car, chair, rifle and table categories. Correspondingly, we separately trained our model on the same 5 categories as LAS-Diffusion for a direct comparison. For all methods, we trained the methods on Synthetic-LineDrawing dataset, and we used real viewpoints as inputs, except for Sketch2Points, as this method doesn't require viewpoint information as input. Furthermore, we sampled 8,192 points to calculate quantitative metrics for all methods, except for the Sketch2Points method, which generates only 1,024 points.

The visualizations of different methods are shown in Fig. 5. The point cloud generated by Sketch2Points has too few points, making it possible to discern only a rough category. DISN and PIFu utilize local features to reconstruct the SDF field representation, but the high degrees of freedom in the SDF representation result in many floating artifacts. For instance, noise-like structures appear on the backrest of a chair. Sketch2Model can only generate shapes with genus 0, and the surfaces are excessively smooth. Sharp transitions, such as the connection between the support surface and the backrest of a chair, are not well represented. In the refine stage of Sketch2Mesh, only the outer contour information of the sketch is utilized, which means that internal details are not faithfully reflected in the 3D shape. For example, hollow sections on the back of a chair or the number of partitions in a table cannot be effectively conveyed. LAS-diffusion employs a diffusion model to learn the distribution of 3D objects, resulting in higher visual quality. However, the information from the sketch is not faithfully represented in the 3D objects. Our method outperforms all competing methods. The reconstructed point clouds by SketchSampler have correct category labels and exhibit a significant level of detail, despite our model being trained only once and being category-agnostic. The overall layout of the point clouds aligns

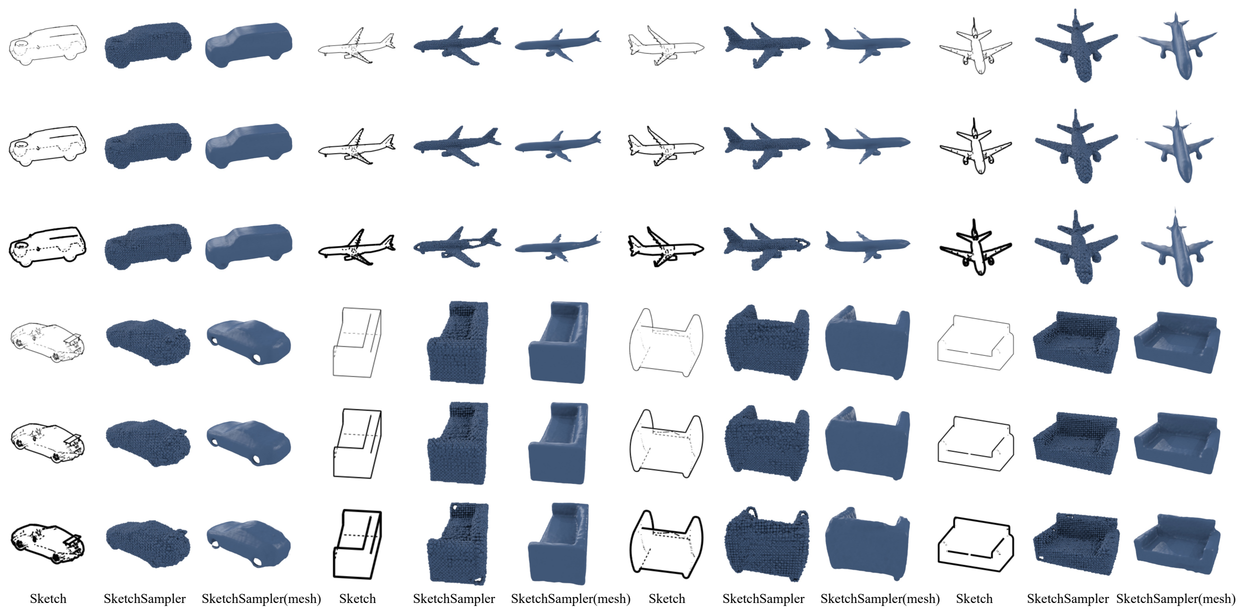


Fig. 7. Robustness on line width variations. The linewidth scales for model inference are **0.5** (top row), **1.0** (middle row), **2.0** (bottom row).

well with the input sketches. Point cloud enhancement further enhances the quality by eliminating artifacts. Even thin parts, such as chair back and table legs, are faithfully captured in our generated 3D shapes. This is because our approach employs normals based on point cloud data to describe surfaces, which introduces a certain degree of regularization, leading to results that are less prone to artifacts compared to DISN and PIFu.

The quantitative results are shown in Table I. Although the performance ranking of these models varies under different evaluation metrics, our method consistently outperforms others in terms of all metrics, on average across categories. Except for EMD, our model performs the best over all categories. It suggests that our reconstructed 3D shape captures both global structure and the local details. For the CD metric, the point cloud generated by SketchSampler is better than the surface-sampled point cloud from SketchSampler(mesh). However, for the EMD and IOU metrics, SketchSampler(mesh) generally performs better. This discrepancy can be attributed to the fact that SketchSampler employs CD as a direct learning objective and CD is not sensitive as EMD and IOU. Therefore, although the mesh surface-sampled point cloud from SketchSampler(mesh) has fewer outliers since it is constrained and has fewer degrees of freedom compared to SketchSampler, SketchSampler(mesh) is inferior to SketchSampler in terms of CD but superior in terms of EMD and IOU. Furthermore, we observed that our models trained on 13 categories outperform those trained on just 5 categories. This observation suggests that our model is capable of capturing more fundamental patterns from a broader range of category data, rather than just recognizing specific instances. Therefore, training on additional categories can enhance the testing performance for the 5-category scenario.

