CLIP-Mesh: Generating textured meshes from text using pretrained image-text models

Nasir Mohammad Khalid
nasirmkhalid24@gmail.com
Concordia University
Montreal, Canada
Mila
Montreal, Canada

Tianhao Xie
tianhao.xie@mail.concordia.ca
Concordia University
Montreal, Canada

Eugene Belilovsky
Belilovsky.Eugene@gmail.com
Concordia University
Montreal, Canada
Mila
Montreal, Canada

Tiberiu Popa
tiberiu.popa@concordia.ca
Concordia University
Montreal, Canada

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 any 3D supervision our method deforms the control shape of a limit subdivided surface along with its texture map and normal map to obtain a 3D asset that corresponds to the input text prompt and can be easily 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, texture or both. To constrain the optimization to produce plausible meshes and textures we introduce a number of techniques using image augmentations and the use of a pretrained prior that generates CLIP image embeddings given a text embedding.

SA '22 Conference Papers, December 6–9, 2022, Daegu, Republic of Korea

ACM ISBN 978-1-4503-9470-3/22/12…$15.00
https://doi.org/10.1145/3550469.3555392

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Mesh geometry models**.

KEYWORDS

CLIP, neural networks, machine learning, geometric modeling

ACM Reference Format:
Nasir Mohammad Khalid, Tianhao Xie, Eugene Belilovsky, and Tiberiu Popa. 2022. CLIP-Mesh: Generating textured meshes from text using pretrained image-text models. In *SIGGRAPH Asia 2022 Conference Papers (SA '22 Conference Papers), December 6–9, 2022, Daegu, Republic of Korea.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3550469.3555392

1 INTRODUCTION

Gaming, virtual reality, films and other multimedia experiences rely on the use of 3D models. While there are various methods of representing these models, many existing games and modeling software use 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 or implicit functions and are restricted to fixed shape categories. While these provide good results the issue is they require additional steps to convert to meshes that can 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 an abstract text description of the object. This would greatly increase the use and accessibility of developing 3D assets. Furthermore, if the shape generated is in the form of a mesh with corresponding texture maps it would easily facilitate integration with a large suite of existing game engines and software that use 3D meshes as primitives.

A big limitation is the lack of large varied datasets of 3D objects and corresponding natural language descriptions. For example, datasets such as Shapenet [Chang et al. 2015] and CO3D [Reizenstein et al. 2021] provide 50 object categories respectively. In contrast there are large datasets containing rich 2D images with a large variety of objects. For example Imagenet-21K [Ridnik et al. 2021] has 21,000 object categories. Furthermore, natural image data can often be accompanied by rich textual descriptions. Recently CLIP has been trained on a large dataset of 400 million image text pairs to learn an aligned visual and textual representation [Radford et al. 2021]. 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 thus consider utilizing the knowledge from a large scale deep learning model 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 such as [Laine et al. 2020] one can obtain images of a shape and then use CLIP to get a score between the images and an input text. Leveraging the differentiability of the renderer and CLIP, an inverse problem can be solved by optimizing the shape and texture of a mesh to maximize the CLIP score of rendered images and input prompt. However, doing this naively can lead to a tangled and noisy mesh as there are insufficient constraints on the shape. Therefore we incorporate a number of constraints and techniques that allow us to generate a plausible shape and texture. First we use a regularization loss and incorporate limit subdivision to further smooth the mesh. Even though this helps us maximize the score 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 the optimizitation reaches a solution leveraging the shape information. Additionally to further improve results we introduce a conditional generative model that uses a pretrained diffusion prior model, that generates a CLIP image embedding given the text prompt, this is similar to current state of the art text to image synthesis work [Ramesh et al. 2022]. 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 3D meshes with their texture maps and normal maps .
- We use the analytical expression of the Loop subdivision limit surface as an implicit regularizer to improve the quality of the generated model.
- We improve on our baseline results by introducing a set of render augmentations and incorporating a text to image embedding prior.

