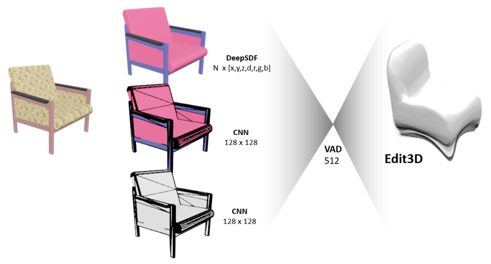
Cross Modal Shape Generation Introspection

Yejoon Jung   Joseph Svrcek   Hongyu Tu

Abstract

New techniques to generate 3D shapes have continued to grow. DeepSDF created a way to generate a mesh from points or features using neural nets, and from there teams have continued to build on the research. Different teams came up with DualSDF and then a cross-modal generative technique integrating a variety of prior models like DeepSDF and Variational AutoDecoders, in a project called Edit3d. While it had great performance there were several shortcomings which they documented in their paper. Those were model resolution, color recreation, and classifying specific subclasses of models (i.e.a type of chair). In this paper we attempt to recreate their results and improve upon their methods.

1. Introduction



In this paper we investigate a new technique for generating 3d models by combining 3d models and 2d images [1]. The technique uses the 3d shapenet dataset and reuses techniques first pioneered in the so-called DeepSDF model and DualSDF. The authors also used 2d datasets to supplement the models. The goal is to be able to use newly created models based on user sketches. Furthermore, the models would be able to be edited by adding additional input either by coloring the original sketch, or by altering the sketch with new lines. There are several challenges that the Edit3d model could not overcome. In this paper we explore various ways to try to solve those original shortcomings.

2. Related Work

As indicated in the original paper there are other works which investigate this domain. The notable efforts are Sketch2Mesh [3], DualSDF [4], and EditNeRF [6] as well as others [7, 9, 10]. Sketch2Mesh also relied on SDFs, and produced a shape which closely approximated the input sketch. Also based on DeepSDF [8], the DualSDF project takes a novel approach to manipulating the meshes by using spheres as reference points. The last comparison is EditNeRF, which uses neural radiance fields to generate the models. The paper we covered is based on DualSDF, and enhances the dataset using per pixel color values.

The result is supposed to 3d models which can be edited using additional sketch or color scribble inputs. The authors of the paper showed performance exceeding the prior work, while able to generate the models using a single image representation.

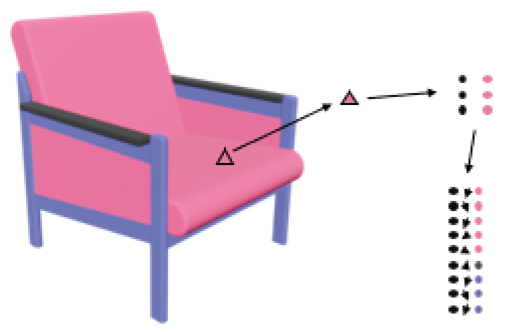
3. Method

3.1. Training Data

We spent a large portion of the project time on implementing loaders for the training data. The project code for edit3d came with pretrained models, but the pipeline for preprocessing the data did not exist. The first author did provide us with a dataset used to train the first model, and gave some advice for creating a preprocessing pipeline. THe dataset is trained using models from shapenet. Since the original papers used airplanes and chairs, we went with a subset of chairs to focus our pipeline.

3.2. Data Representation

Shapenet data is distributed as normalized obj files with materials. An ASCII list of vertices, faces, and materials, where a material is a texture representation for specific faces. In order to correlate color to a location on the mesh we need to embed a representation to that mesh location. We do this by creating a super sampled signed distance function using some helper code from DualSDF. We assign each SDF value an RBG value.



We choose the values based on whatever materials that were already assigned. This has the benefit of relying on the model creators input on which parts of the mesh are related. The drawback is that this can cause poor results when the materials are missing, inaccurate, or too general.

