Computers & Graphics (2022)

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

A Study of Deep Single Sketch-Based Modeling: View/Style Invariance, Sparsity and Latent Space Disentanglement

Yue Zhong[a], Yulia Gryaditskaya[b,c], Honggang Zhang[a], Yi-Zhe Song[b,c]

[a]*Beijing University of Posts and Telecommunications*
[b]*SketchX, Centre for Vision, Speech and Signal Processing (CVSSP), University of Surrey*
[c]*Surrey Institute for People Centred AI, University of Surrey*

ARTICLE INFO

Article history:
Received June 8, 2022

Keywords: Deep sketch-based modeling, Single-view reconstruction

ABSTRACT

Deep image-based modeling has received a lot of attention in recent years. Sketch-based modeling in particular has gained popularity given the ubiquitous nature of touchscreen devices. In this paper, we (i) study and compare diverse single-image reconstruction methods on sketch input, comparing the different 3D shape representations: multi-view, voxel- and point-cloud-based, mesh-based and implicit ones; and (ii) analyze the main challenges and requirements of sketch-based modeling systems. We introduce the regression loss and provide two variants of its formulation for the two most promising 3D shape representations: point clouds and signed distance functions. We show that this loss can increase general reconstruction accuracy, and the view- and style-robustness of the reconstruction methods. Moreover, we demonstrate that this loss can benefit the disentanglement of latent space to view-invariant and view-specific information, resulting in further improved performance. To address the figure-ground ambiguity typical for sparse freehand sketches, we propose a two-branch architecture that exploits sparse user labeling. We hope that our work will inform future research on sketch-based modeling.

1. Introduction

The challenge of being able to obtain a 3D model from a single sketch has intrigued researchers for decades. Typically, proposed methods make assumptions on the type of the input [1, 2] or restrict users to a specific user interface [3, 4, 5]. Being an under-constrained problem, for which it is hard to devise a reliable set of heuristics, it naturally asks for deep learning-based methods.

In light of the recent surge of image-based reconstruction, deep sketch-based modeling is gaining popularity [6, 7, 8, 9]. In this work, we study and compare the state-of-the-art deep single RGB image 3D shape reconstruction methods on sketch inputs. This allows us to identify the main challenges of working with sketch input: style variance between humans, imprecise perspective, and sparsity. We propose targeted solutions to increase the robustness of existing methods on a sketch input.

The first challenge comes from style differences, i.e., each person carries a unique sketching style. To address this problem, we use for training three synthetic datasets: naive, stylized, and one where the style is unified by an additional image processing network, as we proposed in the conference version of this paper [10]. The *naive* dataset represents rendering style with a uniform line width, commonly used in sketch-based reconstruction papers, where the lines are obtained from 2D images or via non-photorealistic rendering from 3D models. In this work, we rely on the latter. The *stylized* dataset is designed to capture the diversity of freehand sketching styles. The

e-mail: y.zh@bupt.edu.cn (Yue Zhong), julia.poste@gmail.com (Yulia Gryaditskaya), zhhg@bupt.edu.cn (Honggang Zhang), y.song@surrey.ac.uk (Yi-Zhe Song)

strategy of the style-unifying image translation network in the context of sketch-based modeling was first proposed in [6] and proved to be efficient on doodle sketches. We aim at more detailed sketches and show that if the sketching style is within an expected variance on line widths and over-sketching then training on the proposed stylized dataset results in more accurate reconstructions compared to a style-unifying network.

Second, we address the robustness of reconstruction methods with respect to viewpoint and style. It is common for deep single image methods to train and test their models on a predefined set of viewpoints [11]. Nevertheless, it was observed by Gryaditskaya et al. [12] that even professional designers, when asked to sketch from a given viewpoint, produce sketches with large angular deviations from the set viewpoint. Therefore, we create a dataset by generating for each shape 48 viewpoints, where 8 viewpoints are fixed and 5 additional viewpoints for each viewpoint are randomly sampled from a normal distribution with the mean matching the parameters of one of the fixed viewpoints. To explicitly account for the variation of styles and viewpoints, we aim at learning style- and viewpoint-invariant shape representation by proposing a regression loss that encourages the correlation of distances between 3D shapes and distances in a latent feature space. We extend [10] by proposing a new formulation of the regression loss, enabling the representation of 3D shapes as signed distance functions in the context of the regression loss. We show that this representation achieves the most accurate reconstruction results. We carefully study the choice of the parameter used in this loss, sampling and training strategies. Moreover, we show the advantage of this loss on the task of feature space disentanglement to view-invariant and view-specific components, allowing to further improve the accuracy of the reconstruction results.

The final challenge in deep sketch-based reconstruction comes from the difficulties in distinguishing the foreground from the background, due to the sparsity of sketch lines. To alleviate this problem we suggest a simple framework, where the user can provide a few sparse labels, and the network propagates these labels to robustly predict shape foreground binary mask, which is then passed to a 3D shape reconstruction network alongside the input sketch. We apply the strategy proposed in [10] and show its efficiency in the context of SDFs.

In summary, we propose the following contributions:

- We discuss the key challenges inherent to sketch input in the context of deep-reconstruction methods.
- We compare alternative strategies to handle freehand sketching styles variations.
- We adopt the regression loss to learn style- and viewpoint-invariant sketch embedding, and provide two formulations of the regression loss targeted two 3D shape representations: point clouds and signed distance functions.
- We study the effectiveness of the regression loss in neighborhood relations preservation between the reconstruction results, its style and view invariance properties.
- Finally, we propose to use an auxiliary network that learns to predict foreground mask from the input sketch and supports user sparse labels when necessary. We demonstrate how such mask can be incorporated as an input to a reconstruction network and allows to better account for the sparsity of input sketches.

2. Related work

For a general overview of existing sketch-based reconstruction methods, please refer to a recent survey by Bonnici et al. [13]. In this paper, we focus on sketch-based modeling relying on recent advances in deep learning. Moreover, we focus on scenarios where the end user provides a detailed sketch, however, possibly with perspective distortions and over-sketching.

Research on sketch-based modeling using deep learning approaches is limited by the lack of large freehand sketch datasets paired with ground-truth 3D shapes. Differentiable rendering, recently proposed to enable unsupervised training on real images [14, 15, 16, 17, 18], is not directly applicable to sketch input. Two recent works [8, 9] leverage differentiable rendering for sketch-based modeling. Zhang et al. [8] claim that 3D information is not needed in their training process, however they note that the training requires ground-truth view point and silhouettes, which can only be obtained with synthetic data generated from 3D meshes. Nevertheless, they use a domain adaptation approach that allows them to train with synthetic paired sketches, and unpaired freehand sketches[1]. Guillard et al. [9], first train the reconstruction network under a supervised setting and then use differentiable rendering to improve the reconstruction results for each sketch input via an optimization procedure, assuming that the sketch's viewpoint is known. The focus of our work is different: (i) we focus on a supervised training setting, (ii) we compare common deep learning shape representations, and (iii) we propose solutions to address sketch sparsity, and robustness to viewpoint and style that can be integrated into most existing reconstruction frameworks. Therefore, in this section, we group reconstruction methods according to used shape representation, and discuss which methods we evaluate in this work.

Multi-view. Multi-view shape representation [19] is one of the earlier approaches to reconstruct 3D shapes, in which multiple viewpoints, depth or normal maps are predicted. Recently, Yao et al. [20] incorporated symmetry analysis into a shape predictive network. Here we evaluate two sketch-dedicated approaches *ShapeMVD* by Lun et al. [11] and *ProSketchMV* by Zhong et al. [7]. Lun et al. [11] predicts normal and depth maps as seen from 12 viewpoints, which are fused to a dense point cloud. The mesh is obtained from multiple views by the Poisson Surface Reconstruction [21]. Zhong et al. [7] built on [11] and introduced attention loss in the context of sketch-based modeling and leverage Spatial Transform Module (STM) to consistently align input sketches. Li et al. [22] target freeform surfaces and introduced the intermediate layer that predicts dense curvature directions. We do not evaluate this method due to

[1] They collect freehand sketches but do not publish them.

its restrictive sketch input assumptions: silhouette lines are assumed to be sketched in black and other lines in gray. For each sketch, they compute the sketch silhouette mask by a flooding process that starts from background pixels and stops at arbitrary sketch pixels. Their method supports sparse labels for depth maps and curvature hints for strokes. Our strategy to handle sketch sparsity is similar, but we use a neural network to predict a foreground/background mask using intuitive binary hints from a user, greatly reducing the requirement for user input.

Voxels. Voxel shape representation [23, 24, 25, 26, 27] has a regular structure and therefore allows to adopt state-of-the-art deep learning techniques for 2D images. The voxels occupancy is converted to a mesh by a marching cube algorithm [28]. The disadvantages of such methods are large memory footprint and low-resolution results. Such representation were also considered in the context of sketch-based modeling [29, 30], however, the proposed solutions are not sketch-specific. Here, we evaluate the classic *3D-R2N2* [24] method as a representative of this class of methods.

Point clouds. Point cloud is a native output of diverse sensing devices, and therefore is a popular 3D shape representation in the context of deep learning [31, 32, 33, 34]. We use the popular *PSGN* [31] as the method that works with point cloud shape representation. Wang et al. [6] adopted [31] for sketch-based 3D reconstruction by proposing an additional image translation network that aims to standardize sketch styles to account for sketch style variability. Similarly to this work, we test the importance of the style-standardization module and explore alternative strategies to enable style- and viewpoint-invariance.

Mesh. A pioneering work on a direct image to mesh translation *Pixel2Mesh* [35] introduced a graph-based convolutional neural network limited to 3D meshes with genus 0, that predicts the deformation of an ellipsoid. *TMNet*, an approach by Pan et al. [36], aims to support the reconstruction of 3D shapes with genus greater than 0. It introduces an additional module for errors predictions and mesh faces removal step. We evaluate both approaches.

Implicit representation. Recently a number of works were proposed which use implicit functions to represent a shape [37, 38, 39]. Some works learn [37, 38, 40] to predict for each point if it is free or occupied. Implicit representations are relatively lightweight as point clouds, and are not so limited to resolution as explicit shape representations. The disadvantage of these methods is, in order to reconstruct the 3D shape, additional processing is still required. Signed distance fields [40] is another popular continuous implicit shape representation: The magnitude of a point in the field represents the distance to the surface boundary and the sign indicates whether the region is inside (-) or outside (+) of the shape. This representation is limited to shapes with closed surfaces. To support an open surfaces representation Chibane et al. [41] introduced *unsigned distance fields*. Chen et al. [39] directly outputs an approximation of a surface, and handles well sharp shape features, what though comes at the cost of poor reconstruction of curved surfaces.