2 RELATED WORK

A variety of recent works focus on text driven 2D image manipulation and generation using CLIP [Radford et al. 2021], a model that learns a joint embedding space for image and text. Leveraging CLIPs joint embedding, many works such as StyleCLIP [Patashnik et al. 2021], VQGAN-CLIP [Crowson et al. 2022] and GLIDE [Nichol et al. 2021] have shown that pretrained image generative models can be guided by text prompts through distance losses in the shared embedding space. Additionally, the current state of the art in text to image generation trains a model directly on CLIP text and image embeddings [Ramesh et al. 2022].

In contrast to this, text to 3D is an underdeveloped field but a number of works have previously attempted to generate 3D models from text by utilizing datasets of text descriptions corresponding to 3D models. For example [Chen et al. 2018; Fukamizu et al. 2019] proposed to train a joint embedding between 3D shapes and text and combine this with generative adversarial networks [Goodfellow et al. 2014] to produce novel outputs. These approaches however are not zero-shot and are thus limited by the lack of available matched 3D models and text descriptions. CLIP-Forge [Sanghi et al. 2021] alleviates the issue of paired text and 3D models by relying only on the 3D models to train an encoder and decoder and then guiding generation of the decoder with CLIP to produce results that match a text prompt, this only partially solves the problem because now the generation is restricted by the 3D data categories available to train it. It also doesn't produce meshes or textures and thus its use is limited. [Hong et al. 2022; Jetchev 2021] focus on stylization of predefined human shape to match an input text prompt and [Michel et al. 2021] generalizes this to any arbitrary mesh and text prompt. Text2Mesh addresses a related but different problem: with a correct starting mesh each vertex is minimally modified along the normal direction and its color.

Dreamfields [Jain et al. 2021] proposed a zero-shot text guided generation using a NeRF model [Mildenhall et al. 2020]. 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 16GB 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. Furthermore, the texture and shape cannot be disentangled. While in our work the shape, texture and normal can be individually modified allowing unique application scenarios. For example we demonstrate multi object optimization within a scene which is straightforward under our method but the same cannot be easily applied using [Jain et al. 2021].

3 METHOD

An overview of our method is shown in Figure 2. We represent a 3D model using three components: (1) a 3D mesh whose vertices $\boldsymbol{V}_0 \in \mathbb{R}^{n\times 3}$ are the control vertices of a Loop [Loop 1987] subdivision surface $\boldsymbol{V} = S(\boldsymbol{V}_0)$, (2) a texture map T and (3) a normal map $\tilde{\boldsymbol{T}}$. 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 differentiable renderer. Our rendering pipeline uses the initial control mesh to compute the limit surface V of the Loop subdivision scheme [Stam 1998]. This limit surface can be computed analytically and it is a differentiable function. The loop subdivision surface V is also, by construction, smooth. Therefore, this surface definition acts as an implicit regularizer and helps avoid triangle inversion during the optimization phase. We render this mesh using a differentiable renderer R [Laine et al. 2020] from several camera positions $D(\varphi, \theta)$. We uniformly sample a camera azimuth angle φ from a range of 0° to 360° and for elevation θ we sample from a Beta distribution with $\alpha_d = 1.0$ and $\beta_d = 5.0$ 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:

$$\boldsymbol{I} = R(D(\varphi_i, \theta_i), \boldsymbol{V}, \boldsymbol{T}, \tilde{\boldsymbol{T}})$$

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

$$\boldsymbol{E} = C^I(\boldsymbol{I})$$

Where E represents a set of encodings for each image in I. The input to our method is a text prompt $\boldsymbol{p}$ that is encoded using the CLIP text encoder C^T:

$$\boldsymbol{e}_t = C^T(\boldsymbol{p})$$

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

$$L_{CLIP}(\boldsymbol{V}, \boldsymbol{T}, \tilde{\boldsymbol{T}}, \boldsymbol{p}) = -\frac{1}{K} \sum_{\boldsymbol{e}_i \in \boldsymbol{E}} \boldsymbol{e}_i^T \boldsymbol{e}_t \tag{1}$$

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 [Stam 1998] and the renderer is differentiable, our entire pipeline is differentiable using the chain rule.

