Text to Mesh Without 3D Supervision Using Limit Subdivision

Nasir Khalid\textsuperscript{1} Tianhao Xie\textsuperscript{1} Eugene Belilovsky\textsuperscript{1,2} Tiberiu Popa\textsuperscript{1}

\textsuperscript{1}Concordia University \textsuperscript{2}Mila

Figure 1. A 3D scene composed of objects generated using only text prompts: lamp shade, round brown table, photograph of a bust of homer, vase with pink flowers, blue sofa, pink pillow, painting in a frame, brown table, apple, banana, muffin, loaf of bread, coffee, burger, fruit basket, coca cola can, red chair, computer monitor, photo of marios cap, playstation one controller, blue pen, excalibur sword, matte painting of a bonsai tree; trending on artstation. (The 3D positioning in the scene was done by a user)

Abstract

We present a technique for zero-shot generation of a 3D model using only a target text prompt. Without a generative model or any 3D supervision our method deforms a control shape of a limit subdivided surface along with a texture map and normal map to obtain a 3D model asset that matches the input text prompt and can be deployed into games or modeling applications. We rely only on a pre-trained CLIP model that compares the input text prompt with differentiably rendered images of our 3D model. While previous works have focused on stylization or required training of generative models we perform optimization on mesh parameters directly to generate shape and texture. To improve the quality of results we also introduce a set of techniques such as render augmentations, primitive selection, prompt augmentation that guide the mesh towards a suitable result.

1. Introduction

Gaming, virtual reality, films and most other multimedia experiences rely on the use of 3D models. While there are many methods of representing these models, many existing games and modeling software used 3D assets consisting of a polygonal mesh coupled with texture and normal maps. However, the creation and texturing of meshes is a time consuming and expensive task that often also needs specialized software. There has been a lot of research focused on synthesizing shapes but these look at generation in the form of point clouds, voxel grids and implicit functions. While they provide good results the issue is that they require additional steps to be used in existing software and this conversion can lead to undesirable results or artifacts.

The ideal scenario would be a technique where a user can generate any arbitrary 3D shape based on only a abstract text description of the object. This would greatly increase the use and accessibility of developing 3D based multimedia. Furthermore, if the shape generated is in the form of a mesh and corresponding texture maps it would be much easier to integrate as existing game engines, tools and software as these expect the user to provide a polygonal mesh with texture.

A big limitation in this type of work is the lack of large varied datasets of 3D object examples and corresponding natural language descriptions. Datasets such as Shapenet [1] and CO3D [13] provide 51 and 50 object categories re-
respectively. In contrast there are large datasets containing rich 2D images with a large variety of objects. For example Imagenet-21K [14] has 21,000 object categories. Furthermore, natural image data can often be accompanied by rich textual descriptions. Recently the CLIP has been trained on a large dataset of 400 million image text pairs to learn an aligned visual and textual representation [12]. This text and image scoring model was trained on text captions with combinations from a set of 500,000 query words, leading to a very large diversity in the potential objects it can represent.

We consider utilizing the knowledge from large scale deep learning models that are trained only on images and texts. This relies on the fact that a 3D shape can be projected to a 2D image from an arbitrary viewpoint through rendering. Using a differentiable renderer one can obtain images of a shape and then use CLIP to get a loss between the images and an input text. By backpropogating this loss the shape and texture of a shape can be changed based on the input prompt. However, doing this naively will lead to a tangled and noisy mesh therefore we incorporate a regularization loss to maintain mesh geometry. However, even this is not enough as the gradients from CLIP tend to be very noisy so we also incorporate limit subdivision to further smooth the mesh. Even though this helps us minimize the loss it often leads to an undesirable result in terms of texture as CLIP may prefer "painting" small artifacts in to the texture rather than deform and globally texture the object. To alleviate this we use multiple augmentations to render the object dynamically such that it optimizes to a good solution.

Our contributions can be summarized as followed:

- We introduce a set of techniques that allow zero-shot text-guided generation with a differentiable renderer
- We use these techniques to directly generate meshes with their texture maps and normal maps
- We present a differentiable implementation of loop limit subdivision that provides the benefits of subdivision smoothing without the overhead of more vertices

2. Related work

A number of works have previously attempted to generate 3-D models from text by utilizing datasets of text descriptions corresponding to 3-D models. For example [2, 3] proposed to train a joint embedding between 3-D shapes and text and combine this with a GAN to produce novel outputs. These approaches however are not zero-shot and are thus limited by the lack of available matched 3-D models and text descriptions. Another work generates shapes from text prompts [15] but it requires training of an encoder and decoder using a set of defined meshes which limits generalizability and they also use a voxel representation which lack textures. [7, 10] focus on stylization of pre-defined object meshes with text descriptions, while we tackle the problem of generating the entire shape and texture from a detailed natural language description.
Concurrent to our work, [6] proposed a zero-shot text guided generation using a NeRF model [11]. Unlike our approach this does not allow direct generation of a mesh but instead trains a neural radiance field. This method requires raycasting and training a set of neural network parameters which has a large computation overhead even for low quality generation where as our figures are all generated on a single 12GB GPU. Additionally editing of the object and getting a mesh is not straightforward since the shape is within the weights of a network and extraction requires a user determined thresholding which can lead to trade offs. Additionally the texture and shape cannot be disentangled but in our work the shape, texture and normal can be individually modified.