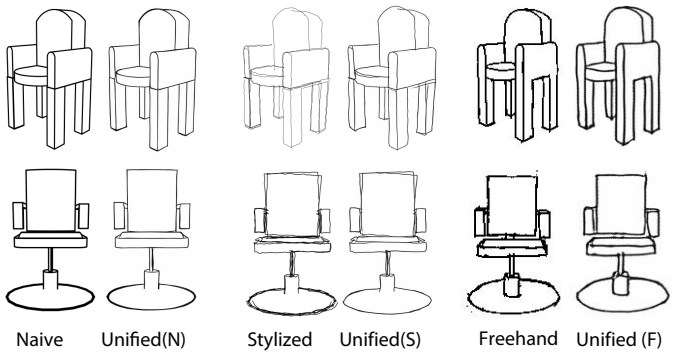

Fig. 1: Example sketches from naive, stylized and style-unified synthetic datasets [10], as well as sketches from ProSketch3D dataset of freehand sketches [7]. *Naive* denotes a synthetic sketch generated from a reference 3D model using the rendering of silhouettes and creases with a uniform stroke width. *Stylized* denotes synthetic sketches that support over-sketching, random global- and stroke-level transformation and varied strokes width. *Unified(N/S/F)* denotes Naive synthetic/Stylized synthetic/Freehand sketches passed through an additional network that aims to unify their appearance. The details on the datasets for completeness are provided in Appendix A.

We evaluate *OccNet [37]* that predicts per point occupancy and *DeepSDF [40]* that represents 3D shapes using signed distance fields (SDFs).

Other. Groueix et al.[42] proposed to approximate the surface locally by mapping a set of squares to a 3D shape. This approach produce not closed meshes that can have holes and self-intersections. Smirnov et al. [43] proposed a new shape representation as an assembly of Coons patches, where the main goal is to obtain the representation, in an end-to-end manner, that can be easily manipulated by designers. Their formulation requires construction of a category-specific template.

3. Overview

First, we study and compare diverse single-image reconstruction methods (Section 4) on sketch input, comparing the different 3D shape representations: multi-view, voxel- and point-cloud-based, mesh-based, and implicit ones. We then analyze and address the main challenges and requirements of single-view sketch-based modeling systems in Sections 5 and 6.

4. Analysis of various 3D shape representations

In the conference version of this paper [10], we have compared quantitatively and qualitatively a set of 3D reconstruction methods. In this work, we extend this set and also consider the popular representation of 3D shapes with signed distance functions [40] and a recently proposed sketch-dedicated method by Zhong et al. [7] that predicts multi-view 3D shape representation. For completeness, we summarize all reconstruction methods used in [10] in Appendix B.

4.1. Implementation details

4.1.1. Datasets

We use the datasets of synthetic sketches as we proposed in [10] and freehand sketches from the ProSketch3D dataset [7]

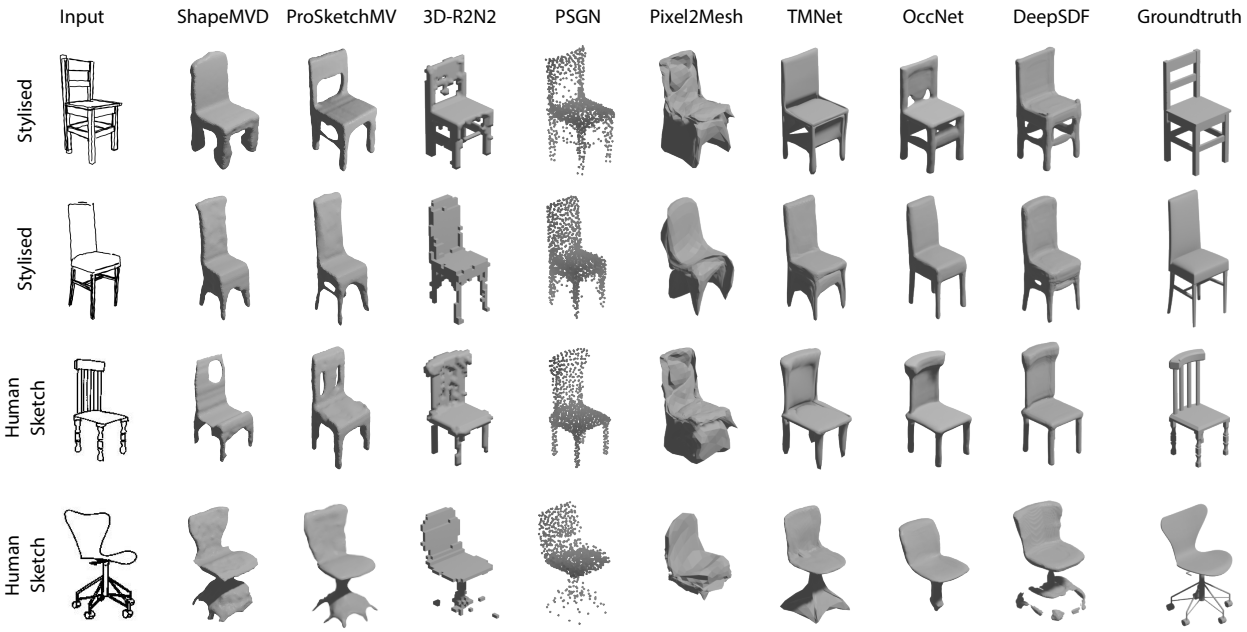

Fig. 2: Visualization of reconstruction results of different baselines on the test synthetic stylized dataset and freehand sketches: ShapeMVD [11], ProSketchMV [7], 3D-R2N2 [24], PSGN [31], Pixel2Mesh [35], TMNet [36], OccNet [37], DeepSDF [40].

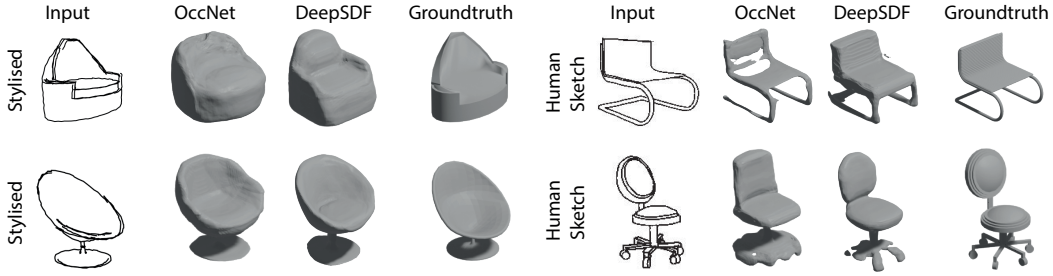

Fig. 3: Visualization of reconstruction results of the models with implicit shape representations on the test synthetic stylized and freehand sketches.

(Fig. A.14). Each shape in synthetic datasets has 48 sketch views (Appendix A.3). We use synthetic datasets for training.

4.1.2. Batch composition

During training, for each 3D shape in the mini-batch, we randomly select one sketch viewpoint from 48 generated viewpoints. For the baselines considered in the conference version of this paper [10], the batch size is set to 32, for DeepSDF [40] we set it to 8, and for ProSketchMV [7] we set it to 16. We train each method to convergence.

4.1.3. Shape-specific SDF sampling

To generate the ground-truth data in the SDFs representation, we follow the approach proposed in [40]: Namely, we randomly sample $250k$ points on the surface of the mesh, weighted by triangle areas. We perturb each surface point along all X, Y, and Z axes with zero mean Gaussian noise with variance values of 0.012 and 0.035 to generate two spatial samples per surface point. In addition, we uniformly sample $25k$ points within the unit box. Thus, in total we sample $525K$ points. During training, we sample a subset of 8,192 points from a pre-computed set of points for each ground-truth and compare their SDF values with the predicted SDF values at those points with an L_1 loss.

4.2. Qualitative analysis of baselines

Figure 2 provides visualizations of 3D reconstructions obtained using different state-of-the-art methods on stylized and freehand sketches, trained on stylized sketches. To generate the results, we use the code provided by the authors of the respective models. This figure demonstrates that all baselines are able to predict an overall 3D shape resembling an input sketch. It can be observed that 3D-R2N2 [24], which predicts voxel occupancy, leads to visually unpleasant results due to the presence of visible 3D voxels. PSGN [31] overall reproduces 3D shapes well but is limited to sparse representations. Pixel2Mesh [35] is limited in its ability to model shapes with genus higher than zero. Note that Pixel2Mesh performs perceptual feature pooling, which may lead to worse performance on sparse sketch inputs than on images. TMNet [36] performs better than Pixel2Mesh, but still contains a large number of artifacts and generally fails to reproduce fine details. The networks that work with implicit shapes representations, OccNet [37] and DeepSDF [40], achieve the best performance alongside ProSketchMV [7] and produce visually appealing results. We observe that DeepSDF predicts finer contours and does better at reconstructing fine details than OccNet. We show a more detailed comparison of OccNet and DeepSDF in Fig. 3 which shows that DeepSDF results in more accurate reconstructions.

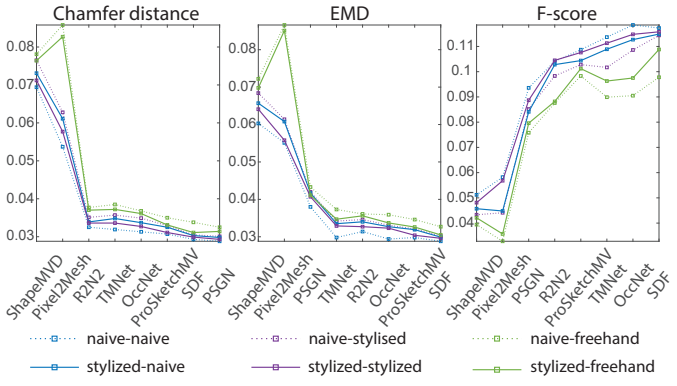

Fig. 4: *Synthetic naive sketch vs stylized sketch.* We train all the baselines with our naive and stylized datasets, and test on the naive sketch test set, stylized dataset and sketches from SketchPro3D which match our test set of shapes. In each pair, e.g. naive-naive, the first refers to the training set and the second to the test set. For the first two measures the lower is the better, while for the F-score the higher is the better. The baselines are sorted according to the performance on the stylized test set when the methods are trained on the stylized dataset.