Laplacian Regularizer. 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 [Hasselgren et al. 2021]. 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:

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

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

Diffusion Prior. To further improve results we also train and incorporate a diffusion prior which attempts to generate image embeddings following $p(e_i|e_t)$. We use this to sample image embeddings given a text encoding. Our formulation follows that of [Ramesh et al. 2022] and [Ho et al. 2020]. Once trained, the diffusion sampling process takes input of noise and the CLIP text embedding $\boldsymbol{e}_t$ and after applying the forward process for N timesteps the output is a CLIP image embedding which follows $p(e_i|e_t)$.

We pretrain this prior on a 400 million image and text pair dataset [Schuhmann et al. 2021] so it can sample a relevant CLIP image embedding when given a CLIP text embedding and during optimization time we sample from it using the previously obtained text embedding $\boldsymbol{e}_t$ to get a relevant CLIP image embedding $\hat{\boldsymbol{e}}_{\boldsymbol{k}}$. As the rendered images are encoded in the same space as the output embedding we can also compute a similarity between them to use as a loss.

$$L_{PRIOR}(\boldsymbol{V}, \boldsymbol{T}, \tilde{\boldsymbol{T}}, \boldsymbol{p}) = -\frac{1}{K} \sum_{\boldsymbol{e}_i \in E} \boldsymbol{e}_i^T \hat{\boldsymbol{e}}_k \tag{3}$$

Since it is conditioned on the text embeding we can use L_{PRIOR} without L_{CLIP}. Practically, in our preliminary experiments we found that combining these losses can be beneficial.

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

$$\min_{\boldsymbol{V}_0, \boldsymbol{T}, \tilde{\boldsymbol{T}}} L_{CLIP}(S(\boldsymbol{V}_0), \boldsymbol{T}, \tilde{\boldsymbol{T}}, \boldsymbol{p}) + \lambda_t L_\delta(\boldsymbol{V}) + \alpha L_{PRIOR}(S(\boldsymbol{V}_0), \boldsymbol{T}, \tilde{\boldsymbol{T}}, \boldsymbol{p}) \tag{4}$$

Practical Considerations and Implementation Details. Our initial shape is a sphere with 600 vertices. The texture map is initialized with random values and is set to a resolution of 512x512. The normal map has the same resolution but is initialized as a uniform blue image. Adam optimizer is used for the vertices and texture maps with a decaying learning starting at 0.001 and a batch size of 25. The

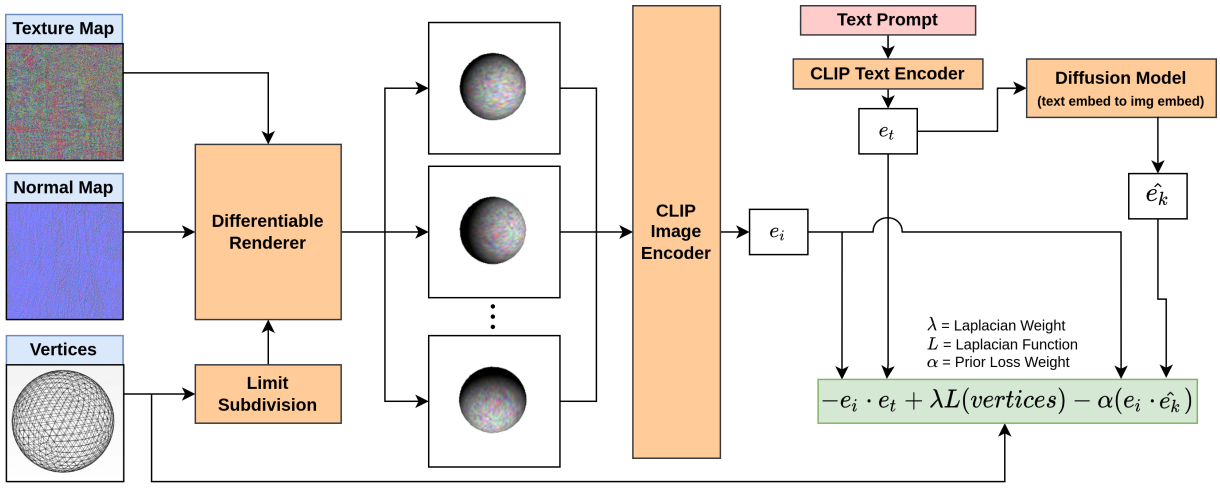

Figure 2: Overview of our optimization pipeline. The differentiable renderer creates views which are encoded and compared to the text encoding as well as the generated image embedding. We optimize for the texture, normal, vertices position.