3. Method

An overview of our method is shown in Figure 3. We represent a 3D model using three components: (1) a 3D mesh whose vertices $V_0 \in \mathbb{R}^{n \times 3}$ are the control vertices of a Loop [9] subdivision surface $V = S(V_0)$, (2) a texture map $T$ and (3) a normal map $\hat{T}$. This is a standard way to represent geometric assets in video games and modeling applications. Furthermore, using a texture map allows to decouple the appearance from the geometry and the combination of normal map and subdivision surface control allows us to reduce the number of optimization parameters of the geometry while maintaining rendering details. Our method creates a 3D model by optimizing these three components using a differential renderer. Our rendering pipeline uses the initial control mesh to compute the limit surface $V$ of the Loop subdivision scheme [16]. This limit surface can be computed analytically and it is a differentiable function. The loop subdivision surface $V$ is also, by construction, smooth, this surface definition acts as an implicit regularizer and helps avoiding triangle inversion during the optimization phase as shown in Figure 3 i,j). We render this mesh using using a differential renderer $R$ [8] from several camera positions $D(\varphi, \theta)$. We uniformly sample a camera azimuth angle $\varphi$ from a range of 0° to 360° and for elevation $\theta$ we sample from a Beta distribution with $\alpha = 1.0$ and $\beta = 5.0$ within a range of 0° to 100° this allows the generation to focus on making the object consistent from a single elevation angle giving it a ”front view” but the distribution allows other elevations so that textures get painted in for triangles in those regions but the shape does not deform significantly. Using these camera positions and orientation we render a set of images $I_i$:

$$I_i = R(D(\varphi_i, \theta_i), V, T, \hat{T})$$

Images $I_i$ are encoded using the CLIP image encoder $C^I$:

$$E_i = C^I(I_i)$$

Where $E_i$ represents a set of encodings for each image in $I_i$. The input to our method is a text prompt $p$ that is encoded using the CLIP text encoder $C^T$:

$$e_t = C^T(p)$$

As the rendered images as well as the text prompt are now encoded in the same space we can compute the similarity:

$$L_{CLIP}(V, T, \hat{T}, p) = \frac{1}{K} \sum_{e_k \in E_t} e_k^T e_t$$

(1)

where $i$ iterates over the images $I_i$. Note that the encoder functions, $C^T$ and $C^I$, include a normalization at the end thus these are cosine similarities. As computing the limit Loop subdivision surface is differentiable [16] and the renderer is differentiable, our entire pipeline is differentiable using the chain rule.

Additionally we use a laplacian regularizer on the shape of the mesh to maintain the geometry and keep it intact as used in other related work [5]. We use the uniformly-weighted Laplacian operator: $\delta_i = v_i - \frac{1}{|N_i|} \sum_{j \in N_i} v_j$ where $N_i$ is the set of one-ring neighbours for vertex $v_i$. With this formulation the laplacian regularizer can be given by:

$$L_{\delta} = \frac{1}{N} \sum_{i=1}^N \| \delta_i \|^2$$

(2)

where $N$ is the number of vertices. This minimizes the difference in position between each vertex and the average position of its neighbouring vertices.

We thus formulate our problem as an optimization problem with the following objective function:

$$\min_{V_0, T, \hat{T}} L_{CLIP}(S(V_0), T, \hat{T}, p) + \lambda L_{\delta}(V)$$

(3)

Practical Considerations and Implementation Details

Our approach for the laplacian regularization follows that of [5], where the weight, $\lambda$, is decayed throughout the optimization process as the shape stabilizes its final form. Initially it is set to a high value when the learning rate is high and then slowly reduces to a minimum value. More specifically, for an epoch $\lambda_t$ it is defined as $\lambda_t = (\lambda_t - \lambda_{min}) \cdot 10^{-kt} + \lambda_{min}$. We set $k = 10^{-6}$ and $\lambda_{min}$ as 2% of the initial weight $\lambda_0$. The initial weight is a hyperparameter, in our examples we find that values between 10 to 50 work best.