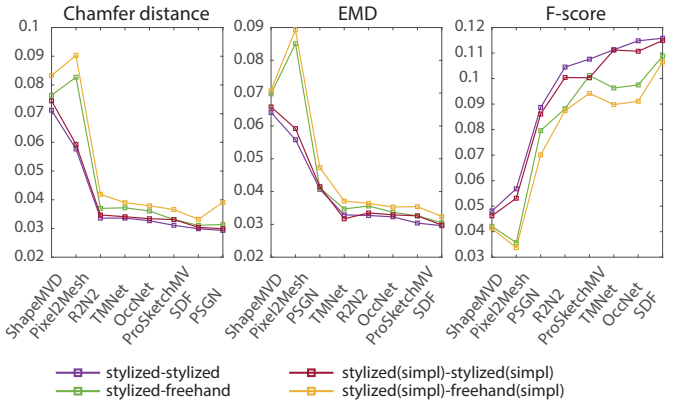

Fig. 5: *Stylized sketch vs style-unified sketch (Section 4.3.3).* Comparison of the performance of the method when training on our stylized dataset against when training using an additional sketch style-unifying network. We use the sketch simplification network by Simo-Serra et al. [45], denoted as '(simpl)'. The baselines are sorted according to the performance on the stylized test set when the methods are trained on the stylized dataset.

4.3. Quantitative analysis of baselines

First, we evaluate the generalization properties of different methods for synthetic sketches in different styles and freehand sketches. Second, we analyze performance across several different shape categories, as well as compare training on single and multiple shape categories.

4.3.1. Evaluation metrics

We use three evaluation metrics to compare the predicted 3D shapes quantitatively to reference shapes: Chamfer distance [31], Earth mover's distance [31] and F-Score [44]. We provide their equations in Appendix B.1.

4.3.2. Test dataset viewpoints

For a fair but tractable evaluation (inference for some baselines is quite costly), for each 3D shape in the synthetic test dataset, we randomly select one of 48 viewpoints. For the dataset of freehand sketches, for each shape, we similarly randomly select one of the three available viewpoints. The random sampling is done once, so the test datasets are exactly the same for all evaluated models.

4.3.3. Style robustness

Synthetic naive sketch vs stylized sketch. We first compare the baselines when trained on naive and stylized datasets on a chair category only. The 6,778 chair shapes from the ShapeNetCoreV2 dataset are split between training and test sets so that the test set consists of the 500 shapes present in the ProSketch3D dataset [7], the remaining 6,278 shapes are used for training. We evaluate the performance on the test set of shapes rendered with a naive style or stylized style, as well as on the corresponding freehand sketches from ProSketch3D.

As seen in Fig. 4, training on stylized sketches improves the performance of freehand sketches as judged by all measures. Yet, one can see that when trained with stylized sketches the performance on naive sketches is worse than when trained on naive sketches, implying that for optimal performance the stylized dataset should include more clean sketch examples. Training on stylized sketches and testing on naive sketches gives better results than training on naive and testing on stylized sketches, indicating the need for stylization.

It can be observed that ShapeMVD [11], ProSketchMV [7] and PSGN [31] are quite robust to different styles, while Pixel2Mesh [35] shows poor generalization properties – Pixel2Mesh performance on freehand sketches is much worse than on synthetic sketches. SDF, PSGN, OccNet, ProSketchMV, 3D-R2N2 and TMNet demonstrate comparable performance, leaving Pixel2Mesh and ShapeMVD behind. Further, it can be observed that implicit shape representations and ProSketchMV [7] achieve the highest accuracy. PSGN achieves good performance according to the Chamfer distance, which this method optimizes for during training, but it loses other methods according to other metrics. Overall, DeepSDF, followed by ProSketchMV, outperforms alternative approaches. It is worth noting that DeepSDF has a shorter training and inference time than ProSketchMV.

Stylized sketch vs style-unified sketch. We then compare the performance of methods trained on our stylized data and tested on stylized/freehand sketch test sets against the performance of methods trained on stylized sketches processed with the style-unifying network [46] and tested on stylized/freehand sketches pre-processed with the same style-unifying network. Figure 5 shows that the performance of methods trained with the stylized dataset outperforms the results obtained with the usage of a style-unifying network. This indicates that when one is interested in an accurate reconstruction using deep learning methods, training data that models diverse sketching styles might be preferable over a style-unifying network. It also highlights the need for datasets of freehand sketches.

4.3.4. Single category vs multiple categories

Additionally, we evaluate how training on multiple categories compares to training on a single category. For training, we generate synthetic sketches of 6,278 chair, 1,822 lamp and 3,548

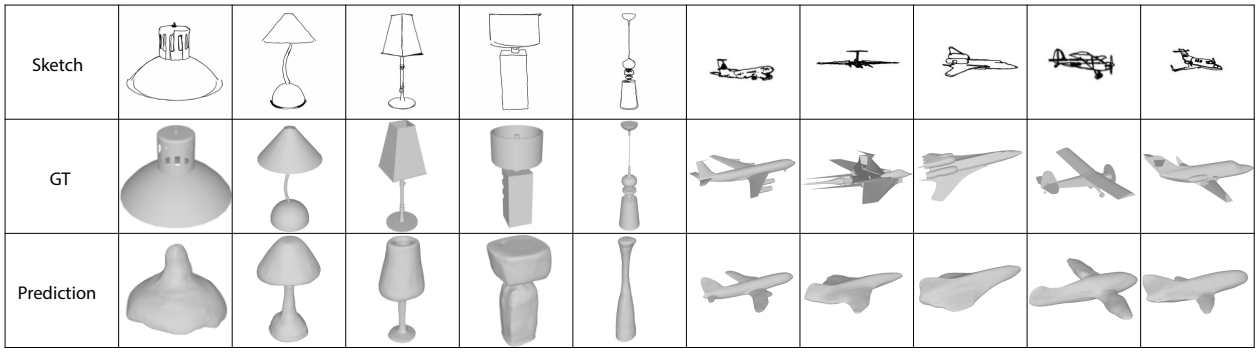

Fig. 6: Qualitative reconstruction results on synthetic stylized sketches from 'lamp' and 'plane' shape categories.

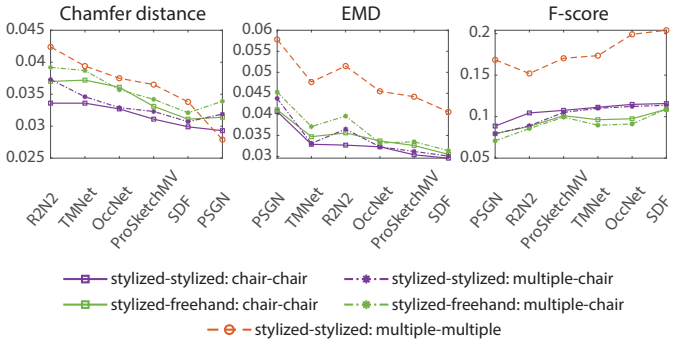

Fig. 7: *Single category vs multiple categories.* Comparison of models performance when training on one category (chairs) versus when trained on multiple categories: chairs, lamps and airplanes. 'multiple-chair' means that the training was done on multiple categories but tested only on chairs. 'multiple-multiple' means that training and testing are done on multiple categories. The baselines are sorted according to the performance on the stylized test set when the methods are trained on the stylized dataset.

airplane 3D shapes in naive in stylized styles. For testing, we select 500 shapes from each of three categories ('chair', 'lamp', 'airplane') non-overlapping with the training set.

It can be seen in Fig. 7 and Table 1 that training on multiple categories and testing on a single chair category results in slightly worse performance for all baselines than training only on a single chair category. The performance for the 'lamp' category is the best across the three considered categories, which can be explained by the distinctive appearances of lamps Fig. 6. On the other hand, worse performance for the 'plane' category can be explained by the general similarity of the plane models, which differ in small details Fig. 6. In addition, these small details are hard to depict in small sketches using synthetic line renderings.

5. Sketch sparsity

Due to the sparsity of information in sketch images, single image reconstruction methods often can not reliably distinguish foreground from the background. To alleviate this problem, we use image translation network [47] and the idea of interactive sparse user labeling [48, 49] to first predict foreground binary mask that we then leverage as an additional input to 3D shape reconstruction methods, as we proposed in the conference paper [10].

Table 1: Quantitative results of training and testing on the stylized synthetic dataset, 'chair-chair' means that the training was done on the 'chair' category and tested on the 'chair' category. 'multiple-multiple' means that training and testing are done on multiple categories: 'lamps', 'chairs' and 'planes'. 'multiple-chair/lamp/plane' means that the training was done on multiple categories but tested only on one indicated category.

Method	CD↓	EMD↓	F-Score↑
stylized-stylized: chair-chair	0.0299	0.0296	0.1158
stylized-stylized: multiple-multiple	0.0338	0.0406	0.2041
stylized-stylized: multiple-chair	0.0307	0.0301	0.1137
stylized-stylized: multiple-lamp	0.0216	0.0232	0.3559
stylized-stylized: multiple-plane	0.0491	0.0685	0.1427

5.1. Binary mask prediction with sparse user labels

In particular, we assume that the user at inference time can provide sparse labels in the form of several binary point indicators of whether the point belongs to the background or to the foreground. During training, we generate these labels automatically, following the sampling strategy proposed in [48].

We sample sparse labels independently from the foreground and background regions. The number of points in each region is drawn from a geometric distribution with its success probability parameter set to $1/8$. We then select the position of background/foreground labels by sampling from a 2D Gaussian distribution with mean $\mu = 0.5[H, W]$ and covariance matrix $\Sigma = diag([(0.5H)^2, (0.5W)^2])$. To predict a binary mask, we train [47] which takes as input to the generator G a sketch and a point mask. Therefore, the training loss is:

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,m,y}[\log D(y|x, m)] + \mathbb{E}_{x,m,z}[\log(1 - D(G(z|x, m)|x, m)] + \lambda_1 \mathcal{L}_{L1}, \quad (1)$$

where x is a sketch, m is a point mask, y is a foreground binary mask, and z is a random noise vector of the same dimension as the latent space. $D(\cdot)$ is a discriminator conditioned on the input sketch and point mask. $\mathcal{L}_{L1}$ is an $L1$ loss that minimizes the per pixel distances between the ground-truth foreground mask y and the predicted $G(z|x, m)$.

