Sketch2Shape: Single-View Sketch-Based 3D Reconstruction via Multi-View Differentiable Rendering

Robin Borth, Daniel Korth

Abstract

In recent years, the creation and manipulation of 3D objects have gained prominence in numerous sectors, from entertainment over interior design to manufacturing. Building on the universal nature of sketches, we present a pipeline that can generate 3D objects from 2D sketches. Our approach uses an encoder to map a single-view hand-drawn sketch into the DeepSDF latent space. From there, we leverage differentiable sphere tracing on silhouette maps to refine and improve the 3D shape. Our method outperforms a retrieval baseline and allows for interactive editing of the shape during the generation process. Code is published at https://github.com/robinborth/sketch2shape.

1. Introduction

Sketches are one of the most fundamental and easy ways to draw real-world objects. Humans effortlessly interpret simple sketches to envision 3D objects, showcasing their intuitive understanding of the world. In contrast, sketch-based reconstruction faces challenges due to the absence of any geometric information. Furthermore, ambiguity in sketches with respect to different views [11] makes accurate 3D reconstruction difficult. Approaches include predicting the view [11] or specifying it as input [12], though this may hinder creativity. Another challenge is the domain gap between computer-generated and hand-drawn sketches [7,11], due to the absence of a high-quality hand-drawn sketch dataset.

We present an approach to generate shapes by leveraging a combination of a view-agnostic encoder that maps sketches into DeepSDF latent space and a differentiable rendering pipeline based on multi-view silhouettes to optimize the DeepSDF latent vector at inference. The view-agnostic encoder should make it easier for artists to not restrict their creativity by constraining them to sketch from a specific view or provide exact view information of the sketch. The generated multi-view silhouettes allow us to optimize and refine invalid shapes at inference. As a result, our method enables artists to interactively edit and manipulate the silhouettes during test-time optimization from both single- or multiple views.

2. Related Work

Sketch-based modeling has been an active topic of research for a long time [10]. Recently, neural architectures dominated the field, using an encoder-decoder architecture to model 3D shapes. **Sketch2Model** [11] uses two different latent spaces for view and shape to resolve the view ambiguity in hand-drawn sketches. **Sketch2Mesh** [7] extracts contour lines of sketches or also real drawings and optimizes the mesh using differentiable rendering. This also allows them to edit shapes after generation. **SketchSampler** [6] translate sketches to more useful representation and then sample density maps and lift the representation to 3D by predicting the z-values of a density map. Given the recent advancements in diffusion models, researchers have also started adapting them to 3D and sketch models. **LAS-Diffusion** [12] introduces a two-step diffusion process to first diffuse the coarse occupancy grid and then refine the sketch on the higher resolution by SDF-diffusion.

3. Method

We consider a single-view sketch of an object as input and want to generate a full 3D representation. To achieve this, we use three main components. (1) A DeepSDF auto-decoder to represent our 3D shapes. (2) A view-agnostic encoder that maps a sketch from an unknown view into the latent space of DeepSDF. This latent can serve as an effective initial point for optimization. (3) A differentiable rendering module that allows for test-time optimization between the 2D sketch and 3D shape. In particular, we first generate consistent multi-view silhouettes of the input sketch and use them to further optimize our latent vector. Figure 1 shows an overview of our method.

3.1. Auto-Decoder

We use the popular DeepSDF auto-decoder formulation to represent the 3D shape. Formally, let $\mathcal{D}_\theta$ denote the auto-decoder, $c \in \mathcal{R}^3$ a coordinate in 3D space, and z_i a latent vector for shape i. Then, the auto-decoder learns to map 3D coordinates to signed distance values. The surface is then defined by the isosurface $\mathcal{D}_\theta(\cdot) = 0$. In particular, we use the variant Curriculum DeepSDF [4] which yields better