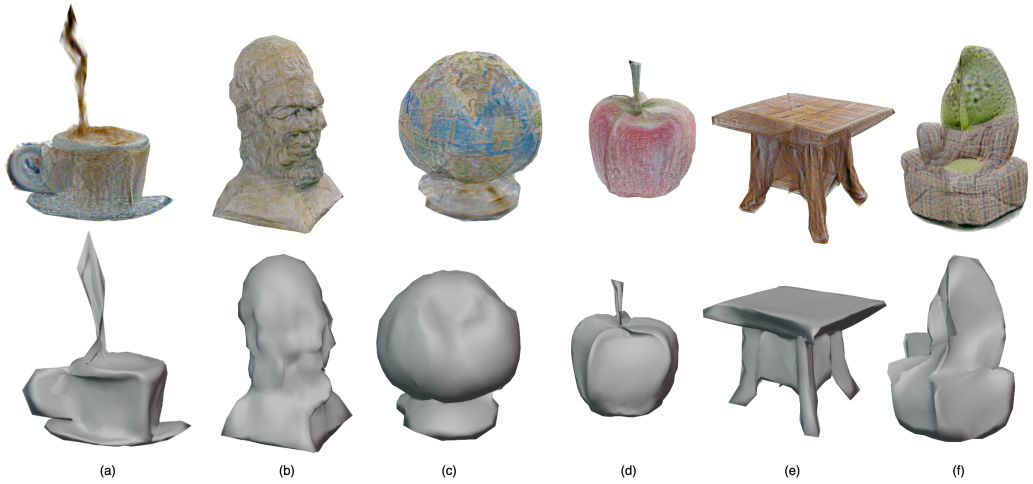

Figure 3: Results from a wide variety of prompts. Top: rendered result. Bottom: 3D mesh. a) "a coffee" b) "a photograph of a bust of homer" c) "Globe" d) "a apple" e) "a brown table" f) "an armchair in the shape of an avocado"

diffusion prior follows the same configuration setup as [Ramesh et al. 2022] except ours is scaled down.

The approach for the laplacian regularization follows that of [Hasselgren et al. 2021], where the weight, λ, 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 t it is defined as $\lambda_t = (\lambda_{t-1} - \lambda_{min}) \cdot 10^{-kt} + \lambda_{min}$. The initial weight and decay parameters are hyperparameters that can be tuned.

The look-at and up vectors of the cameras are set towards the origin and the y-axis respectively. Due to the known texture bias of visual recognition models such as CLIP [Geirhos et al. 2019] 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° to 60° and varying the distance of the camera from the object to between 3.0 to 7.0. This variance in the field of view and distance has a zoom in/out effect that encourages changes in the vertex positions versus only changes in the texture. CLIP takes 224x224 input images but we find that rendering at a larger 512x512 resolution and down scaling to 224x224 improves results, it also plays well with the differentiable render we use [Laine et al. 2020] since it relies on anti aliasing for gradients and rendering at a larger resolution means more pixels are affected by anti aliasing which reduces gradient noise.