The look-at and up vectors of the cameras are set towards the origin and the y-axis respectively. The distance of the camera from the object is set to 5.0 in our examples. Due to the known texture bias of visual recognition models such as CLIP [4] naively performing the optimization can lead to over emphasis on the texture versus shape. To deal with this we add in some randomization to the view generation.
process by randomly selecting a camera field of view between $30^\circ$ to $60^\circ$. This variance in the field of view has a zoom in/out effect that encourages changes in the vertex positions versus only changes in the texture. Furthermore, we add two randomized features to improve the results: random choice of background in the renderings and a random offset to the position of the object in the rendering.

The initial shape for generation is selected from a set of basic primitives: horizontal or vertical cuboid and a sphere. Initially all shapes are rendered once with the same texture from multiple views and the shape with the lowest average score across the views is selected as an initial shape. This helps speed up training and leads to better mesh results since picking the right initial primitive can minimize the total deformation required. The texture map, normal map and image background are initialized with random values.

4. Results and Discussion

We have used our method on a wide variety of prompts with results shown in Figures 1 and 3. To emphasize the flexibility of generating directly the ready modeling asset (i.e. mesh, texture, and normal maps) in Figure 1 we import these directly into Blender and place the objects into a scene. We observe that we can generate a diverse set of objects with multiple attributes and including diverse objects such as hats, game controllers, and paintings.

Figure 3 a-d) shows additional examples illustrating the wide variety of shape categories that can be generated and Figure 3 g-j) illustrates the effect of texture, normal map and the loop limit surface on the geometry.

Figure 3 e,f) shows a side by side example of the same text prompt generated by our method (Figure 3 e) and the concurrent work of Jain et al. [6] Figure 3 f). In contrast to [6] our method generates 3D modeling assets ready to be deployed into games or modeling applications and is overall much more efficient. Figure 3 e, f) shows our and their result for the prompt “red chair”. We executed both codes on the same GPU: NVIDIA Titan XP with 12GB. Our method took 17 minutes on one GPU while their method took 1 hour and required the use of two GPUs.

In Figure 3 we show an ablation study using the prompt “a red chair”. Figure 3 k) shows the result without a random translation of the object and the shape struggles to grow sharp features. Figure 3 l) shows the result without randomized backgrounds and Figure 3 m) does not vary the field of view. Both give a similar result where the CLIP loss is minimized but the resulting shape is not qualitatively desirable. Figure 3 n) shows the final result.

5. Conclusions and Future Work

We have demonstrated an approach for directly generating diverse 3D models using only text descriptions. Our generated models consists of a 3D mesh, texture and normal maps making them ready to be used as assets in games and modeling applications. Future work will aim to further improve shape based constraints as well as investigate approaches for creating multiple possible objects for a given input. Additionally, we will look at methods to provide more user control in the generative process.

References

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

[2] Kevin Chen, Christopher B Choy, Manolis Savva, Angel X Chang, Thomas Funkhouser, and Silvio Savarese. Text2shape: Generating shapes from natural language by learning joint embeddings. arXiv preprint arXiv:1803.08495, 2018.

[3] Kentaro Fukamizu, Masaaki Kondo, and Ryuichi Sakamoto. Generation high resolution 3d model from natural language by generative adversarial network. arXiv preprint arXiv:1901.07165, 2019.

[4] Robert Geirhos, PatriciaRubisch, ClaudioMichaelis, MatthiasBethge, Felix A. Wichmann, and Wieland Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In International Conference on Learning Representations, 2019.

[5] Jon Hasselgren, Jacob Munkberg, Jaakko Lehtinen, Miika Aittala, and Samuli Laine. Appearance-driven automatic 3d model simplification. In Eurographics Symposium on Rendering, 2021.

[6] Ajay Jain, Ben Mildenhall, Jonathan T Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. arXiv preprint arXiv:2112.01455, 2021.

[7] Nikolay Jetchev. Clipmatrix: Text-controlled creation of 3d textured meshes. arXiv preprint arXiv:2109.12922, 2021.

[8] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. ACM Transactions on Graphics, 39(6), 2020.

[9] Charles Loop. Smooth Subdivision Surfaces Based on Triangles. PhD thesis, January 1987.

[10] Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. Text2mesh: Text-driven neural stylization for meshes. arXiv preprint arXiv:2112.03221, 2021.

[11] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In ECCV, 2020.

[12] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In International Conference on Machine Learning, pages 8748–8763. PMLR, 2021.

[13] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Shordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In International Conference on Computer Vision, 2021.

[14] Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. Imagenet-21k pretraining for the masses, 2021.

[15] Aditya Sanghi, Hang Chu, J. Lambourne, Ye Wang, Chin-Yi Cheng, and Marco Fumero. Clip-forge: Towards zero-shot text-to-shape generation. ArXiv, abs/2110.02624, 2021.