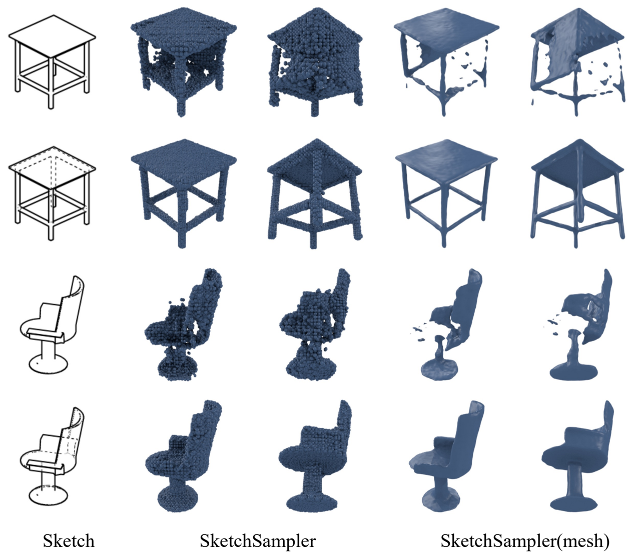


Fig. 8. Robustness on sketch without hidden lines. To facilitate the observation of the impact of hidden lines, we present two perspectives: front view and back view.

F. Robustness

Line Width Variation. To test the sensitivity of our method to sketch line width, we conducted evaluations on sketches with varying line widths. Based on Synthetic-Line drawing, we doubled and halved the line width of the sketches to create thinner and thicker versions, referred to as “thin” and “fat” sketches, respectively (as shown in Fig. 7). These altered line-width sketches were then fed into the Point Cloud Generation network and Mesh Generation network. The results of these tests are shown in Fig. 7. It can be observed that our method exhibits a certain degree of robustness to changes in line width. Increasing the line width tends to lead to a more pronounced

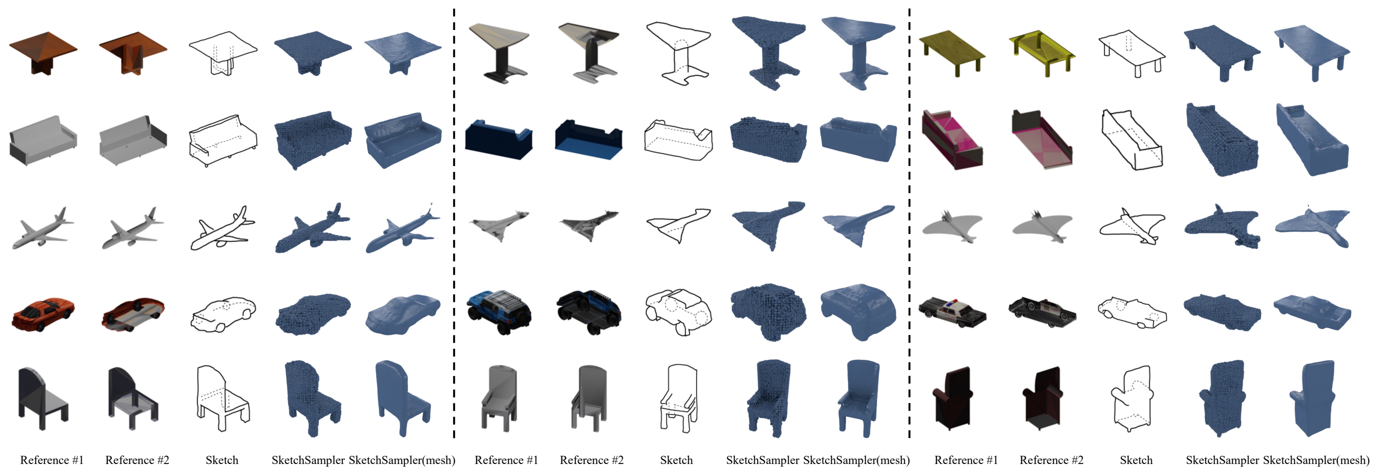


Fig. 9. Robustness on hand-drawn sketches. The model has not undergone any fine-tuning. Point cloud and mesh results are both generated by the model trained on Synthetic-LineDrawing.