Evaluation of the foreground mask prediction. Figure 8 shows that with a few sparse user labels our foreground mask prediction network allows to achieve a nearly perfect prediction of the

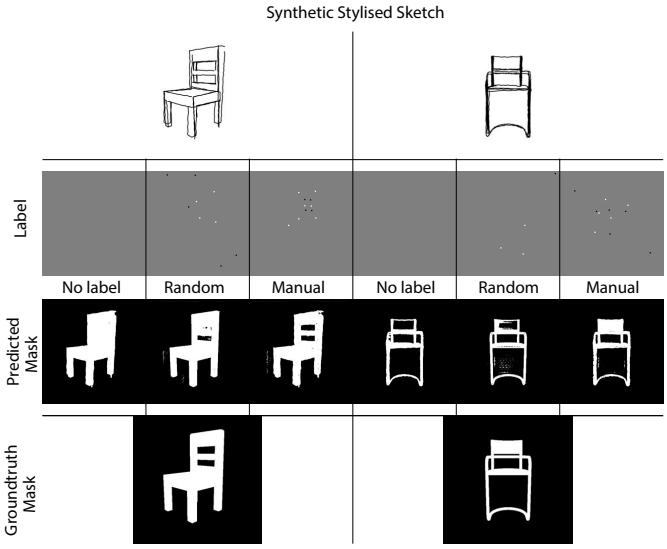

Fig. 8: An example of mask predictions: *'No label'* without any label, *'Random'* with automatically generated labels as described in Section 5.1, and *'Manual'* with labels created by a human.

Table 2: The evaluation of accuracy of the foreground mask prediction on a synthetic stylized test dataset when trained on a stylized training set.

Label	IoU	Precision	Recall
Random Label	0.931	0.951	0.979
No Label	0.834	0.887	0.892

foreground mask. Table 2 provides numerical evaluation when no labels or automatically generated random labels are used.

Note that better performance can be achieved using more recent architectures, for example by training with the Wasserstein GAN and additional loss for the point mask, as suggested in [49].

5.2. Leveraging binary masks

For the PSGN [31], OccNet [37] and DeepSDF [40] methods, we build additional architectures that take both a sketch and a predicted foreground binary mask as input. The mask and sketch are individually passed through two convolutional layers. Each convolutional layer has a kernel size of 3 and stride 1. The first layer maps the $224 \times 224 \times 3$ input to the $224 \times 224 \times 16$ dimensional volume. The second layer preserves the dimension. We then concatenate the outputs of the convolutional layers for sketch and mask inputs and pass them to an encoder.

Figure 9 shows that when an additional foreground mask is used as input, the performance of all baselines is improved in all three considered measures. It also shows that the performance of DeepSDF [40] according to Chamfer distance approaches the one of PSGN when using the additional mask branch. We also observe that the mask branch improves the accuracy on freehand sketches from the ProSketch3D dataset, improving the generalization properties of the method. Figure 10 shows the qualitative improvement in reconstruction results when using additional mask input.

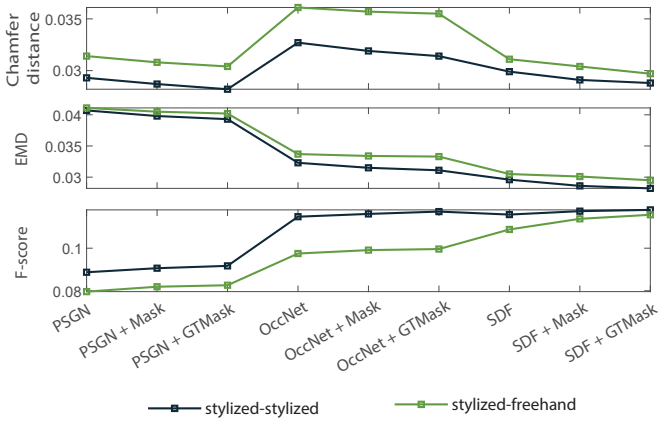

Fig. 9: Evaluation of the effect of the proposed foreground mask branch. '+Mask' represents the results with the masks automatically obtained with our foreground prediction network with randomly generated labels, and '+GTMask' represents the results obtained with a ground-truth input mask, providing an upper estimate of the achievable performance, simulating the case when human labels are used.

6. Viewpoint/style invariant embedded space

An ideal sketch to 3D shape prediction system should be expected to produce consistent 3D geometries independent of the sketch style or viewpoint. The majority of single view reconstruction methods obtain one global feature vector for each input which is then passed to a decoder. Therefore, the requirement on viewpoint/style invariance means that the embeddings of sketches of the same 3D geometry in different styles and from different viewpoints should be identical. To encourage this, in the conference paper [10], we adopt the regression loss, introduced in [50] for deformation-aware 3D model embedding and retrieval. This loss is inspired by the Stochastic Neighbor Embedding (SNE) approach [51], which is a dimensionality reduction method aiming to preserve neighbor identities in a low-dimensional space. In [10], we proposed the formulation that assumes that 3D shapes are represented as point clouds. Here, we first summarize the formulation from [10], and then introduce a new formulation for the 3D shapes represented as SDFs.

6.1. Loss formulation

Let $\mathbf{X}' \subset \mathbf{X}$ be a set of 3D shapes in some mini-batch, which is a randomly sampled subset of all 3D shapes $\mathbf{X}$ in the training set, different for each iteration. First, for each pair of shapes in the mini batch $\mathbf{A}, \mathbf{B} \subset \mathbf{X}'$, we compute the probability of how likely they are to be 'neighbors', based on how similar are the two shapes according to some distance metric in 3D space:

$$p(\mathbf{A}, \mathbf{B}) = \frac{\exp\left(-d^2(\mathbf{A}, \mathbf{B})/2\sigma^2\right)}{\sum_{\mathbf{B}' \in \mathbf{X}'} \exp\left(-d^2(\mathbf{A}, \mathbf{B}')/2\sigma^2\right)}. \quad (2)$$

σ is a pre-computed constant, which we calculate according to a three-sigma rule

$$\sigma = \frac{0.997}{3} d_{max}, \quad (3)$$

where d_{max} is an estimate of the maximum distance among the distances between 3D shapes.

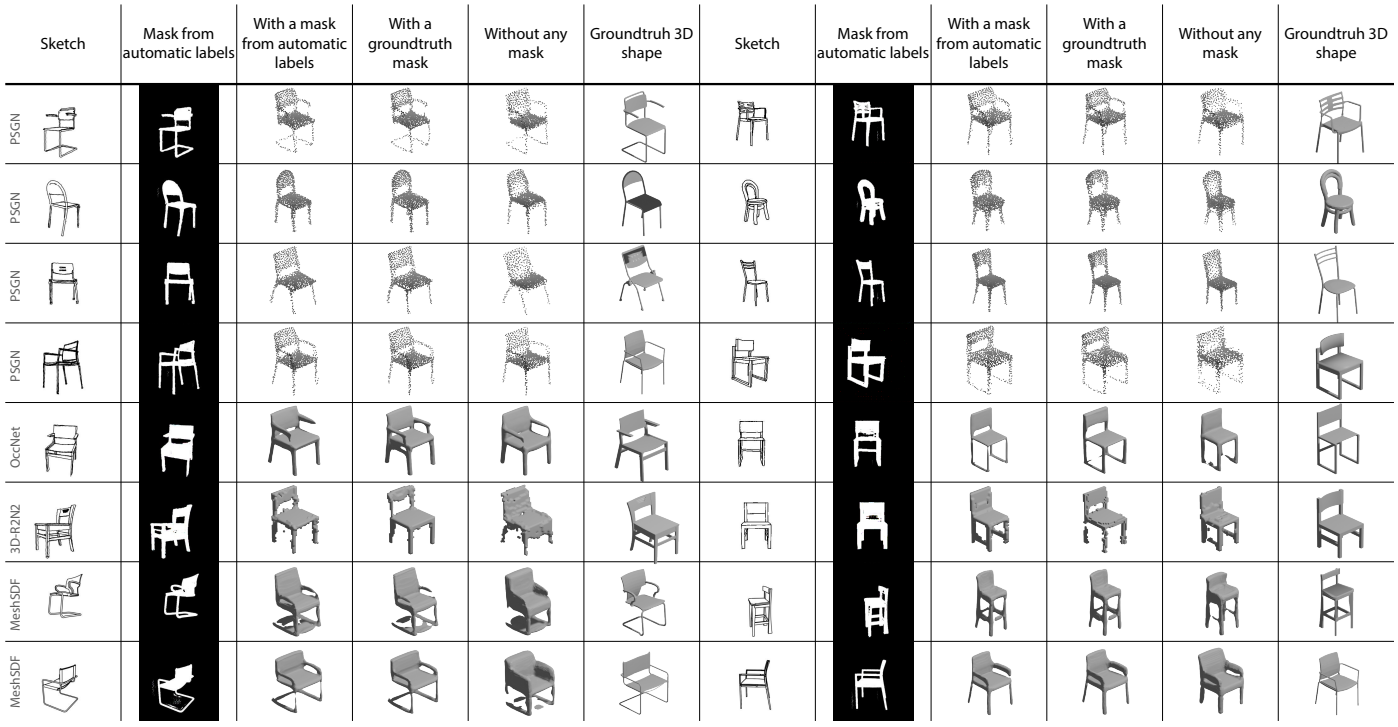

Fig. 10: Visual comparison of reconstruction results with and without an additional input mask: ground-truth or predicted using automatically generated labels.

Let $f^i_{\mathbf{S_A}}$ be an embedding of shape A sketch from some viewpoint i, and $f^j_{\mathbf{S_B}}$ be an embedding of shape B sketch from some viewpoint j. Then, we compute the distances in the embedded space as dot products and convert them into a probability distribution as follows:

$$\hat{p}(f^i_{\mathbf{S_A}}, f^j_{\mathbf{S_B}}) = \frac{\exp(f^i_{\mathbf{S_A}} \cdot f^j_{\mathbf{S_B}})}{\sum_{\mathbf{B}' \in \mathbf{X}'} \exp(f^i_{\mathbf{S_A}} \cdot f^j_{\mathbf{S_{B'}}})}. \quad (4)$$

Then the regression loss is defined as an L_1 loss:

$$\mathcal{L}_R(\mathbf{X}') = \frac{1}{|\mathbf{X}'|} \sum_{\mathbf{A} \in \mathbf{X}'} \sum_{\mathbf{B} \in \mathbf{X}'} \sum_i \sum_j |\hat{p}(f^i_{\mathbf{S_A}}, f^j_{\mathbf{S_B}}) - p(\mathbf{A}, \mathbf{B})|. \quad (5)$$

6.1.1. Loss formulation for SDFs

We consider alternative choices of distance computation and regression loss forms that give better results when 3D shapes are represented as SDFs.