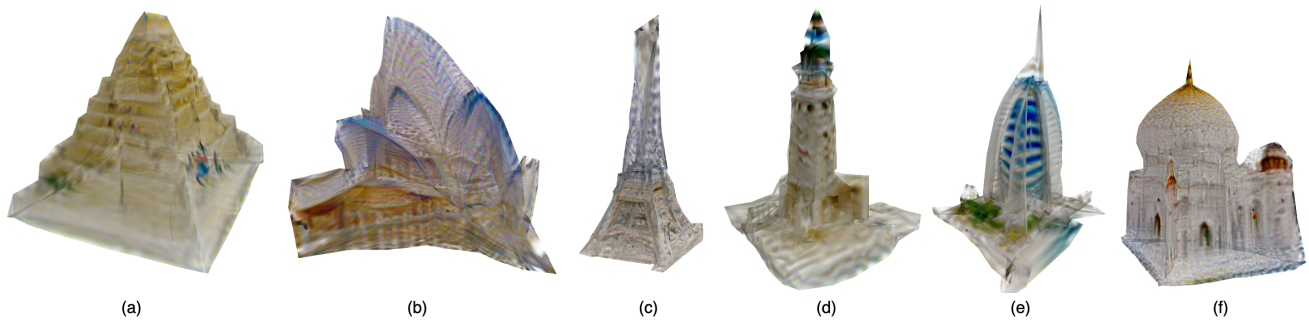

Figure 4: Reconstruction of famous landmark around the world: a) "pyramid of giza " b) "Sydney opera house" c) "Eiffel Tower" d) "lighthouse of alexandria" e) "Burj Al Arab" f) "Taj Mahal"

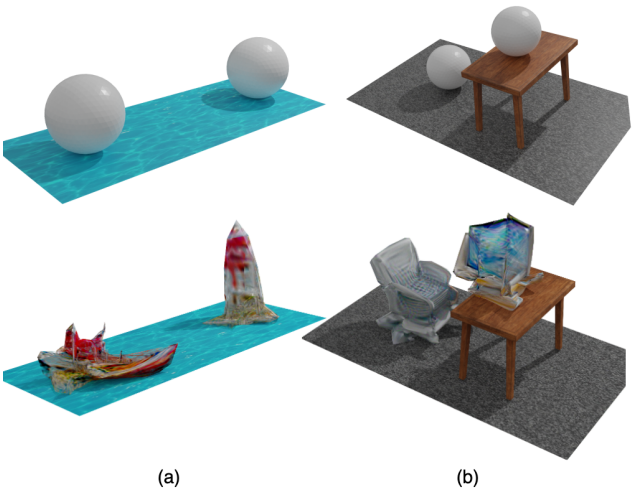

Figure 5: Multiple object optimization. Prompts: a) "boat and red lighthouse" b) "office chair and a desk and a computer monitor"

Random Augmentations. To further improve results we use augmentations throughout the optimization process for Eq. 4. At each iteration we render images with a random background similar to [Jain et al. 2021] using either gaussian noise, a solid color or a random color checkboard pattern. Additionally, we randomly offset the object so that it is not always positioned in the center of the image. These augmentations are used so that the CLIP optimization does not rely on the background or position of the object to minimize the loss but is instead forced to generate a shape that is coherent under all conditions.

4 RESULTS AND EVALUATIONS

We evaluated our methods on a wide variety of prompts and a few different generation scenarios. We first look at the single object generation scenario and compare our method with Jain et al. [Jain et al. 2021]. We then follow up with additional modeling scenarios unique to our method. Finally we provide quantitative evaluations of our results as well as ablation studies to illustrate the improvement provided by each step of our method.

4.1 Single Object Generation

In Figure 1 we illustrate a number of household objects generated using the proposed method. The flexibility of the assets created is illustrated as we import and place them into a 3D scene. In Figure 3 we further illustrate a diverse set of objects and their corresponding shape (removing the texture). Finally in Figure 4 we further show the diversity of possible objects that can be generated using the knowledge of the CLIP model by producing famous landmarks which are visually recognizable. In all these figures we use the CLIP ViT/B-32 model for training.

We also provided visual comparisons to [Jain et al. 2021]. Fig. 7 shows the results of our methods results with five prompts from [Jain et al. 2021] with the results shared in their paper and project website. We render the meshes from similar angles. Fig. 7 shows a second comparison with [Jain et al. 2021] where we chose new prompts and generated the results using the code available online. Note that because their work uses a NeRF representation and requires ray casting it comes with a large resource constraint. Therefore we use the smallest CLIP ViT-B/16 model for the generations and use the medium quality configuration provided in their codebase.