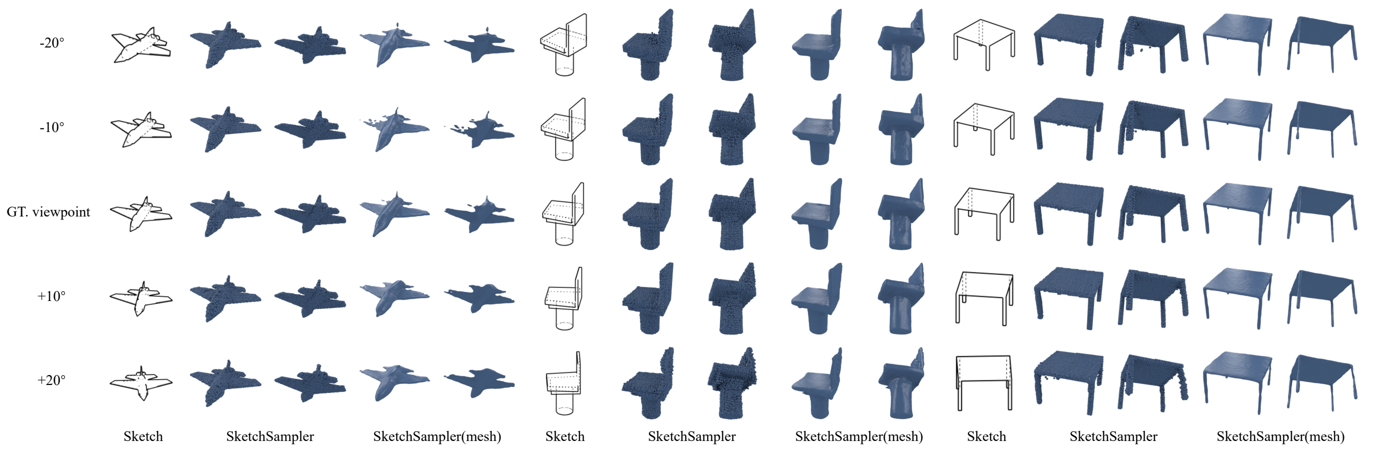


Fig. 10. Robustness on noise views. We use the corresponding camera pose to perturbed azimuth angle. The noisy pose is utilized as an input parameter for the projection and back-projection functions.

degradation in generation quality compared to decreasing the line width. For example, there are two planes and a sofa show some incompleteness in the reconstructed points when the line width is increased. Fortunately, due to the point cloud enhancement, these incompleteness are not reflected in the generated mesh.

Sketch without Hidden Lines. Then we evaluate the impact of hidden lines. To simulate the absence of hidden lines, we substituted the “composed” channel with the “front” channel and set all values in the “back” channel to white. The results of these tests are presented in Fig. 8. It is evident that the quality of the reconstructed shapes significantly deteriorates when hidden lines are removed. These results were expected since our model was trained on datasets that include hidden lines. Therefore, removing hidden lines during testing introduces a significant distribution shift, resulting in unsatisfactory performance. We further conducted an experiment to show the impact of hidden lines in Sec. IV-G.

Hand-drawn Sketches. Subsequently, we proceeded to evaluate the performance of our method on hand-drawn sketches. Given the current absence of datasets that combine hand-drawn sketches with hidden lines, we took it upon ourselves to manually curate a collection of hand-drawn sketches featuring hidden lines. For each object, we systematically sampled an azimuth angle ranging from 0 to 360 degrees and a elevation angle between 25 to 30 degrees to ensure diversity. Subsequently, we rendered corresponding front and back view images for each sketch, which were then presented to users as visual references. During the sketching process, users were granted the flexibility to toggle between different line styles, enabling them to distinguish between drawing the front and back views. Additionally, they could switch the displayed reference images between the front and back views at their drawing. The reference images, hand-drawn sketches, and the results generated by our method are shown in Fig. 9. Our method demonstrates a commendable ability to handle hand-drawn sketches, yielding remarkably accurate outcomes.

Noisy Viewpoints. Then we conducted evaluations of our method's performance under different noisy viewpoints. We introduced perturbations to the azimuth angles of the sketches in the Synthetic-LineDrawing dataset. The magnitudes of these perturbations were set to $\{-20°, -10°, +10°, +20°\}$. For each

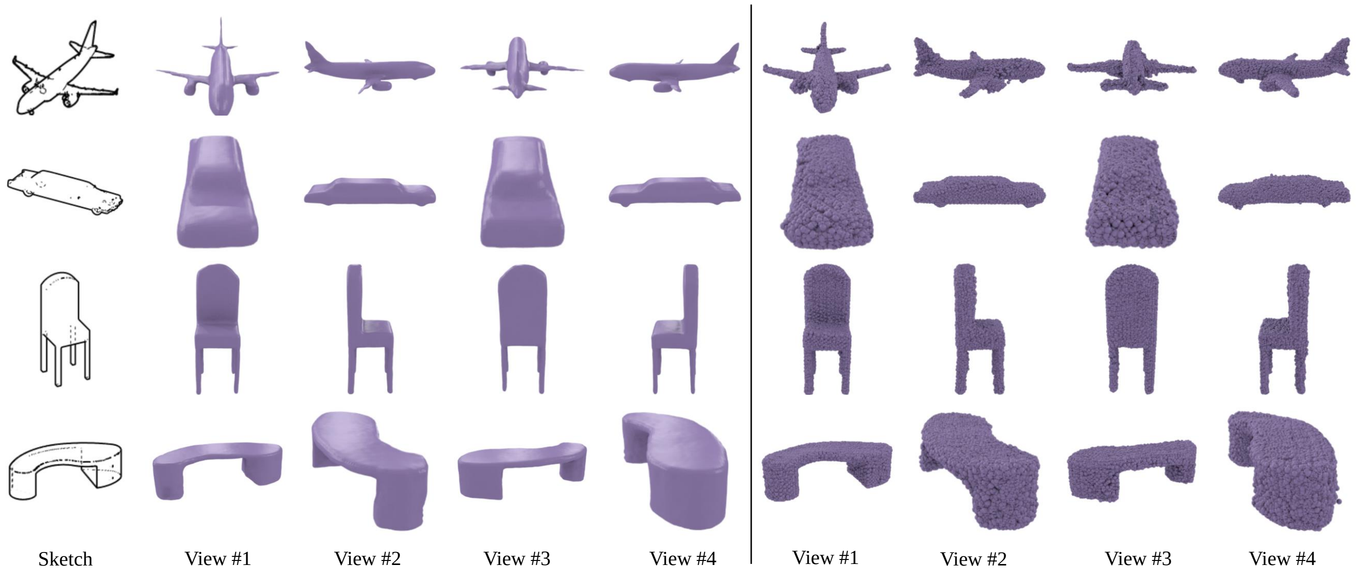


Fig. 11. Multi-view visualizations of meshes (left) and point clouds (right) generated by our method. View #1 to #4 correspond to azimuth angles of {0°, 45°, 90°, 270°} respectively. The elevation angle is set to 20°.