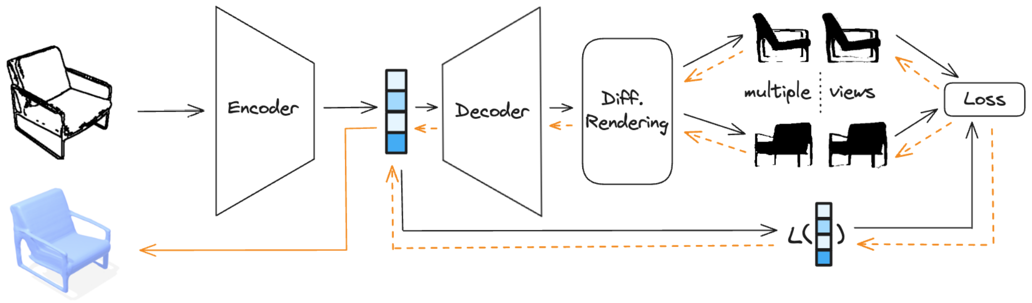

Figure 1. Description of our architecture at inference. Input: single-view sketch. (1) We pass the sketch through our view-agnostic encoder to get an initial latent. (2) We use the DeepSDF auto-decoder & differentiable sphere tracing to optimize the latent vector on multi-view silhouettes. Once the final latent vector is found, we can obtain our shape by extracting SDF values on a grid structure and running the marching cubes algorithm.

results for thin geometry.

3.2. View-Agnostic Encoder

In our setting, only a 2D sketch and no additional view information are provided. However, for free-hand sketches, we cannot guarantee that they are drawn from a single predefined view since (1) users might not be able to sketch a shape perfectly from a specified view, and (2) views might obscure some details important for the artists. Therefore, we train a view-agnostic encoder into the DeepSDF latent space. Let $\mathcal{E}_\phi$ be the encoder that maps the input sketch x_i to its corresponding latent vector z_i. We train it to minimize the L1 distance:

$$\mathcal{L}_{\text{enc}} = \sum_{i=0}^{N} \sum_{v \in \mathcal{V}} ||\mathcal{E}_\phi(x_i^v) - z_i||_1$$

where $\mathcal{V}$ is the number of different views per shape and x_i^v represents the sketch input i from view v.

3.3. Differentiable Rendering

In general, the encoder already gives us a globally correct shape. However, imperfections in the latent code and signed distance values for sphere tracing can lead to missing surfaces - especially for thin geometry. Therefore, we want to optimize our shape through differentiable rendering to fix the aforementioned issue at inference. Our goal is to get a silhouette map of the sketch. We start with our latent vector $z^0 = \mathcal{E}_\phi(x)$. We run sphere tracing and store points p with corresponding minimal distance s to the surface for every ray. Afterwards, we calculate the normal $n = \frac{d\mathcal{D}_\theta(z,p)}{dx}$. If s is negative, we know that there is a surface along the ray. If s is positive but within an threshold to the zero-crossing, we project points to the surface by taking a step into the negative normal direction $p' = p - n$ and then backproject the points to the image coordinates. This process can lead to noise due to imperfect normal estimates, which we alleviate by running a Gaussian blur filter on the image and removing values below a threshold ϵ. The result of this process are silhouettes $\mathcal{S}$ that can be seen as ground-truth data for the following optimization. This process can be applied to arbitrary views, which are consistent with one another and make optimization more stable. Figure 2 depicts the multi-view silhouettes of a specific sketch input.

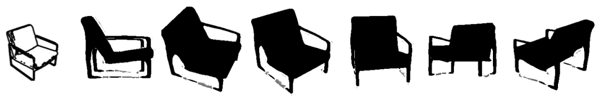

Figure 2. Multi-View silhouettes of the sketch on the left.

Once we calculate the silhouette map $\mathcal{S}$ from different views, we can use differentiable rendering to optimize the latent vector z of the test object.

$$\mathcal{L}_{\text{sil}} = \mathcal{S} max(0, \mathcal{S}_r - \epsilon) + (1 - \mathcal{S}) max(0, -(\mathcal{S}_r - \epsilon))$$

where ϵ the threshold value for hitting the surface and $\mathcal{S}_r$ the silhouette rendered via differentiable sphere tracing. The loss function allows the generation of missing surfaces and the removal of undesired ones.