First, we compute the distances in the embedded space as L_2 distances and convert them into a probability distribution as follows:

$$\hat{p}(f^i_{\mathbf{S_A}}, f^j_{\mathbf{S_B}}) = \frac{\exp(-\|f^i_{\mathbf{S_A}} - f^j_{\mathbf{S_B}}\|^2)}{\sum_{\mathbf{B}' \in \mathbf{X}'} \exp(-\|f^i_{\mathbf{S_A}} - f^j_{\mathbf{S_{B'}}}\|^2)}. \quad (6)$$

Second, we define the regression loss as a sum of Kullback-Leibler divergences between the two distributions $p(\mathbf{A}, \mathbf{B})$ and $\hat{p}(f^i_{\mathbf{S_A}}, f^j_{\mathbf{S_B}})$ over neighbors for each object:

$$\mathcal{L}_{KL} = \sum_A \sum_B \sum_i \sum_j p(\mathbf{A}, \mathbf{B}) \log \frac{p(\mathbf{A}, \mathbf{B})}{\hat{p}(f^i_{\mathbf{S_A}}, f^j_{\mathbf{S_B}})}. \quad (7)$$

Note that we can use same shapes $A = B$, but different viewpoint $i \neq j$, or we can use same shapes $A = B$ and same viewpoint but different sketching styles. We also experimented with the Wasserstein metric, but it gives similar performance to the formulation above.

6.2. Choice of distance metric in 3D

The distance d in Eq. (2) computation approach is selected based on the used shape representation. Previously, we introduced the regression loss in the context of the network that works with point clouds [10]. In this paper, we extend the loss formulation to work with shapes represented as SDFs.

Note that while it is theoretically possible to precompute distances between 3D shapes using the original Chamfer distance formulation, this is very computationally expensive. Namely, when the training set consists of N shapes, the runtime of this precomputation is $O(N^2)$. A computation of the Chamfer distance between a pair of shapes on the TITAN X graphics card takes around 0.2 seconds. In particular, the precomputation for the training set of 6,278 will take around 6.5 weeks. Therefore, in this work, we consider the formulation for the shapes represented as SDFs that allows us to compute the distances on the fly. Our training converges in 4 to 5 days, which is much faster than the precomputation process. In addition, computing the distances on the fly provides more flexibility. For instance, new shapes can be easily added without requiring additional precomputations.

To compute σ in Eq. (3), we estimate d_{max} by computing the distances between 3D shapes on a subset of 3,000 3D shapes taking the computation runtime into consideration.

6.2.1. Distance between SDFs

When using SDFs to represent 3D shapes, computing distance d between two ground-truth 3D shapes in Eq. (2) is less straightforward than when point cloud shape representation is used [10]. Let's see what makes it challenging. It was

Table 3: *SDF sampling: Section 6.2.1.* Models with a regression loss are trained implementing Eqs. (6) and (7). *Val.* denotes metric values, and Δ denotes the relative improvement with the regression loss over the baselines: *SDF* – the shape-specific sampling strategy is used, *SDF-Grid* – the sampling is done on a uniform 3D grid. *d(SDF)* denotes that the L_1 distance is computed only on SDF values, and *d(SDF,X,Y,Z)* denotes that the L_1 distance is computed on SDF values and X,Y,Z points coordinates. For the SDF-based baselines, the distances d are computed between the pair of points from the two 3D shapes SDF representations that have the most similar X,Y,Z coordinates.

Method	CD↓		EMD↓		F-Score↑	
	Val.	Δ	Val.	Δ	Val.	Δ
SDF	0.0299	0	0.0296	0	0.1158	0
SDF+Reg d(SDF)	0.0272	-0.0027	0.0265	-0.0031	0.1184	0.0026
SDF+Reg d(SDF,X,Y,Z)	0.0247	-0.0052	0.0243	-0.0053	0.1225	0.0067
SDF-Grid	0.0311	0	0.0304	0	0.1086	0
SDF-Grid+Reg d(SDF)	0.0250	-0.0061	0.0245	-0.0059	0.1223	0.0137

shown that training DeepSDF [40] with a shape-specific sampling (Section 4.1.3) allows achieving more accurate results than sampling ground-truth values on a uniform 3D grid of points. Our experiments confirm this: we evaluate DeepSDF on our sketches when we use either the shape-specific sampling ($525K$ points), as described in Section 4.1.3, or the uniform sampling on a 3D grid with a resolution of 81 along each dimension ($531K$ points). During training, we sample random subsets of 8,192 points for both uniform and shape-specific sampling. Table 3 confirms that the shape-specific sampling results in a better reconstruction accuracy.

Yet, the shape-specific sampling makes it difficult to compare the two shapes represented with SDFs, as it is not meaningful to compare SDF values in arbitrary points. To overcome this challenge, we propose to first find for each point of shape A the spatially nearest point of shape B, and then compute the distance as a sum of L_1 distances between both points' 3D coordinates and SDFs values. Namely, a shape A is represented as a set of M points, where each i-th point p_i is a quadruple consisting of spatial coordinates $A_i \in \mathbf{R}^3$ and an SDF value $S(A_i) \in \mathbf{R}^1$:

$$A = \{p_i^A = [A_i, S(A_i)] \mid i = [1 \dots M]\}. \quad (8)$$

To compute the distance from shape A to shape B, for each point i of shape A, we find the point of shape B with the nearest spatial coordinates according to the L_1 norm: $\bar{j} = \arg\min_j \|A_i - B_j\|_1$. We then compute the distance $d(p_i^A, p_j^B)$ between the two points as a sum of L_1 distances between them:

$$d(p_i^A, p_j^B) = \|A_i - B_j\|_1 + \|S(A_i) - S(B_j)\|_1. \quad (9)$$

We also consider an approach where only L_1 distances between SDF values are taking into account. Table 3 shows that using both spatial coordinates and SDF values results in better performance. It also shows that when the uniform grid sampling is used, the regression loss gives a larger advantage compared to the baseline without the regression loss performance. Nevertheless, the shape-specific sampling combined with the regression loss still gives superior performance over the case when the uniform grid points sampling is used.

Table 4: *Regression loss; choice of the distance in the feature space*: dot product Eq. (4) or L_2 distance Eq. (6); and L_1 loss Eq. (5) vs KL divergence Eq. (7). All methods in this table are trained and tested on stylized synthetic datasets for a 'chair' category. And tested on viewpoints selected as described in Section 4.3.2.

Method	CD ↓	EMD↓	F-Score↑
PSGN	0.0293	0.0407	0.0887
PSGN+Reg Eqs. (4) and (5)	0.0264	0.0371	0.0933
PSGN+Reg Eqs. (6) and (7)	0.0269	0.0382	0.0917
SDF	0.0299	0.0296	0.1158
SDF+Reg Eqs. (4) and (5)	0.0261	0.0253	0.1207
SDF+Reg Eqs. (4) and (7)	0.0254	0.0249	0.1220
SDF+Reg Eqs. (5) and (6)	0.0256	0.0248	0.1218
SDF+Reg Eqs. (6) and (7)	**0.0247**	**0.0243**	**0.1225**

6.3. Regression loss performance analysis

6.3.1. Regression loss formulation

We analyze the performance of PSGN [31] and DeepSDF [40] with and without the regression loss. For PSGN, we use the original formulation, using Eqs. (4) and (5), that we found to work best for PSGN, while for DeepSDF we found that Eqs. (6) and (7) result in better performance, as shown in Table 4.

In Fig. 11, we show reconstruction results obtained from different viewpoints when we use the proposed regression loss and not. The reconstruction results are obtained with the DeepSDF-based baseline. This Figure shows that the regression loss allows to achieve the reconstructions more consistent among different viewpoints.

6.3.2. Feature space analysis

In this section, we analyze the feature space properties when DeepSDF [40] is trained with and without the regression loss. When training with the regression loss we use Eqs. (6) and (7).

Neighborhood relations preservation. To study the role of the regression loss in preserving distances between 3D shapes, we plot in Fig. 12(a) the correlation between the ground-truth SDF distances and the predicted SDF distances. This Figure shows that the regression loss indeed allows to better preserve neighborhood relations: Without the regression loss the distances between the predicted 3D shapes tend to be smaller than between their ground-truth counterparts.

Style-view invariance. To study how efficient the regression loss is in ensuring view and style consistency, we randomly select 200 shapes and 4 sketch viewpoint from 8 viewpoints for each shape. We then compute the distances between the reconstructions from different sketches of the same shape and plot them in Fig. 12(b). The clustering of points closer to zero when the regression loss is used demonstrates the efficiency of our strategy.

View information. By design the regression loss aims to bring the latent vectors of 3D shape sketches from different viewpoints together, while the previous work [8] shows that view information can be used to decrease the ambiguity of sketch input.

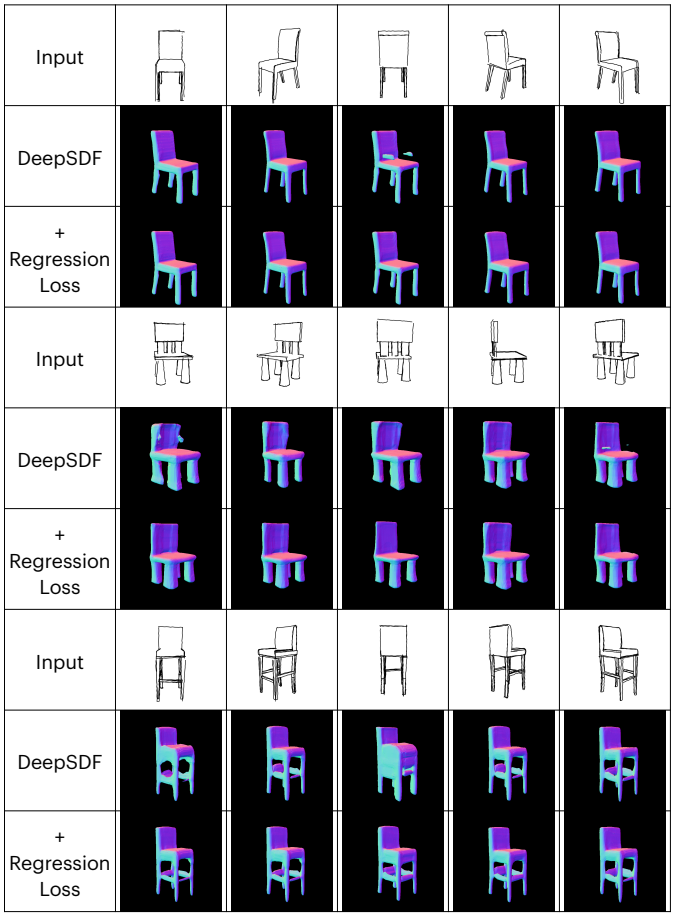

Fig. 11: Comparison of the reconstructions obtained with [40] with and without proposed regression loss Eqs. (6) and (7).