original input sketch sample, we generated four variations by applying these angle perturbations, resulting in a total of four different perspective-distorted input sketch samples for each original. Within our method, two operations incorporate viewpoint information. Firstly, in the point cloud generation stage, the reverse projection function π^{-1} integrates viewpoint information. Secondly, in the mesh generation stage, the projection function π incorporates viewpoint information. Notably, the second application of the projection function's viewpoint noise can effectively counterbalance the viewpoint noise introduced by the reverse projection function. As a result, the noise in viewpoint information does not lead to erroneous mappings between 2D features and 3D positions. Nonetheless, the noise in viewpoint information can cause a misalignment between the input point cloud generated during the mesh generation phase and the world coordinate system. This discrepancy arises due to deviations from the distribution during training. As shown in Fig. 10, it is apparent that the noise in viewpoint information exerts a negligible impact on the quality of the generated results. This finding highlights the robustness of our method to variations in viewpoint noise.

Visualization from Multiple Viewpoints. We also present visualizations of meshes and point clouds from multiple perspectives to further demonstrate the robustness of our model. Specifically, we set the elevation angle at 20° and the azimuth angles at $\{0°, 45°, 90°, 270°\}$ degrees to visualize the meshes and point clouds generated by our method. Figure 11 displays the results of these multi-view visualizations, confirming that the meshes and point clouds produced by our approach maintain fidelity when observed from various angles.

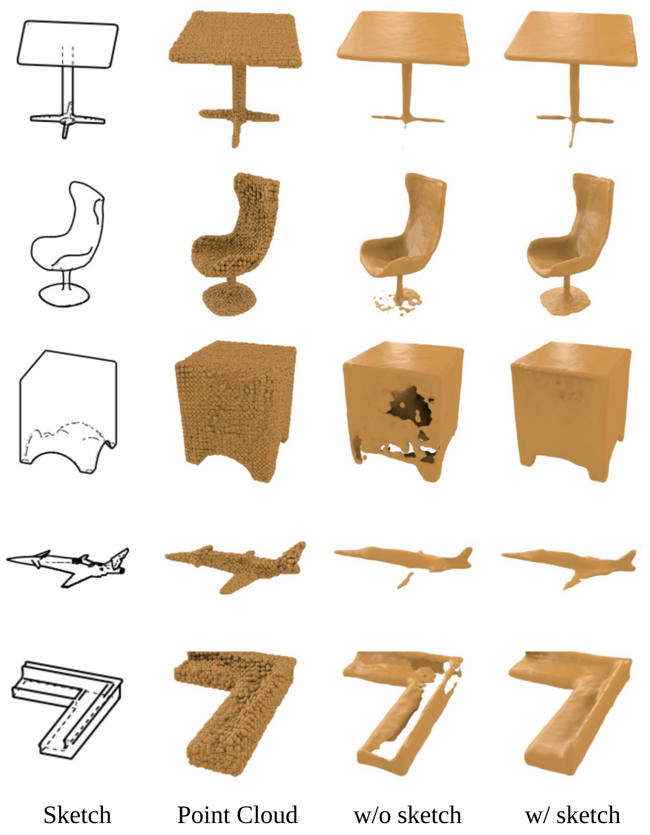


Fig. 12. Ablation results on sketch-aware mesh generation.

G. Ablation Study

Density-guided Sampler. In the point cloud generation process, our method utilizes a density map as guidance for sampling the u and v coordinates. To further validate the effectiveness of the density map, we compare our sampler with another two variants, which treat the 3D information as uniformly distributed in the 2D space. Specifically, the term "uniform sampler" refers to the variant that samples points uniformly from a foreground region, which is derived from the ground truth density map with a threshold set to zero, during

TABLE II
THE IMPACT OF DIFFERENT DENSITY MAPS ON THE POINT CLOUD RECONSTRUCTION. (EARTH MOVER'S DISTANCE(↓) $\times 10^{-2}$)

	Categories													**mean**
	airplane	bench	cabinet	car	chair	display	lamp	speaker	rifle	sofa	table	phone	boat	
ours	7.717	8.283	8.489	7.953	10.607	8.651	17.574	11.356	4.655	10.018	8.385	6.134	9.346	9.167
real sampler	7.559	8.073	8.267	7.779	10.594	8.540	17.315	11.142	4.519	10.024	8.378	6.034	9.105	9.025
uniform sampler	8.110	8.849	8.884	8.021	11.701	8.899	16.033	11.416	4.699	10.298	8.875	6.035	8.989	9.293
uniform sampler (retrained)	8.337	8.922	9.068	8.359	11.958	9.003	16.181	11.735	4.749	10.680	8.847	6.057	9.121	9.463

TABLE III
THE IMPACT OF DIFFERENT SKETCH REPRESENTATIONS ON THE POINT CLOUD RECONSTRUCTION. (EARTH MOVER'S DISTANCE(↓) $\times 10^{-2}$)

	Categories													**mean**
	airplane	bench	cabinet	car	chair	display	lamp	speaker	rifle	sofa	table	phone	boat	
ours (front + back + composed)	7.717	8.283	8.489	7.953	10.607	8.651	17.574	11.356	4.655	10.018	8.385	6.134	9.346	9.167
front $\times$ 3	8.017	8.837	11.749	8.011	11.918	9.293	18.581	14.366	4.844	10.053	10.260	6.553	8.622	10.085
composed $\times$ 3	7.955	8.573	9.075	8.499	10.847	8.694	17.964	11.438	4.896	9.672	9.357	6.070	9.180	9.402
front + hidden + composed	7.855	8.400	8.761	7.945	10.778	8.608	17.861	11.630	4.784	10.019	8.696	6.038	9.216	9.276
predicted 3-channel sketch	8.061	8.878	8.814	8.637	11.194	8.987	18.752	11.815	5.072	10.604	8.748	6.489	9.745	9.677