Additionally, we need to regularize the latent vector to stay in valid latent space. We assume that the initial predicted latent vector z^0 is already a good guess and that the test-time optimization should not deviate too far from it. Thus, our reguarization loss is the L1 distance with $\mathcal{L}_{\text{reg}} = ||z^0 - z^n||_1$. Our final loss for test-time optimization then becomes

$$\mathcal{L} = \sum_{k=1}^{K} \mathcal{L}_{\text{sil}}^k + \mathcal{L}_{\text{reg}}$$

where K is the number of different views of the silhouette.

4. Results

Dataset. ShapeNetCore is a subset of the full ShapeNet [2] dataset with 3D shapes and manually verified category and alignment annotations. We base our experiments on the chair class and create a train/val/test split with 4096/128/256 shapes. To obtain training data from the shapes, we render the chairs from multiple views and extract sketches by running a canny edge detector [1]. Additionally, to generate more training data, we traverse the DeepSDF latent space to generate new valid chairs with valid latent vectors. We end up with 400k unique latent vectors and 800k sketches for training. Figure 3 shows the sketches and how the latent traversal affects the chairs. To evaluate our method on hand-drawn sketches, we follow [5] and create a hand-drawn sketch dataset for the test set.

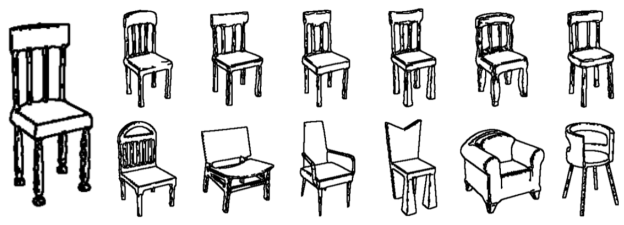

Figure 3. Traversal Dataset. Interpolations between the seed chair (left) and the target chairs (bottom) to create new chairs (top).

Baseline. We train a siamese neural network [9] using triplet loss [3] on the sketches to obtain a retrieval baseline for our method. At inference, we encode the sketch into the embedding space and retrieve the ground truth (GT) mesh of the closest shape in the embedding space.

Evaluation Metrics. We follow standard evaluation metrics from previous works [6, 7, 12]. We use four different metrics to evaluate our method: Chamfer Distance (CD), Earth Mover Distance (EMD), Fréchet inception distance (FID), and CLIPScore [12]. For CD and EMD, we sample 30k and 4k points, respectively, from the GT and reconstructed mesh. The units of CD and EMD are 10^{-3} and 10^{-2}. To evaluate FID and CLIPScore, we render from 10 different views and compare the ground truth sketch image with the rendered sketch image.

Training Details. Curriculum DeepSDF is trained for 17h 23m 33s on NVIDIA GeForce RTX 2080 Ti. The pretrained ResNet18 encoder [8] is trained for 5h 57m 14s on NVIDIA GeForce RTX 2080 Ti.

4.1. Synthetic Sketch Reconstruction

Method	CD↓	EMD↓	FID↓	CLIPScore↑
Retrieval	10.43	11.23	**33.82**	93.96
Encoder	4.40	**8.17**	46.53	93.99
+ Silhouette	**4.17**	8.30	46.78	**94.10**
+ Global	4.17	8.31	46.68	94.08

Table 1. Quantitative evaluation for synthetic sketches.

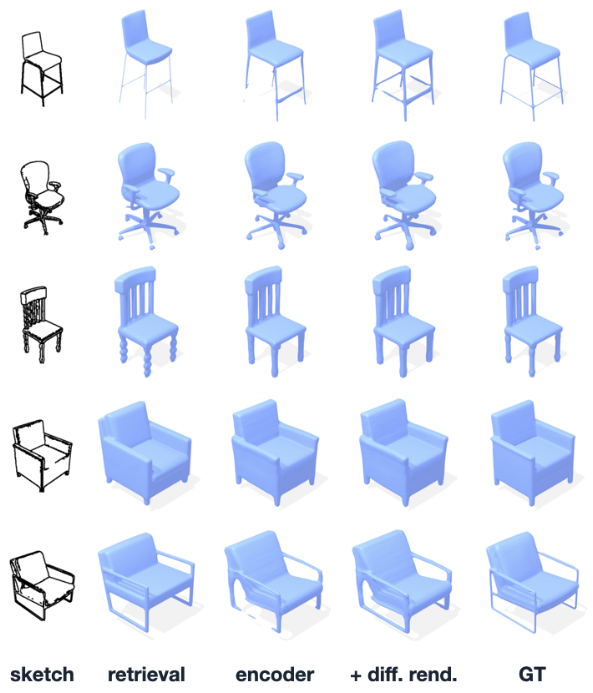

Figure 4. Qualitative results for synthetic sketches.