In terms of speed our method is much faster than Dreamfields [Jain et al. 2021] where each shape took over 24 hours to generate using 4 NVIDIA A100 GPUs. For similar configurations our experiments revealed that our method is faster by a factor of 100 as each of our shapes required 50 minutes on a single NVIDIA P100 (16GB) GPU. In short the reason for this is two-fold: 1) the number of optimizing parameters in Dreamfields is much higher (all the weights of a complex neural network as opposed to vertex positions, texture and normal maps) 2) our rasterization based rendering is much faster.

4.2 Complex Modeling Scenarios

Another powerful feature of our method (and unique among NERF based approaches such as [Jain et al. 2021]) is the flexibility of our optimization framework. The texture and shape are decoupled allowing us to selectively optimize them if needed, and to generate

Figure 6: Multiple object optimization where one of the objects has fixed shape. a) initial shapes. The following are results of the following captions: b) "cactus and sand" c) "wooden boat and blue water" d) "brown wooden table and iranian carpet" e) "fruit basket on grass"

Figure 7: Comparison with [Jain et al. 2021] results from their paper/project website. Top: results from [Jain et al. 2021]. Bottom: our results. Prompts: a) "matte painting of a bonsai tree; trending on art station" b) "matte painting of a castle made of cheesecake surrounded by a moat made of ice cream; trending on artstation; unreal engine" c) "a cluster of pine trees are in a barren area" d) "a cluster of pine trees are in a barren area" e) "a sculpture of a rooster"

multiple objects in context. This provides a number of unique possibilities for user control of the generation. Additionally since we use meshes it is trivial to combine multiple meshes in to a single mesh while also freezing some vertices and allowing others to be optimized. All this allows us to perform simultaneous optimization of multiple objects as well as separate the shape and texture optimization. This can be be very useful when modeling a scene where some objects have fixed shape while other objects are allowed to vary.

Figure 5 shows an example of this multiple object optimization. In Figure 5a) the text caption used was "boat and red lighthouse", the initial setup was a plane with fixed water texture and 2 spheres on either ends. Vertices and texture for the water were frozen but spheres allowed to optimize. The final result created two distinct shapes for each object in the caption that fits the scene. In Figure 5b) a similar setup is followed where the carpet and table are static, but the chair and computer monitor are automatically generated from initial spheres. Note that while the starting position of one of the sphere was on the table, we did not specify anywhere explicitly that the monitor should be on the top of the table or that the chair must face the monitor, all of this was inferred implicitly by the model. Figure 6 shows another example of our methods diversity and simultaneous optimization where the sphere allows for shape, texture and normal map optimization while the plane allows only for texture and normal map optimization. We show results for various distinct captions and also note that the texture and normal map of the plane optimize to support the object such as a picnic mat texture appearing when the caption is a "fruit basket on grass"

4.3 Quantitative Evaluation

We quantitatively evaluate our method, comparing it directly with the current closest work of [Jain et al. 2021]. We follow the same