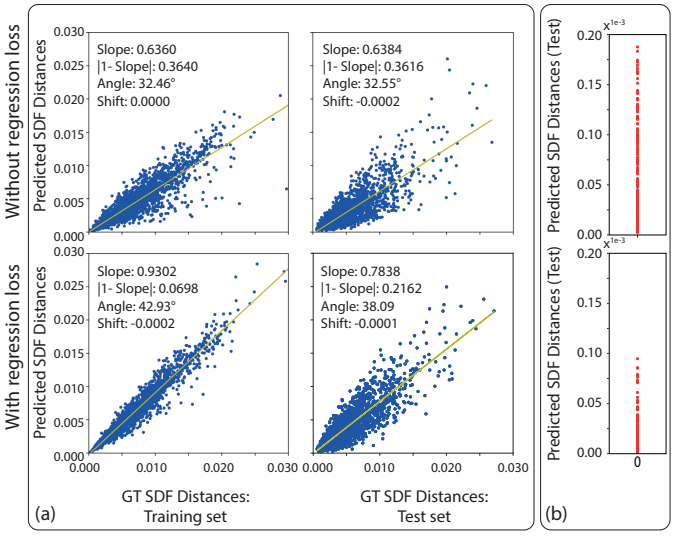

Fig. 12: *Neighborhood relations preservation and style/view invariance: Section 6.3.2.* To generate this Figure, we select a random subset of 200 shapes from the test set of shapes and of 200 shapes from the training set. As the predicted SDF values are defined at the equally spaced 3D grid points, to make this plot, we sample ground-truth SDF values using the same grid points, but the shape-specific sampling is used for training. (a) The x-axis is the L_1 distances between the SDF values of the ground-truth 3D shapes, and the y-axis is the L_1 distances between the SDF values of the estimated 3D shapes from randomly selected sketch-viewpoints of the respective ground-truth 3D shapes. (b) The y-axis is the L_1 distances between the SDF values of the estimated 3D shapes from 4 randomly selected sketch-viewpoints of the same ground-truth 3D shape, for 200 shapes.

Table 5: *Evaluation of the disentanglement of the feature space with and without the proposed regression loss.* +Cam *denotes the usage of the disentanglement to view-invariant and view-specific components of the feature vector.* SDF(N) *denotes that N diverse viewpoints were used for training.*

Method	Chamfer Distance ↓	EMD↓	F-Score↑
SDF(48)	0.0299	0.0296	0.1158
SDF(8)	0.0305	0.0300	0.1147
SDF(8)+Cam	0.0295	0.0291	0.1164
SDF(8)+Reg Eqs. (6) and (7)	0.0249	0.0246	0.1221
SDF(8)+Reg+Cam Eqs. (6) and (7)	**0.0245**	**0.0239**	**0.1230**

First, we study if without the regression loss the feature space has a viewpoint as one of the important cues for the encoder. We select a subset of 500 shapes, and 8 base sketch-viewpoints for each shape. In Fig. 13 (a-b) we visualize the latent feature space of DeepSDF with and without the proposed regression loss using t-SNE, where the feature vectors are 256×1 dimensional. We plot the results for the regression loss when 4 sketches of the same 3D shape in a mini-batch are used. Since the goal of regression loss is to encode different views of the same shape close to each other, the same views of different shapes are expected to be scattered in the feature space. It can be seen that even without the regression loss (Fig. 13 (a)) the feature space does not seem to exhibit any view-specific structure.

Next, we change the architecture, so that the encoder predicts 256×1 dimensional feature vector f_{S_A} for each sketch S_A encoding view-invariant information, and a 16×1 vector $v^i_{S_A}$ encoding view i information. In this architecture, only f_{S_A} is passed to the decoder for the reconstruction. We apply the regression loss only on f_{S_A}. We consider a simplified scenario here and train only with 8 base viewpoints. We also implement a viewpoint decoder $P(\cdot) \in \mathbb{R}^{8\times 1}$ consisting of 3 fully-connected layers to which we pass $v^i_{S_A}$, and which predicts a probability of the current sketch to be from one of 8 base viewpoints. We use the cross entropy loss function to train the network to disentangle feature space: $L_{CE} := -\sum_i p'_i \log(p_i)$, where $p_i = P(v^i_{S_A})$ is the predicted probability vector, and p'_i denotes the ground-truth viewpoint probability vector. Figure 13 (c-d) visualizes the latent feature space (including f_{SA} and v_{SA}) of DeepSDF with and without the proposed regression loss using t-SNE when the described above disentanglement is used. It can be observed that the regression loss results in better disentanglement. Finally, we test the models with the described above disentanglement on our test set that contains sketches randomly selected from 48 viewpoints. Table 5 shows that this disentanglement further allows to increase the accuracy of the reconstruction results.

7. Conclusion

This work extends our conference paper [10] by considering a recently proposed sketch-specific method [7] and the increasingly popular representation of 3D shapes as Signed Distance Fields (SDFs). In this extension, we propose a new formulation of the regression loss fitted to work with SDFs. We consider and evaluate several sampling strategies and distance

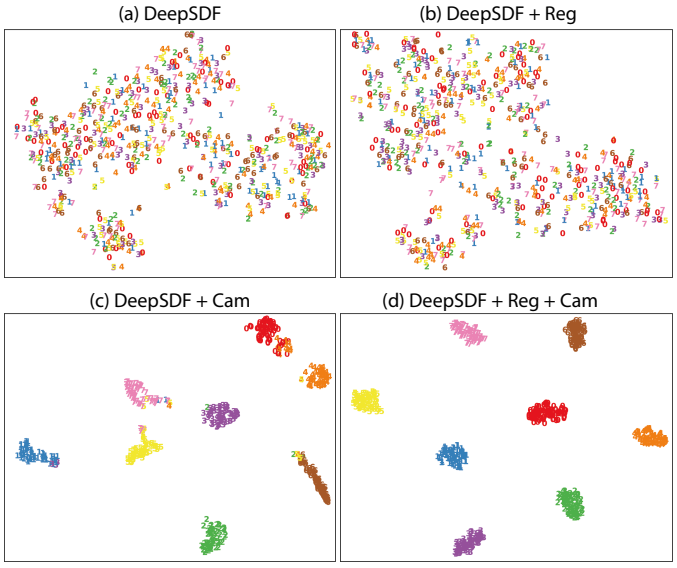

Fig. 13: T-SNE comparison of the latent space distributions: (a) DeepSDF [40], trained with random viewpoints; (b) DeepSDF + regression loss Eqs. (6) and (7) (4-sketches of the same 3D shape in a mini-batch); (c) DeepSDF with disentangled feature space; (d) Same as (c) + regression loss Eqs. (6) and (7). +*Cam* denotes the usage of the disentanglement to view-invariant and view-specific components of the feature vector.
We assign a unique color for each viewpoint and use the number (1-8) to denote a viewpoint.

functions to compare two 3D shapes encoded with SDFs. We also perform a careful analysis of the choice of the parameter for the regression loss. We study the properties of the feature space that the proposed regression loss enables. Furthermore, we show that the accuracy of the reconstruction can be improved when multiple sketches are used for training for the same 3D shape, lifting the limitation on a batch size due to a GPU memory limit. Finally, we show how the regression loss can be combined with the latent space disentanglement for view-invariant and view-specific information allowing to increase the accuracy of 3D reconstructions. We believe that our work will serve as a reference for deep sketch-based modeling and will encourage the future development of dedicated reconstruction networks that take sketch specifics into account. All the datasets and models are available at `https://tinyurl.com/DeepSketchModeling`.

References

[1] Xu, B, Chang, W, Sheffer, A, Bousseau, A, McCrae, J, Singh, K. True2form: 3d curve networks from 2d sketches via selective regularization. ACM Trans Graph 2014;33(4):131.

[2] Gryaditskaya, Y, Hähnlein, F, Liu, C, Sheffer, A, Bousseau, A. Lifting freehand concept sketches into 3d. ACM Trans Graph 2020;39(6).

[3] Bae, SH, Balakrishnan, R, Singh, K. Ilovesketch: As-natural-as-possible sketching system for creating 3d curve models. In: Proceedings of the 21st annual ACM symposium on User interface software and technology. 2008, p. 151–160.

[4] Igarashi, T, Matsuoka, S, Tanaka, H. Teddy: a sketching interface for 3d freeform design. ACM Trans Graph (Proc SIGGRAPH) 2006;:11–es.

[5] Schmidt, R, Khan, A, Singh, K, Kurtenbach, G. Analytic drawing of 3d scaffolds. In: ACM Trans Graph (Proc. SIGGRAPH Asia). 2009, p. 1–10.

[6] Wang, J, Lin, J, Yu, Q, Liu, R, Chen, Y, Yu, SX. 3d shape reconstruction from free-hand sketches. arXiv preprint arXiv:200609694 2020;.

[7] Zhong, Y, Qi, Y, Gryaditskaya, Y, Zhang, H, Song, YZ. Towards practical sketch-based 3d shape generation: The role of professional sketches. IEEE Transactions on Circuits and Systems for Video Technology 2020;.

[8] Zhang, SH, Guo, YC, Gu, QW. Sketch2model: View-aware 3d modeling from single free-hand sketches. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2021,.

[9] Guillard, B, Remelli, E, Yvernay, P, Fua, P. Sketch2mesh: Reconstructing and editing 3d shapes from sketches. In: Proceedings of the International Conference on Computer Vision (ICCV). 2021,.

[10] Zhong, Y, Gryaditskaya, Y, Zhang, H, Song, YZ. Deep sketch-based modeling: Tips and tricks. In: Proceedings of the IEEE Conference on 3D Vision (3DV). IEEE; 2020, p. 543–552.

[11] Lun, Z, Gadelha, M, Kalogerakis, E, Maji, S, Wang, R. 3d shape reconstruction from sketches via multi-view convolutional networks. In: Proceedings of the IEEE Conference on 3D Vision (3DV). IEEE; 2017,.