TABLE IV
THE EFFECTIVENESS OF INCORPORATING SKETCH IN THE MESH RECONSTRUCTION. (EARTH MOVER'S DISTANCE(↓) $\times 10^{-2}$)

	Categories													**mean**
	airplane	bench	cabinet	car	chair	display	lamp	speaker	rifle	sofa	table	phone	boat	
G_m w/ sketch (Ours)	5.953	8.064	7.124	5.823	8.225	6.500	15.698	10.229	4.355	5.780	6.825	4.724	6.351	7.358
G_m w/o sketch	6.091	8.134	7.225	5.936	8.591	6.757	16.101	10.111	4.447	5.898	6.877	4.768	6.353	7.484

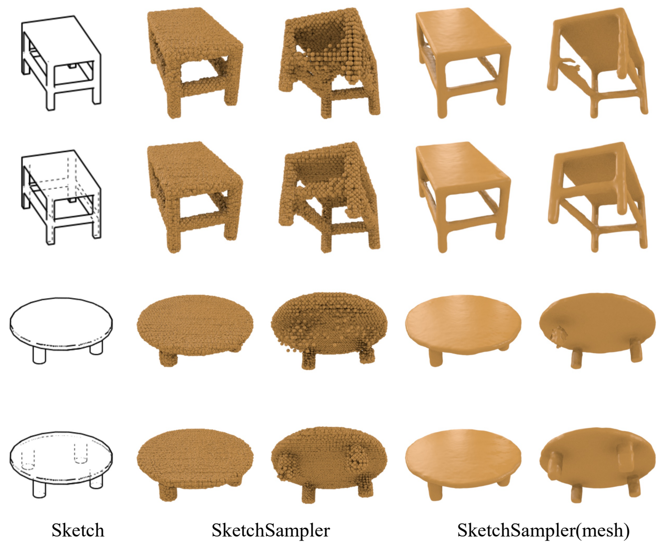


Fig. 13. Impact of hidden lines. For each example, the first row shows the results of our model trained on sketches *without* hidden lines, while the second row shows the results of our final model, which is trained on sketches *with* hidden lines. It is clear to see that introducing hidden lines can significantly help to eliminate the ambiguity of sketches and lead to better performance.

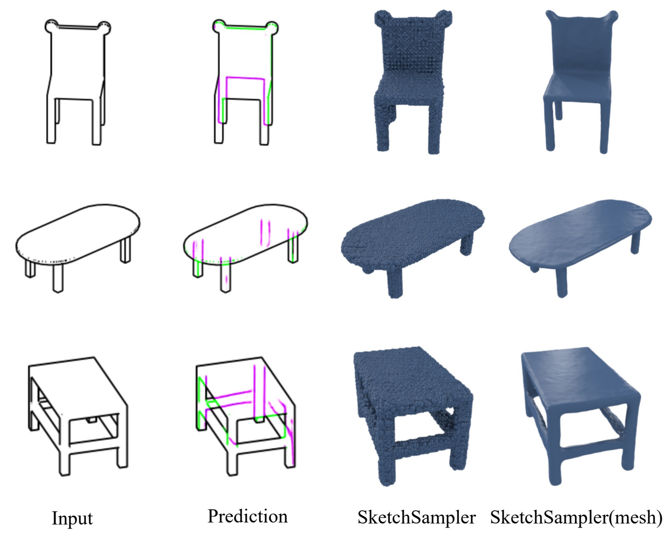


Fig. 14. Adaptation to the front view sketch. The green lines in 'Prediction' represent lines that only appear in the front view, the purple lines represent lines that only appear in the back view, and the black lines represent lines that appear in both the front view and the back view.

inference. We also experimented by retraining our network to directly predict the foreground mask. This variant is denoted as "uniform sampler (retrained)". Additionally, we employed real density maps to generate projection points for testing, which we refer to as the "real sampler".

The experimental results are shown in Table II. The uniform sampling strategy does not allow the sampler to perceive the differences in the distribution of $p(d|u, v; I)$ at different positions. We can observe that the reconstruction performance is worse when using the uniform sampler compared to our method. Moreover, if the quality of the density maps reaches real data, then the overall quality is further improved. This comparison demonstrates the effectiveness of our density-guided sampler. By incorporating the density map as guidance, our method can capture the distribution characteristics of the 3D information in the 2D space, leading to improved reconstruction performance.

Sketch-aware Point Cloud Enhancement. In our point cloud enhancement network, we introduce sketch local features as additional information. This is because the initial reconstructed point clouds may contain various types of noise. By incorporating sketch as additional input, we can improve the quality of the enhanced point cloud. We removed the sketch and its corresponding sketch translator, then retrained the model for the mesh generation stage. Subsequently, we conducted tests to validate the effectiveness of this design alteration. As shown in Tab. IV, by leveraging the sketch local features, our point cloud enhancement network achieves higher-quality reconstructions. Sketch information provides valuable guidance for normal prediction and denoising the initial reconstructed point clouds. The comparisons presented in Fig. 12 demonstrate the enhanced details and better alignment with the input sketches in the reconstructed mesh. Sketch information helps to capture finer details, improve the overall shape accuracy, and enhance the structural integrity of the reconstructed mesh.