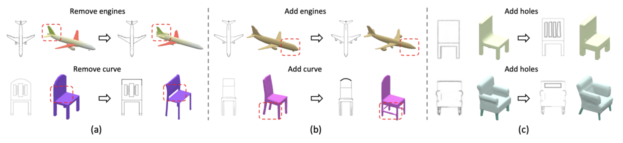
In order to create two dimensional representations of the objects we use blender to render a color image. To generate sketches we replace the color meshes with white, and use the line art feature of blender. This works reasonably well for certain objects but very detailed meshes will sometimes result in an unnatural amount of line strokes. Then we pass the resulting render through a binary filter just to ensure it is black and white.

3.3. Improvement Attempts

The original authors trained with a significant amount of data samples and training iterations. While we were able to reproduce similar results to the authors, due to time constraints and limits on technical resources, we were not able to reproduce results of the same quality. Investigating the problem set our focus tended to be around two main data points. The first being to investigate the initial data representation to resolve color issues and to add noise to prevent overfitting.

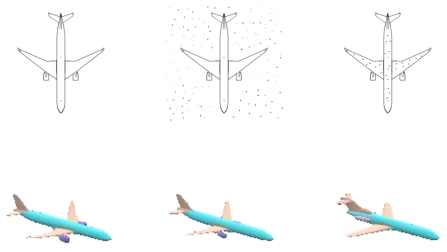
3.3.1 Editing via sketch

As shown in the image, when making changes to the sketches, the parts that should not be involved in the change got changed as well. From an optimization point of view, we propose to fix it by including terms that will contribute to the loss function which punishes unwanted change, as a form of extra level of restriction.



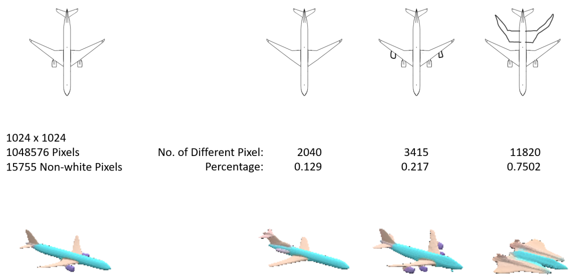
Throughout this project, three attempts were made and will be discussed in more detail below:

1. Ignoring tiny differences
   The first attempt to minimize unwanted changes is to have the model ignore tiny changes. From the previous work, when provided with two similar sketches with only extra random noise added, the model will produce two different models.This shows the lack of consistency in the model. To fix this problem, for each entry, random noises will be added to the input and the output will still be compared against the expected output. This encourages the model to maintain the max level of similarity between images when the difference is not significant. Achieving this is relatively simple too, all we need is to add the code for randomly changing the original input and record the loss in each training iteration.



2. Limiting changes with respect to portion
   Besides discouraging the model from making tiny changes to the 3D model, we don't want the model to not make any changes at all. For all cases where the changes are not technically tiny, we still want the model to change the 3D object proportionally to the amount that the 2D has changed. More specifically, for all entries, the difference between 2D images will be recorded and so will the difference between 3D shapes. Because we want the ratio of those two changes to be as close as possible, the loss function was set to RMSLE (Root Mean Squared Logarithmic Error), where y is the 2D change rate (between 0 and 1) and $\hat{y}$ is 3D change rate, between 0 and 1 as well.

$$loss_{portion} = \sqrt{((\log(y_i + 1) - (\log(\hat{y_i} + 1))^2}$$



3. Understanding Context
   Fundamentally, the reason why the output is not consistent because the model isn't really doing editing but instead regenerating. Therefore, there's no connection between the two shapes, which is essential for the shape to stay consistent. With the advance in computer vision and graphics, there are two papers that became useful for solving this task: "Segment Anything Model" by Meta [5] and "3D Highlighter" by Decatur [2]. Given an edited image, SAM can be used to identify the word for the part that was changed, for example, engine as shown below. Then, feed the word prompt to 3D highlighter as well as the 3D shape, it will be able to identify the engine on the airplane. Lastly, with some wrappers, the highlighted part was removed and no changes will be added to other parts of the plane.