[12] Gryaditskaya, Y, Sypesteyn, M, Hoftijzer, JW, Pont, S, Durand, F, Bousseau, A. Opensketch: A richly-annotated dataset of product design sketches. ACM Trans Graph 2019;38(6):232.

[13] Bonnici, A, Akman, A, Calleja, G, Camilleri, KP, Fehling, P, Ferreira, A, et al. Sketch-based interaction and modeling: Where do we stand? AI EDAM 2019;33(4).

[14] Loper, MM, Black, MJ. Opendr: An approximate differentiable renderer. In: IEEE European Conference on Computer Vision (ECCV). Springer; 2014, p. 154–169.

[15] Kato, H, Ushiku, Y, Harada, T. Neural 3d mesh renderer. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2018, p. 3907–3916.

[16] Liu, S, Li, T, Chen, W, Li, H. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In: Proceedings of the International Conference on Computer Vision (ICCV). 2019, p. 7708–7717.

[17] Kato, H, Beker, D, Morariu, M, Ando, T, Matsuoka, T, Kehl, W, et al. Differentiable rendering: A survey. arXiv preprint arXiv:200612057 2020;.

[18] Remelli, E, Lukoianov, A, Richter, SR, Guillard, B, Bagautdinov, T, Baque, P, et al. Meshsdf: Differentiable iso-surface extraction. In: Conference on Neural Information Processing Systems (NeurIPS). 2020,.

[19] Tatarchenko, M, Dosovitskiy, A, Brox, T. Multi-view 3d models from single images with a convolutional network. In: Proceedings of the European Conference on Computer Vision (ECCV). Springer; 2016, p. 322–337.

[20] Yao, Y, Schertler, N, Rosales, E, Rhodin, H, Sigal, L, Sheffer, A. Front2back: Single view 3d shape reconstruction via front to back prediction. arXiv preprint arXiv:191210589 2019;.

[21] Nealen, A, Sorkine, O, Alexa, M, Cohen-Or, D. A sketch-based interface for detail-preserving mesh editing. ACM Trans Graph (Proc SIGGRAPH) 2005;:1142–1147.

[22] Li, C, Pan, H, Liu, Y, Tong, X, Sheffer, A, Wang, W. Robust flow-guided neural prediction for sketch-based freeform surface modeling. ACM Trans Graph 2018;37(6):1–12.

[23] Wu, J, Zhang, C, Xue, T, Freeman, B, Tenenbaum, J. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In: Advances in Neural Information Processing Systems. 2016, p. 82–90.

[24] Choy, CB, Xu, D, Gwak, J, Chen, K, Savarese, S. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In: Proceedings of the European Conference on Computer Vision (ECCV). Springer; 2016, p. 628–644.

[25] Girdhar, R, Fouhey, DF, Rodriguez, M, Gupta, A. Learning a predictable and generative vector representation for objects. In: Proceedings of the European Conference on Computer Vision (ECCV). Springer; 2016, p. 484–499.

[26] Liao, Y, Donne, S, Geiger, A. Deep marching cubes: Learning explicit surface representations. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2018, p. 2916–2925.

[27] Wu, J, Zhang, C, Zhang, X, Zhang, Z, Freeman, WT, Tenenbaum, JB. Learning shape priors for single-view 3d completion and reconstruction. In: Proceedings of the European Conference on Computer Vision (ECCV). 2018, p. 646–662.

[28] Lorensen, WE, Cline, HE. Marching cubes: A high resolution 3d surface construction algorithm. ACM SIGGRAPH Computer Graphics 1987;21(4):163–169.

[29] Delanoy, J, Aubry, M, Isola, P, Efros, AA, Bousseau, A. 3d sketching

using multi-view deep volumetric prediction. ACM on Computer Graphics and Interactive Techniques 2017;1(1):1–22.
[30] 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. 2020, p. 1–10.
[31] Fan, H, Su, H, Guibas, LJ. A point set generation network for 3d object reconstruction from a single image. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017,.
[32] Achlioptas, P, Diamanti, O, Mitliagkas, I, Guibas, L. Learning representations and generative models for 3d point clouds. In: Proceedings of the International Conference on Machine Learning (ICML). 2018,.
[33] Gadelha, M, Wang, R, Maji, S. Multiresolution tree networks for 3d point cloud processing. In: Proceedings of the European Conference on Computer Vision (ECCV). 2018, p. 103–118.
[34] Yang, Y, Feng, C, Shen, Y, Tian, D. Foldingnet: Point cloud auto-encoder via deep grid deformation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2018, p. 206–215.
[35] Wang, N, Zhang, Y, Li, Z, Fu, Y, Liu, W, Jiang, YG. Pixel2mesh: Generating 3d mesh models from single rgb images. In: Proceedings of the European Conference on Computer Vision (ECCV). 2018, p. 52–67.
[36] Pan, J, Han, X, Chen, W, Tang, J, Jia, K. Deep mesh reconstruction from single rgb images via topology modification networks. In: Proceedings of the International Conference on Computer Vision (ICCV). 2019, p. 9964–9973.
[37] Mescheder, L, Oechsle, M, Niemeyer, M, Nowozin, S, Geiger, A. Occupancy networks: Learning 3d reconstruction in function space. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2019, p. 4460–4470.
[38] Chen, Z, Zhang, H. Learning implicit fields for generative shape modeling. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2019, p. 5939–5948.
[39] Chen, Z, Tagliasacchi, A, Zhang, H. Bsp-net: Generating compact meshes via binary space partitioning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2020, p. 45–54.
[40] Park, JJ, Florence, P, Straub, J, Newcombe, R, Lovegrove, S. Deepsdf: Learning continuous signed distance functions for shape representation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2019, p. 165–174.
[41] Chibane, J, Mir, A, Pons-Moll, G. Neural unsigned distance fields for implicit function learning. arXiv preprint arXiv:201013938 2020;.
[42] Groueix, T, Fisher, M, Kim, VG, Russell, BC, Aubry, M. A papier-mâché approach to learning 3d surface generation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2018, p. 216–224.
[43] Smirnov, D, Bessmeltsev, M, Solomon, J. Learning manifold patch-based representations of man-made shapes. In: International Conference on Learning Representations. 2020,.
[44] Tatarchenko, M, Richter, SR, Ranftl, R, Li, Z, Koltun, V, Brox, T. What do single-view 3d reconstruction networks learn? In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2019, p. 3405–3414.
[45] Simo-Serra, E, Trulls, E, Ferraz, L, Kokkinos, I, Fua, P, Moreno-Noguer, F. Discriminative learning of deep convolutional feature point descriptors. In: Proceedings of the International Conference on Computer Vision (ICCV). 2015, p. 118–126.
[46] Simo-Serra, E, Iizuka, S, Sasaki, K, Ishikawa, H. Learning to Simplify: Fully Convolutional Networks for Rough Sketch Cleanup. ACM Transactions on Graphics (SIGGRAPH) 2016;35(4).
[47] Isola, P, Zhu, JY, Zhou, T, Efros, AA. Image-to-image translation with conditional adversarial networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2017;.
[48] Zhang, R, Zhu, JY, Isola, P, Geng, X, Lin, AS, Yu, T, et al. Real-time user-guided image colorization with learned deep priors. arXiv preprint arXiv:170502999 2017;.
[49] Su, W, Du, D, Yang, X, Zhou, S, Fu, H. Interactive sketch-based normal map generation with deep neural networks. ACM on Computer Graphics and Interactive Techniques 2018;1(1):1–17.
[50] Uy, MA, Huang, J, Sung, M, Birdal, T, Guibas, L. Deformation-aware 3d model embedding and retrieval. In: Proceedings of the European Conference on Computer Vision (ECCV). Springer; 2020,.
[51] Hinton, G, Roweis, ST. Stochastic neighbor embedding. In: Conference on Neural Information Processing Systems (NeurIPS); vol. 15. Citeseer; 2002, p. 833–840.
[52] Bessmeltsev, M, Solomon, J. Vectorization of line drawings via polyvector fields. ACM Trans Graph 2019;38(1):1–12.
[53] Liu, C, Rosales, E, Sheffer, A. Strokeaggregator: Consolidating raw sketches into artist-intended curve drawings. ACM Trans Graph 2018;37(4):1–15.

Appendix A. Datasets

In this section, we describe the datasets that we use in the experiments in the paper. These are the same datasets as we used in [10], their descriptions here are provided for completeness.

Appendix A.1. Synthetic datasets

We generate three datasets with distinctive styles, which we refer to as naive, stylized and style-unified (Figure A.14).

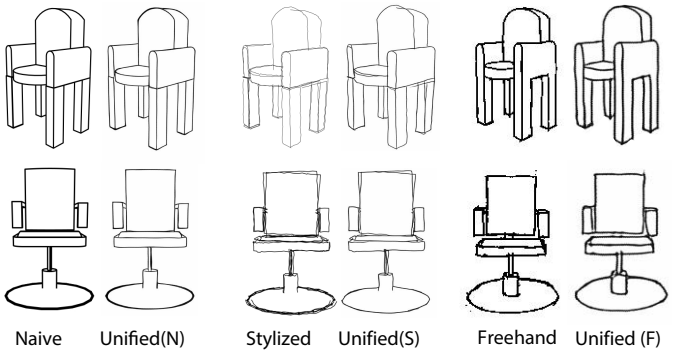

Fig. A.14: Example sketches from our naive, stylized and style-unified synthetic datasets, as well as sketches from ProSketch3D dataset of human sketches.

Appendix A.1.1. Naive

Naive sketch style denotes a synthetic sketch generated from a reference 3D model using the rendering of silhouettes and creases with a uniform stroke width that we set to 2.5. We render such 256×256 sketches using Blender Freestyle[2]. Such types of sketches are clean and contain accurate perspective, and therefore the reconstruction results on them can achieve higher accuracy than on human sketches. Such synthetic sketches differ from human sketches that commonly exhibit perspective and mechanical inaccuracies, as well as oversketching. Despite the existence of multiple recent solutions that aim at converting such rough sketches to cleaner ones [45, 52], these solutions are prone to interpretation mistakes. Thereby, we consider two additional datasets for training: the one that aims at directly mimicking human sketching styles, and the one that represents sketches with the style unified by an additional sketch processing network.

Appendix A.1.2. Stylized

When generating sketches in a naive style, we generate both raster images and vector images with the Blender SVG exporter. To obtain a dataset of stylized sketches, we apply a set of random global and local deformations to each stroke of a naive sketch in a vector format, exploiting the '*svg_disturber*' script from the open-source library[3].