We evaluate how well our method can generate 3D shapes based on the synthetic sketches used during training. In the qualitative results 4, we observe that both our retrieval baseline and the encoder can reconstruct the input sketches well. Using differentiable rendering allows us to recover some of the missing geometry. Quantitative results are shown in Table 1. Our encoder demonstrates superior performance over the retrieval baseline on metrics such as CD and EMD, indicating enhanced preservation of global structure and improved generalization to novel shapes. Despite this, there is no noticeable improvement in CLIPScore, implying that the encoder consistently generates perceptually coherent shapes. However, it is worth noting that the retrieval baseline outperforms our encoder on FID. We hypothesize that this could be attributed to the fact that our retrieval process retrieves GT meshes, consequently leading to the extracted sketches aligning closely with the distribution of ”sketches” used for evaluation. As a result, FID is more sensitive to this alignment than CLIP. While we observe qualitative differences, differentiable rendering based on silhouettes does not improve upon the encoder quantitatively. We attribute this to two main reasons: (1) For the majority of the chairs, the encoder already gives good results close to the ground truth, leading to no performance gains. (2) Metrics like CD and EMD do not give large importance to small or thin structures for which silhouette loss was designed. We also experiment with a global loss, which calculates the distance between the sketch and rendered images, based on our encoder. However, this does not lead to any performance gains.

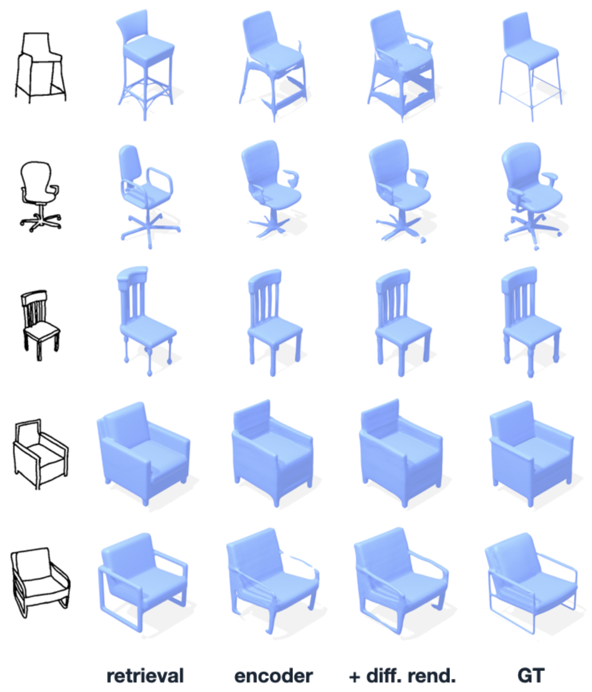

Figure 5. Qualitative results for hand-drawn sketches.

Method	CD↓	EMD↓	FID↓	CLIPScore↑
Retrieval	14.29	13.41	**37.14**	92.82
Encoder	8.99	11.42	50.04	92.86
+ Silhouette	**8.59**	**11.21**	49.60	**93.14**
+ Global	8.60	11.23	49.64	93.12

Table 2. Quantitative evaluation for hand-drawn sketches.

4.2. Free Hand-Drawn Reconstruction

We evaluate our method on hand-drawn sketches to simulate real-world use cases. Quantitative results can be found in Table 2, and qualitative results in Figure 5. In general, we observe a large domain gap between synthetic and hand-drawn sketches, which is reflected in the drop in performance when comparing the results of the previous section. Results behave similarly to synthetic sketches. However, incorporating differentiable rendering can enhance the performance of the encoder baseline by a small margin. This improvement stems from the increased potential for refinement, allowing the silhouette loss to effectively guide the encoder towards a closer alignment with the ground truth.