With Prompt: "Engine"

3.3.2 Improving Editing Via Scribble

Recoloring the 3D surface by applying color scribbles on the 2D rendering offers a fast and convenient method to alter existing shapes, without the need for extensive knowledge of 3D editing tools or training new models. This color manipulation process involves adjusting the initial latent variable z_0 to align with a specific modification represented by e^M.

Given 2D control $\mathcal{G}^M(z)$ of modality $M \in \{S, R\}$, optimal updated latent is given by

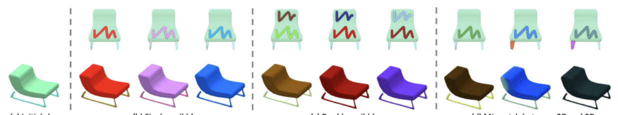
$$\hat{z} = \underset{z}{\operatorname{argmin}} \mathcal{L}\_edit(\mathcal{G}^M(z), e^M)$$

The paper also includes a regularization term $\mathcal{L}_{\nabla]\}}(z)$, which promotes the latent $\hat{z}$ to stay in the prior of VAD. Thus, the complete optimization problem for editing is given by

$$\hat{z} = \underset{z}{\operatorname{argmin}} \mathcal{L}\_edit(\mathcal{G}^M(z), e^M) + \mathcal{L}_{\text{reg}}(z)$$

Where

$$\mathcal{L}_{\text{reg}}(z) = \gamma \max(||z||_2^2, \beta)$$



However, as depicted in the diagram, the current model is unable to produce outcomes that lie outside of the pre-established latent distribution. For instance, when introducing two distinct scribbles to a chair that features a seamless boundary between its parts, the model erroneously extends one of the scribbles to an unintended section or causes a blending of two colors. In order to address this issue, we explored two different approaches.

1. Re-tuning Hyperparameters The regularization term $\mathcal{L}_{\text{reg}}(z) = \gamma \max(||z||_2^2, \beta)$ encourages the resulting latent to be generalized over the shape, and γ is a hyperparameter that controls this amount. The default value of γ is set to 0.02 in the existing model, and we observed results produced from each γ value of 0.001,0.02,0.1.

2. Taking area outside scribble into account The term $(\mathcal{G}^M(z), e^M)$ only takes account of the area of scribbles, which is prone to undesired changes of parts without scribbles. Thus, we added a term that will encourage areas outside of scribbles to remain as initial colors. A new loss function $\mathcal{L}_{edit}$ is given by

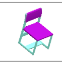
$$\mathcal{L}\_edit = ||\mathcal{G}^M(z) - e^M|| + \lambda||\mathcal{G}^M(z\_0) - \mathcal{G}^M(z)||$$

   where $\mathcal{G}^M(z_0)$ is initial 2D control of modality, and controls amount of contribution of area outside scribbles.

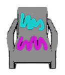
   We observed results produced from each λ value of 0,0.1,0.5,1.

   (a) Simple Chair

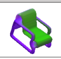
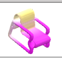
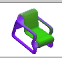
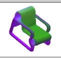
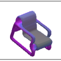
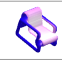
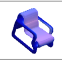
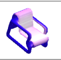
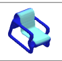
Chair with seat and arm

	λ=0	0.1	0.5	1
γ=0.001				
0.02				
0.1				

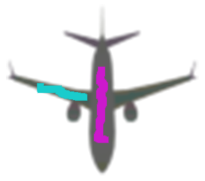
(b) Smooth Chair



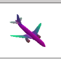
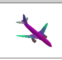
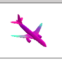
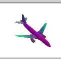
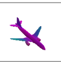
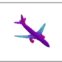
Smooth Couch Chair