The global stroke deformation consists of stroke rotation, scaling and translation. The rotation angle is randomly sampled from $[0, 2.5°]$. We apply stroke global scaling, which does not preserve the stroke aspect ratio, where the scale factor is randomly sampled from the interval $[0.9, 1.1]$. Finally, the translation vector is randomly sampled from the disk with a 2.5 radius. We enable coherent local noise, where the offset is sampled randomly from the $[0, 1.3]$ interval. In addition, we enable over-sketching, meaning that strokes are traced several times. Each stroke is traced at most two times.

The strokes width varies within a sketch and is randomly sampled from the normal distribution with a mean set to 2.5 and a variance equal to 1.5.

Appendix A.1.3. Style-unified sketch.

Inspired by recent work [6], which deploys an image translation network prior to a 3D reconstruction, and variety of methods aiming at sketch consolidation/simplification/beautification [46, 52, 53], we generate an additional dataset by passing stylized sketches through the fully convolutional sketch simplifying neural network [46] (Figure A.14). We use the model available on the paper's website as is. At inference, each sketch is passed through the same simplifying network, which allows us to unify the style of different sketches.

Appendix A.2. Selected shapes

Most of our experiments are conducted on the models from a chair category of the ShapeNetCore dataset[4], complemented by two additional categories: planes and lamps. We selected these categories guided by the next principles:

Easy to sketch. 3D shape should have a simple structure and should be easy to draw for a human.

Generality. We focus on common categories, that are well familiar to humans. For instance, chairs are common for everyday life, while rifles is an example of a more specific category.

View differentiability. Each shape is expected to have a distinct appearance at distinct viewpoints.

Shape genus higher than 1. We select chairs, which contain 6778 models, due to variability in the level of details and typologies.

Large inter-category variance. We complement chairs with the two other common categories, where airplanes represent shapes with large variability of surface curvatures (4045 models) and lamps contain fine-scaled details (2318 models).

For the training, we use 6278 chair shapes, 1822 lamp shapes and 3548 airplane shapes.

Appendix A.3. Selected viewpoints

For each 3D shape, we first generate sketches for 8 base viewpoints, where the camera elevation is set to 10 degrees to imitate human perspective in the real world, and azimuth takes values equidistantly sampled from 0 to 360 degrees. Following the observations by Gryaditskaya et al. [12] that even professional designers, when asked to sketch from a given viewpoint, produce sketches with large angular deviations from the

[2] https://www.blender.org/

[3] https://gitlab.inria.fr/D3/contour-detect//blob/master/svg_tools/svg_disturber.py

[4] https://www.shapenet.org/

set viewpoint, we generate a set of additional viewpoints to increase method robustness. For each base viewpoint, we generate 5 additional viewpoints by sampling the camera elevation and azimuth angles from normal distributions with a mean matching the elevation and azimuth of one of the base viewpoints. We set variance to 7 degrees. To avoid viewpoints deviating too much from base viewpoints, we add the lower and upper thresholds of 5 and 15 degrees, respectively. The distance between a virtual camera and a 3D shape is randomly sampled from the normal distribution with the mean set to 1.5 in the range [1.4,1.6]. We use a perspective camera for all viewpoints.

Appendix A.4. Human sketch dataset

We exploit a ProSketch3D [7] dataset of professional human sketches of chairs from the ShapeNetCore dataset, which we use only as a test set. The ProSketch3D dataset was collected using an ISKN Slate 2[5] digital drawing tablet. This dataset contains sketches in PNG and SVG formats. This dataset was deliberately designed to contain little style variations and perspective inaccuracies by carefully selecting the participants and providing them with a drawing example, and letting to sketch over the reference viewpoint. Yet, it contains small mechanical inaccuracies non-present in naive sketches and thus presents an interesting test case. Each shape has three viewpoints: front, side and 45°. We use their provided PNG sketches, where the line width rendering roughly matches the 2.5 line width of the sketches in our dataset rendered with a naive sketch style.

Appendix B. Reconstruction Baselines

We study a number of single image reconstruction baselines on sketch input from our synthetic and human sketch datasets. We extend the baselines considered in [10] and add a recently became popular representation of 3D shapes with signed distance functions [40], as well as recently proposed [7] that predicts multi-view 3D shape representation. For completeness, we summarize below all the reconstruction methods that we evaluate in this work, which we classify as view-based and volume-based explicit or implicit, depending on the 3D shape representation used.

View-based. We selected a recent approach by Lun et al. [11], to which we further refer as *ShapeMVD*, dedicated to depth and normal maps predictions from input sketches, which are then merged into a dense point cloud. The method allows fine-tuning if the camera parameters are known, by rendering the reconstructed mesh and smoothly deforming the mesh so that the rendered contour matches the input sketches. We do not use this step, since in a general setting the viewpoint for a human sketch is unknown. In the original paper, the model was trained with two input sketches: the two orthographic projections of the shape from the front and from the side views with clean uniform line-rendering. Here we retrain their architecture with our datasets, using only a single perspective sketch as an input. In addition, we evaluate the recent extension of [11] to sketch-based modeling task by Zhong et al. [7].

[5] https://www.iskn.co/uk/

Volume-based, explicit. Voxels: We evaluate the classic *3D-R2N2* [24] for the task of single view sketch-based 3D shape reconstruction.

Point clouds: As a method working with the point-cloud shape representation, we exploit the popular *PSGN* [31]. We use their vanilla encoder-decoder architecture with the Chamfer distance as a reconstruction loss.

Mesh: We evaluate two baselines that directly predict a mesh: *Pixel2Mesh* [35], which is commonly used as a comparison baseline, and *TMNet*, an approach by Pan et al. [36], which we select due to its design aiming to support the reconstruction of 3D shapes with genus greater than 0.

Volume-based, implicit. We evaluate *OccNet* that predicts per point occupancy and *DeepSDF* that represents 3D shapes using signed distance fields (SDFs).

Appendix B.1. Evaluation metrics

We use three evaluation metrics to compare the predicted 3D shapes to the reference shapes: Chamfer distance [31], Earth mover's distance [31] and F-Score [44].

Chamfer distance (CD). The Chamfer distance measures the squared distance between each point in one point set to its nearest neighbor in the other set:

$$d_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2,$$

where $S_1 \subset \mathbb{R}, S_2 \subset \mathbb{R}$ are two subsets of points. We compute Chamfer distance by sampling uniformly 2048 points from the predicted meshes and $100K$ points from the reference (this set is fixed for all baselines).

Earth mover's distance (EMD). The EMD is the solution of the optimization problem that aims at transforming one set to the other:

$$d_{EMD}(S_1, S_2) = \min_{\phi: S_1 \to S_2} \sum_{x \in S_1} \|x - \phi(x)\|_2,$$

where $\phi : S_1 \to S_2$ is a bijection, and the two sets $S_1, S_2 \subseteq \mathbb{R}^3$ are of equal sizes. Due to the limited capacity of a GPU, we sample 2048 points from both the predicted meshes and the reference to compute this measure.

F-Score. Tatarchenko et al. [44] observed that the Chamfer distance is sensitive to outliers and proposed to use F-score for comparison of two point clouds. The F-score is defined as a harmonic mean between precision and recall. Precision and recall count the percentage of the points in one set for which there is a point in the other set within a distance threshold, set to 0.01 in our experiments. The ground-truth shapes are normalized to have the largest dimension of 1.

Appendix C. Multi-view sketching

Here we study the effect of providing multiple sketches for each shape, using SDFs as a 3D shape representation. We concatenate multiple sketches along their channel dimension. For example, 1 sketch input is a $224 \times 224 \times 1$ dimensional vector, and 3 sketches input is a $224 \times 224 \times 3$ dimensional vector.

Table C.6 shows the comparison of reconstruction accuracy when we vary the number of sketches used for each 3D shape. Each batch has 8 different 3D shapes. It shows that the more viewpoints are used for each 2D shape, the more accurate are the reconstruction results.

Table C.6: Quantitative evaluation of the accuracy of the reconstruction results when different number of sketches is used for each 3D shape in a mini-batch (Appendix C).

Number of viewpoints	**Chamfer Distance ↓**	**EMD↓**	**F-Score↑**
1	0.0247	0.0243	0.1225
2	0.0242	0.0237	0.1236
3	0.0239	0.0233	0.1241
4	0.0237	0.0230	0.1245
5	**0.0236**	**0.0228**	**0.1248**

Appendix D. Feature space analysis

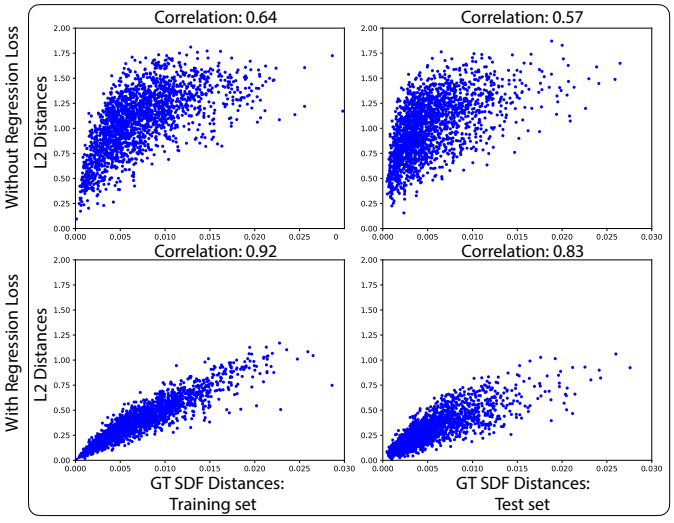

Fig. D.15: *Analysis of correlations of pairwise distances in the latent space and pairwise distances between input 3D shapes.* To generate this figure, we select a random subset of 200 shapes from the test set of shapes and of 200 shapes from the training set. To make this plot, we sample ground-truth SDF values using the same grid points, but the shape-specific sampling is used for training. The y-axis is the L_2 distances in the space of sketches embeddings, and x-axis is the L_1 distances between the SDF values of corresponding 3D shapes. For each subplot we compute the Pearson correlation coefficient, which is shown above each subplot.

We plot the correlation between L_2 distances in the latent space and L_1 distances of GT SDFs in Fig. D.15. It shows that the regression loss brings the embeddings of the sketches of similar shapes closer together and pushes the embedding of different shapes further apart. In addition, for each subplot, we calculate the Pearson correlation coefficients, which show that the regression loss increases the correlation between the pairwise distances of sketch encodings and the pairwise distances of 3D shapes.