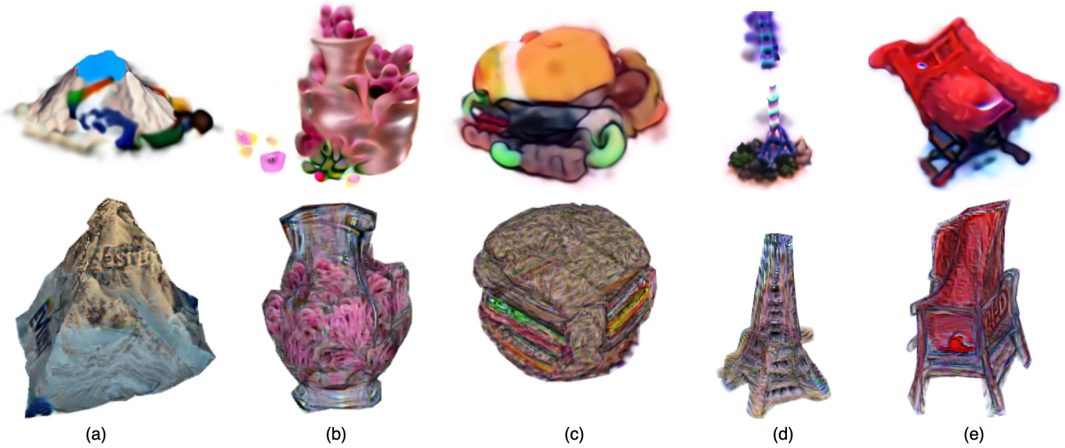

Figure 8: Comparison with [Jain et al. 2021]. Shapes generated using CLIP ViT/B-16 Top: results from [Jain et al. 2021]. Bottom: our results. Prompts: a) "mount everest" b) "a vase with pink flowers" c) "a hamburger" d) "Eiffel tower" e) "a red chair"

Table 1: Quantative comparision of our work with dreamfields on COCO caption object generation

Generation Model	CLIP ViT-B/16		CLIP ViT-B/32	
Evaluation Model	ViT-B/16	ViT-B/32	ViT-B/16	ViT-B/32
Dreamfields [Jain et al. 2021]	93.5	59.8	74.2	86.6
CLIP-Mesh [Ours]	**96.7**	**67.8**	**75.8**	**91.4**

experiment setup outlined in their paper: two shapes are generated per caption for a set of 153 text captions, for a total of 306 generated shapes. During evaluation they are rendered from a held-out pose not seen during training, a CLIP-R precision score [Park et al. 2021] is then computed between the held-out pose renderings and the captions used to generate the shapes. The captions used are from [Jain et al. 2021] and the held out pose is also the same as theirs at a 45° elevation where as training is limited to a 30° elevations, we experiment with different sized CLIP models for generating the shapes and computing the precision.

Table 1 shows the quantitative results of the evaluation. Note that in this evaluation we do not include the diffusion prior loss as the dataset for training the prior contains only CLIP ViT-B/32 embeddings, so we are unable to train a prior that supports the generation model of CLIP ViT-B/16. Regardless, we find that our work outperforms [Jain et al. 2021] across the generation and evaluation models without it.

4.4 Ablation Studies

In Table 2 an ablation study is shown for the various components of our pipeline. We start from a stripped down version of our method (baseline) and sequentially add in the components of the method. We follow the same evaluation methodology as in Table 1 but a single shape is generated per caption here instead of two as we found that it does not have a significant impact on the metric and reduces the time required per evaluation. Our results show that

Table 2: Ablation study on the R-Precision quantitative metric where higher score is better. We observe that starting from a baseline approach, adding limit subdivision, augmentation, large rendering, and the generative prior systematically improves performance

Method (CLIP B/16)		CLIP R-Precision ↑		
		B/16	*B/32*	*L/14*
Shape	Baseline Method	75.8	41.8	50.9
	+ Limit Subdivision	77.7	47.7	53.5
Augmentations	+ Background	81	47.7	58.8
	+ Repositon Shape	90.1	60.5	73.2
Render	+ 512^2 renders	92.1	62.7	70.5
Prior	+ Prior Loss	91.5	77.7	74.5

the limit subdivision provides an improvement across all retrieval models. We then add the image augmentations which both provide improvements, offsetting the mesh from the center of the image provides the largest boost to the final results. Similarly, rendering the images at a higher resolution and then linearly scaling to the CLIP 224x224 resolution does improve results in all cases except for the largest ViT-L/14 model where it hurts performance. We get our best overall results when adding the prior loss.

5 CONCLUSIONS, LIMITATIONS AND FUTURE WORK

We have demonstrated a method for generating diverse 3D objects in different modeling scenarios using only an input text prompt. The results consist of a mesh, texture map and normal map which allow them to be directly loaded to be used as assets in games and modelling applications. While the work we propose provides interesting results there are some limitations of our method.

Genus. The genus of the generated object is set by the initial template mesh. We address this issue partially by allowing a transparency channel in the texture, but a more principled approach is desirable.