	λ=0	0.1	0.5	1
γ=0.001				
0.02				
0.1				

(c) Airplane

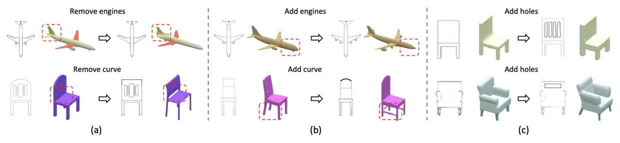


Airplane

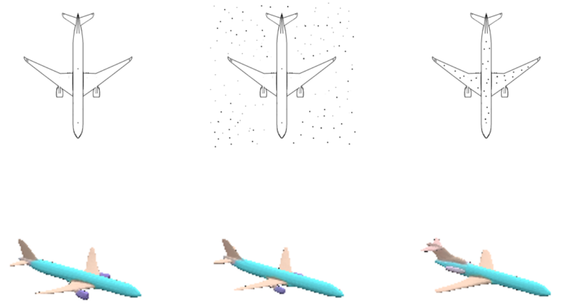
	λ=0	0.1	0.5	1
γ=0.001				
0.02				
0.1				

Observations

- For the initial chair, the values of $\gamma = 0.02$ and $\lambda = 0, 0.1$ yielded the most satisfactory desired outcomes.
- Remarkably, when $\gamma = 0.001$ and $\lambda = 0.1$ were used, it resulted in the propagation of certain magenta colors to the rear of the chair.
- This observation highlights that the original hyperparameter settings deliver the optimal result for this particular chair and color configuration.
- Regarding the second chair, which belongs to a category that the original paper struggled to generate the desired outcome for, the process of hyperparameter tuning did not lead to significant improvements.
- Although most of the results obtained with $\gamma = 0.001$ and $\lambda = 0.1$ resulted in unintended colorization outcomes, it did successfully separate the back of the chair from the seat. This particular achievement could potentially serve as an indication for future advancements and enhancements in the methodology.
- Unlike previous findings, in the case of airplanes, the model predominantly generated optimal outcomes when $\lambda = 1$, which deviates from the original hyperparameter settings.
- This observation serves as evidence that the inclusion of a term accounting for the area outside the scribbles did contribute to the improvement of certain shapes within the generated results.



1. Ignore tiny changes



References

[1] Zezhou Cheng, Menglei Chai, Jian Ren, Hsin-Ying Lee, Kyle Olszewski, Zeng Huang, Subhransu Maji, and Sergey Tulyakov. Cross-modal 3d shape generation and manipulation, 2022. 1

[2] Dale Decatur, Itai Lang, and Rana Hanocka. 3d highlighter: Localizing regions on 3d shapes via text descriptions. 2022. 3

[3] 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

[4] Zekun Hao, Hadar Averbuch-Elor, Noah Snavely, and Serge Belongie. Dualsdf: Semantic shape manipulation using a two-level representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. 1

[5] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023. 3

[6] Steven Liu, Xiuming Zhang, Zhoutong Zhang, Richard Zhang, Jun-Yan Zhu, and Bryan Russell. Editing conditional radiance fields, 2021. 1

[7] Zhaoliang Lun, Matheus Gadelha, Evangelos Kalogerakis, Subhransu Maji, and Rui Wang. 3d shape reconstruction from sketches via multi-view convolutional networks. In *2017 International Conference on 3D Vision (3DV)*, pages 67–77. IEEE, 2017. 1

[8] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019. 1

[9] Fang Wang, Le Kang, and Yi Li. Sketch-based 3d shape retrieval using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 1

[10] Jiayun Wang, Jierui Lin, Qian Yu, Runtao Liu, Yubei Chen, and Stella X Yu. 3d shape reconstruction from free-hand sketches. In *Computer Vision–ECCV 2022 Workshops: Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part VIII*, pages 184–202. Springer, 2023. 1