4.3. Multi-View Robustness

A crucial characteristic of our encoder is that we train it to be view-agnostic. Figure 6 shows qualitative results of

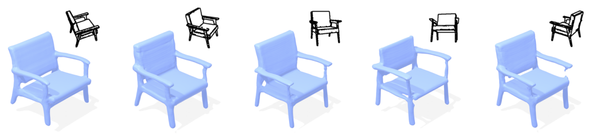

Figure 6. Qualitative evaluation of view robustness.

this property. While different views may result in approximately similar shapes, achieving perfect consistency across all views remains unattainable. Due to the inherent ambiguity in sketches, it remains uncertain whether perfect view-robustness is feasible.

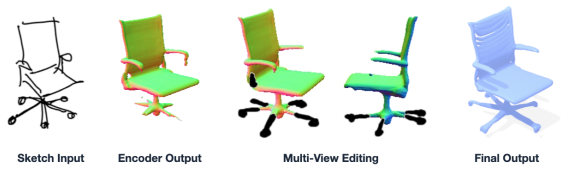

Figure 7. Real-time editing of the silhouettes. From left to right, you see the input sketch and the initial reconstruction of the sketch. Since the bottom of the chair is not correct, we edit the silhouette (from multiple views) and run differentiable rendering. The final output fixes the bottom of the chair.

4.4. Interactive Editing

Our approach allows for live editing by manipulating the silhouettes after the initial shape has been generated. We accomplish this by developing an interactive painting tool that allows artists to paint over the silhouette to either add or remove geometry or introduce new features not originally present. The modified silhouette can be utilized for differentiable rendering. This editing functionality is applicable for both single and multiple views; see Figure 7.

5. Conclusion

We studied the task of 3D reconstruction from 2D sketches. Our approach, based on encoding a sketch into the DeepSDF latent space and optimizing the latent during inference, presents a flexible way to interact with the shape via live editing. While our encoder significantly outperforms the retrieval baseline, the differentiable rendering module can only further improve the shape in a few circumstances where the global structure is preserved and thin structures are missing.

We identify the following areas of future work. Shape Diversity: Currently, our method is restricted to only chairs; enabling training on multiple types remains open for further exploration. Domain Gap: Due to the diversity of hand-drawn sketches and the lack of high-quality datasets, it remains unclear how to close the gap between hand-drawn and synthetically generated sketches.

References

[1] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8:679 – 698, 12 1986. 3

[2] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 2

[3] Xingping Dong and Jianbing Shen. Triplet loss in siamese network for object tracking. In *Proceedings of the European conference on computer vision (ECCV)*, pages 459–474, 2018. 3

[4] Yueqi Duan, Haidong Zhu, He Wang, Li Yi, Ram Nevatia, and Leonidas J Guibas. Curriculum deepsdf. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VIII 16*, pages 51–67. Springer, 2020. 1

[5] Mathias Eitz, James Hays, and Marc Alexa. How do humans sketch objects? *ACM Transactions on graphics (TOG)*, 31(4):1–10, 2012. 3

[6] Chenjian Gao, Qian Yu, Lu Sheng, Yi-Zhe Song, and Dong Xu. Sketchsampler: Sketch-based 3d reconstruction via view-dependent depth sampling. In *European Conference on Computer Vision*, pages 464–479. Springer, 2022. 1, 3

[7] Benoit Guillard, Edoardo Remelli, Pierre Yvernay, and Pascal Fua. Sketch2mesh: Reconstructing and editing 3d shapes from sketches. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13023–13032, 2021. 1, 3

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 3

[9] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015. 3

[10] Luke Olsen, Faramarz F Samavati, Mario Costa Sousa, and Joaquim A Jorge. Sketch-based modeling: A survey. *Computers & Graphics*, 33(1):85–103, 2009. 1

[11] Song-Hai Zhang, Yuan-Chen Guo, and Qing-Wen Gu. Sketch2model: View-aware 3d modeling from single free-hand sketches. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6012–6021, 2021. 1

[12] Xin-Yang Zheng, Hao Pan, Peng-Shuai Wang, Xin Tong, Yang Liu, and Heung-Yeung Shum. Locally attentional sdf diffusion for controllable 3d shape generation. *arXiv preprint arXiv:2305.04461*, 2023. 1, 3