Impact of Hidden Lines. To demonstrate the impact of hidden lines on the task of 3D reconstruction from a single sketch, we conducted a retraining process on both the point cloud generation model and the mesh generation model using the Synthetic-LineDrawing dataset *without* hidden lines. For the newly trained models, the input sketches for all three channels were front view sketches. We then evaluated the quality of both point cloud and mesh generation. The quantified results, presented in Table II and Table IV, clearly indicate a noticeable decline in performance when sketches lack hidden lines. Figure 13 provides visual examples illustrating the results, showing that the reconstructions without hidden lines, while maintaining acceptable quality from a frontal perspective, exhibit evident deficiencies when viewed from the back. Notably, discontinuities are observable, such as in the connections between table legs and among the table legs themselves.

Furthermore, we evaluated the adaptation to the front view sketch. Specifically, we introduced a network to automatically predict the tri-channel representations from a front view sketch and then feed the prediction to SketchSampler. Visual and quantitative results are presented in Fig. 14 and Table III, respectively. It can be observed that the adaptive version of Sketchsampler outperforms the Sketchsampler trained solely on front view sketches, further indicating the benefits of explicit hidden lines for 3D reconstruction tasks.

V. Limitations

While our approach accurately captures the intricacies present in hand-drawn illustrations, it is not a probabilistic generative model. This implies that it does not explicitly model the distribution of the 3D object. Consequently, our method might produce point clouds and meshes that deviate from the true distribution. For example, Fig. 15 shows several failure cases where the input sketches feature complex structures. In such cases, the resulting point clouds exhibit poor quality, and the subsequently predicted normals have low accuracy, making the generation of a plausible mesh unattainable. In the future, incorporating probabilistic generative models can further address this issue, enhancing the visual quality of generated objects.

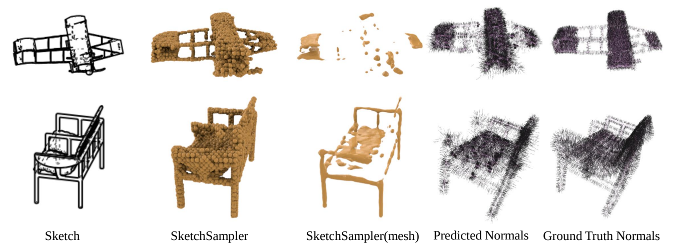


Fig. 15. Failure cases. Our approach tends to introduce confusion in normal prediction during point cloud enhancement when the input sketches feature complex structures, thereby hindering mesh surface reconstruction.

VI. Conclusion

In this study, we introduce a novel method for sketch-based single-view 3D reconstruction, termed *SketchSampler*. Our approach not only generates point clouds from a single sketch but also provides the option to create a mesh. For point cloud generation, we introduce a depth sampling algorithm and use a density map as guidance for the sampling process. In mesh generation, we incorporate sketch features as supplementary information to aid in mesh reconstruction. Additionally, we create a dataset named *Synthetic-LineDrawing*, comprising paired sketches with hidden lines and their corresponding 3D objects. Compared to previous works on sketch-based 3D generation, our method effectively leverages the local features extracted from the sketch. Furthermore, we explore the role of hidden lines in sketch-based 3D reconstruction. Quantitative and qualitative results indicate that our method outperforms state-of-the-art approaches and demonstrates a significant level of robustness to input sketches.

References

[1] 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.

[2] J. Wang, J. Lin, Q. Yu, R. Liu, Y. Chen, and S. X. Yu, “3d shape reconstruction from free-hand sketches,” *arXiv preprint arXiv:2006.09694*, 2020.

[3] S.-H. Zhang, Y.-C. Guo, and Q.-W. Gu, “Sketch2model: View-aware 3d modeling from single free-hand sketches,” in *CVPR*, 2021.

[4] B. Guillard, E. Remelli, P. Yvernay, and P. Fua, “Sketch2mesh: Reconstructing and editing 3d shapes from sketches,” in *ICCV*, 2021.

[5] X.-Y. Zheng, H. Pan, P.-S. Wang, X. Tong, Y. Liu, and H.-Y. Shum, “Locally attentional sdf diffusion for controllable 3d shape generation,” *ACM Transactions on Graphics (SIGGRAPH)*, vol. 42, no. 4, 2023.

[6] 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.

[7] S. Liu, T. Li, W. Chen, and H. Li, “Soft rasterizer: A differentiable renderer for image-based 3d reasoning,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 7708–7717.

[8] E. Remelli, A. Lukoianov, S. Richter, B. Guillard, T. Bagautdinov, P. Baque, and P. Fua, “Meshsdf: Differentiable iso-surface extraction,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 22 468–22 478, 2020.

[9] Q. Xu, W. Wang, D. Ceylan, R. Mech, and U. Neumann, “DISN: deep implicit surface network for high-quality single-view 3d reconstruction,” in *NeurIPS*, 2019.

[10] S. Saito, Z. Huang, R. Natsume, S. Morishima, A. Kanazawa, and H. Li, “Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization,” in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 2304–2314.

[11] L. Li, Z. Huang, C. Zou, C.-L. Tai, R. W. Lau, H. Zhang, P. Tan, and H. Fu, "Model-driven sketch reconstruction with structure-oriented retrieval," in *SIGGRAPH ASIA 2016 Technical Briefs*, 2016, pp. 1–4.

[12] J. D. Camba, P. Company, and F. Naya, "Sketch-based modeling in mechanical engineering design: Current status and opportunities," *Computer-Aided Design*, vol. 150, p. 103283, 2022.

[13] C. Gao, Q. Yu, L. Sheng, Y.-Z. Song, and D. Xu, "Sketchsampler: Sketch-based 3d reconstruction via view-dependent depth sampling," in *European Conference on Computer Vision*. Springer, 2022, pp. 464–479.

[14] H. Xie, H. Yao, X. Sun, S. Zhou, and S. Zhang, "Pix2vox: Context-aware 3d reconstruction from single and multi-view images," in *ICCV*, 2019.

[15] H. Xie, H. Yao, S. Zhang, S. Zhou, and W. Sun, "Pix2vox++: Multi-scale context-aware 3d object reconstruction from single and multiple images," *IJCV*, vol. 128, no. 12, pp. 2919–2935, 2020.

[16] S. Popov, P. Bauszat, and V. Ferrari, "Corenet: Coherent 3d scene reconstruction from a single RGB image," in *ECCV*, 2020.

[17] H. Fan, H. Su, and L. J. Guibas, "A point set generation network for 3d object reconstruction from a single image," in *CVPR*, 2017.

[18] 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 *ECCV*, 2016.

[19] J. Wu, Y. Wang, T. Xue, X. Sun, B. Freeman, and J. Tenenbaum, "Marrnet: 3d shape reconstruction via 2.5d sketches," in *NeurIPS*, 2017.

[20] M. Wang, L. Wang, and Y. Fang, "3densinet: A robust neural network architecture towards 3d volumetric object prediction from 2d image," in *ACM MM*, 2017.

[21] Z. Chen, V. G. Kim, M. Fisher, N. Aigerman, H. Zhang, and S. Chaudhuri, "Decor-gan: 3d shape detailization by conditional refinement," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 15 740–15 749.

[22] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong, "O-cnn: Octree-based convolutional neural networks for 3d shape analysis," *ACM Transactions On Graphics (TOG)*, vol. 36, no. 4, pp. 1–11, 2017.

[23] C. Häne, S. Tulsiani, and J. Malik, "Hierarchical surface prediction for 3d object reconstruction," in *2017 International Conference on 3D Vision (3DV)*. IEEE, 2017, pp. 412–420.

[24] P.-S. Wang, C.-Y. Sun, Y. Liu, and X. Tong, "Adaptive o-cnn: A patch-based deep representation of 3d shapes," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 6, pp. 1–11, 2018.

[25] C. Choy, J. Gwak, and S. Savarese, "4d spatio-temporal convnets: Minkowski convolutional neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 3075–3084.

[26] Z. Huang, S. Stojanov, A. Thai, V. Jampani, and J. M. Rehg, "Zeroshape: Regression-based zero-shot shape reconstruction," *arXiv preprint arXiv:2312.14198*, 2023.

[27] G. Yang, X. Huang, Z. Hao, M.-Y. Liu, S. Belongie, and B. Hariharan, "Pointflow: 3d point cloud generation with continuous normalizing flows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 4541–4550.

[28] C.-H. Lin, C. Kong, and S. Lucey, "Learning efficient point cloud generation for dense 3d object reconstruction," in *proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[29] N. Wang, Y. Zhang, Z. Li, Y. Fu, H. Yu, W. Liu, X. Xue, and Y.-G. Jiang, "Pixel2mesh: 3d mesh model generation via image guided deformation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 10, pp. 3600–3613, 2020.

[30] 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.

[31] C. Wen, Y. Zhang, Z. Li, and Y. Fu, "Pixel2mesh++: Multi-view 3d mesh generation via deformation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1042–1051.

[32] Z. Chen and H. Zhang, "Learning implicit fields for generative shape modeling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5939–5948.

[33] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4460–4470.

[34] 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/CVF conference on computer vision and pattern recognition*, 2019, pp. 165–174.

[35] W. Bian, Z. Wang, K. Li, and V. A. Prisacariu, "Ray-onet: Efficient 3d reconstruction from A single RGB image," in *British Machine Vision Conference(BMVC)*, 2021.

[36] Z.-X. Zou, Z. Yu, Y.-C. Guo, Y. Li, D. Liang, Y.-P. Cao, and S.-H. Zhang, "Triplane meets gaussian splatting: Fast and generalizable single-view 3d reconstruction with transformers," *arXiv preprint arXiv:2312.09147*, 2023.

[37] Y. Hong, K. Zhang, J. Gu, S. Bi, Y. Zhou, D. Liu, F. Liu, K. Sunkavalli, T. Bui, and H. Tan, "Lrm: Large reconstruction model for single image to 3d," *arXiv preprint arXiv:2311.04400*, 2023.

[38] Y. Zhong, Y. Qi, Y. Gryaditskaya, H. Zhang, and Y.-Z. Song, "Towards practical sketch-based 3d shape generation: The role of professional sketches," *IEEE Transactions on Circuits and Systems for Video Technology(T-CSVT)*, vol. 31, no. 9, pp. 3518–3528, 2020.

[39] Y. Zhong, Y. Gryaditskaya, H. Zhang, and Y.-Z. Song, "Deep sketch-based modeling: Tips and tricks," in *2020 International Conference on 3D Vision (3DV)*. IEEE, 2020, pp. 543–552.

[40] L. Wang, C. Qian, J. Wang, and Y. Fang, "Unsupervised learning of 3d model reconstruction from hand-drawn sketches," in *ACM MM*, 2018.

[41] A. Sanghi, P. K. Jayaraman, A. Rampini, J. Lambourne, H. Shayani, E. Atherton, and S. A. Taghanaki, "Sketch-a-shape: Zero-shot sketch-to-3d shape generation," *arXiv preprint arXiv:2307.03869*, 2023.

[42] C. Li, H. Pan, A. Bousseau, and N. J. Mitra, "Sketch2cad: Sequential cad modeling by sketching in context," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–14, 2020.

[43] ——, "Free2cad: Parsing freehand drawings into cad commands," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 4, pp. 1–16, 2022.

[44] A.-D. Nguyen, S. Choi, W. Kim, and S. Lee, "Graphx-convolution for point cloud deformation in 2d-to-3d conversion," in *ICCV*, 2019.

[45] D. Shin, C. C. Fowlkes, and D. Hoiem, "Pixels, voxels, and views: A study of shape representations for single view 3d object shape prediction," in *CVPR*, 2018.

[46] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *CVPR*, 2017.

[47] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[48] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

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

[50] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger, "Convolutional occupancy networks," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*. Springer, 2020, pp. 523–540.

[51] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[52] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3d u-net: learning dense volumetric segmentation from sparse annotation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2016: 19th International Conference, Athens, Greece, October 17-21, 2016, Proceedings, Part II 19*. Springer, 2016, pp. 424–432.

[53] S. Peng, C. Jiang, Y. Liao, M. Niemeyer, M. Pollefeys, and A. Geiger, "Shape as points: A differentiable poisson solver," *Advances in Neural Information Processing Systems*, vol. 34, pp. 13 032–13 044, 2021.

[54] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," in *Seminal graphics: pioneering efforts that shaped the field*, 1998, pp. 347–353.

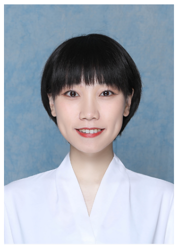
[55] B. O. Community, *Blender - a 3D modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. [Online]. Available: http://www.blender.org

[56] S. Laine, J. Hellsten, T. Karras, Y. Seol, J. Lehtinen, and T. Aila, "Modular primitives for high-performance differentiable rendering," *ACM Transactions on Graphics*, vol. 39, no. 6, 2020.

[57] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

Chenjian Gao received his B.S. degree from the School of Software, Beihang University, Beijing, China in 2021. He is currently pursuing a Master's degree at Beihang University. His current research interests include 3D reconstruction and generative models.

Xilin Wang received his B.S. degree from the School of Software, Beihang University, Beijing, China in 2022. He is currently pursuing a Master's degree at Beihang University. His recent research interests include machine learning and computer vision, with the special interests in 3D reconstruction, computer-aided design, and their combination with LLM.



Qian Yu received her Ph.D. degree from Queen Mary University of London, UK, in 2018. From 2018 to 2019, she was a post-doctoral researcher at UC Berkeley. She is now an associate professor at Beihang University. She was a coauthor of a paper that won the Best Paper Award at the British Machine Vision Conference (BMVC) in 2015. Her research interests include computer vision and deep learning, with special interests in sketch understanding and applications.

Lu Sheng received the B.E. degree from Zhejiang University, China, in 2011, and the Ph.D. degree from The Chinese University of Hong Kong, Hong Kong, in 2016. From 2017 to 2019, he was a Post-Doctoral Researcher with the Multimedia Laboratory (MMLab), The Chinese University of Hong Kong. He is currently an Associate Professor with the College of Software, Beihang University, China. His research interests include 3D computer vision and embodied AI.

Jing Zhang received her Ph.D. degree from University of Wollongong, Australia, in 2019. She is now an associate professor in School of Software, Beihang University, Beijing, China. Her recent research interests include machine learning and computer vision, with the special interests in transfer learning, multi-modality learning, open-vocabulary learning, and their applications in data-efficient computer vision.

Xiaoguang Han is now an assistant professor with The Chinese University of Hong Kong, Shenzhen. His research interests include computer vision and computer graphics. He already published over 100 papers in top-tier conferences, such as CVPR, ICCV, ECCV, Siggraph and Siggraph Asia. He is now serving as Area chairs for NeurIPS, CVPR, ECCV etc, Program Committee members of Siggraph Asia and Associate Editor of IEEE TVCG and Computer&Graphics.

Yi-Zhe Song is a Chair Professor of Computer Vision and Machine Learning, and Director of SketchX Lab at the Centre for Vision Speech and Signal Processing (CVSSP), University of Surrey. He obtained a PhD in 2008 on Computer Vision and Machine Learning from the University of Bath, a MSc (with Best Dissertation Award) in 2004 from the University of Cambridge, and a Bachelor's degree (First Class Honours) in 2003 from the University of Bath. He is an Associate Editor of the IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), and International Journal of Computer Vision. He was a Program Chair for British Machine Vision Conference (BMVC) 2021, and regularly serves as Area Chair for flagship computer vision and machine learning conferences, most recently at CVPR'22 and ICCV'21. He is a Senior Member of IEEE, a Fellow of the Higher Education Academy, as well as full member of the EPSRC review college.

Dong Xu received the B.E. and Ph.D. degrees from the University of Science and Technology of China, Hefei, China, in 2001 and 2005, respectively. While pursuing the Ph.D. degree, he was an Intern with Microsoft Research Asia, Beijing, China, and a Research Assistant with The Chinese University of Hong Kong, Hong Kong, for more than two years. He was a Post-Doctoral Research Scientist with Columbia University, New York, NY, USA, for one year. He also worked as a Faculty Member at Nanyang Technological University, Singapore, and the Chair of computer engineering at The University of Sydney, Camperdown, NSW, Australia. He is currently a Professor with the Department of Computer Science, The University of Hong Kong, Hong Kong. He was a coauthor of a paper that won the Best Student Paper Award at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) in 2010 and a paper that won the Prize Paper award in IEEE TRANSACTIONS ON MULTIMEDIA (T-MM) in 2014. His current research interests include computer vision, statistical learning, and multimedia content analysis.