CLIP Limitations. Using an image model to generate 3D shapes comes with its own challenges, since the model is trained with images it often projects artifacts to the mesh. Some examples of this can be seen in 4 where the pyramid has small people on its side and 8 where the mount Everest has the text "Everest" on its side and tip, note that we find using the larger CLIP ViT/B-32 model alleviates the text issue.

In future work we will aim to further improve shape based constraints and explore methods to provide more user control in the generative process.

ACKNOWLEDGMENTS

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) [RGPIN-2021-04104 and RGPIN-2021-03477]. This research was enabled in part by support provided by Calcul Quebec (calculquebec.ca) and the Digital Research Alliance of Canada (alliancecan.ca).

REFERENCES

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

Kevin Chen, Christopher B Choy, Manolis Savva, Angel X Chang, Thomas Funkhouser, and Silvio Savarese. 2018. Text2Shape: Generating Shapes from Natural Language by Learning Joint Embeddings. *arXiv preprint arXiv:1803.08495* (2018).

Katherine Crowson, Stella Biderman, Daniel Kornis, Dashiell Stander, Eric Hallahan, Louis Castricato, and Edward Raff. 2022. VQGAN-CLIP: Open Domain Image Generation and Editing with Natural Language Guidance. *arXiv preprint arXiv:2204.08583* (2022).

Kentaro Fukamizu, Masaaki Kondo, and Ryuichi Sakamoto. 2019. Generation High resolution 3D model from natural language by Generative Adversarial Network. *arXiv preprint arXiv:1901.07165* (2019).

Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. 2019. ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness.. In *International Conference on Learning Representations*. https://openreview.net/forum?id=Bygh9j09KX

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems* 27 (2014).

Jon Hasselgren, Jacob Munkberg, Jaakko Lehtinen, Miika Aittala, and Samuli Laine. 2021. Appearance-Driven Automatic 3D Model Simplification. In *Eurographics Symposium on Rendering*.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems* 33 (2020), 6840–6851.

Fangzhou Hong, Mingyuan Zhang, Liang Pan, Zhongang Cai, Lei Yang, and Ziwei Liu. 2022. AvatarCLIP: Zero-Shot Text-Driven Generation and Animation of 3D Avatars. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–19.

Ajay Jain, Ben Mildenhall, Jonathan T Barron, Pieter Abbeel, and Ben Poole. 2021. Zero-Shot Text-Guided Object Generation with Dream Fields. *arXiv preprint arXiv:2112.01455* (2021).

Nikolay Jetchev. 2021. ClipMatrix: Text-controlled Creation of 3D Textured Meshes. *arXiv preprint arXiv:2109.12922* (2021).

Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. 2020. Modular Primitives for High-Performance Differentiable Rendering. *ACM Transactions on Graphics* 39, 6 (2020).

Charles Loop. 1987. *Smooth Subdivision Surfaces Based on Triangles*. Ph. D. Dissertation. https://www.microsoft.com/en-us/research/publication/smooth-subdivision-surfaces-based-on-triangles/

Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. 2021. Text2Mesh: Text-Driven Neural Stylization for Meshes. *arXiv preprint arXiv:2112.03221* (2021).

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*.

Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. 2021. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741* (2021).

Dong Huk Park, Samaneh Azadi, Xihui Liu, Trevor Darrell, and Anna Rohrbach. 2021. Benchmark for Compositional Text-to-Image Synthesis. In *NeurIPS Datasets and Benchmarks*.

Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. 2021. StyleCLIP: Text-Driven Manipulation of StyleGAN Imagery. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2085–2094.

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

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical Text-Conditional Image Generation with CLIP Latents. *ArXiv* abs/2204.06125 (2022).

Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. 2021. Common Objects in 3D: Large-Scale Learning and Evaluation of Real-life 3D Category Reconstruction. In *International Conference on Computer Vision*.

Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. 2021. ImageNet-21K Pretraining for the Masses. arXiv:2104.10972 [cs.CV]

Aditya Sanghi, Hang Chu, Joseph G Lambourne, Ye Wang, Chin-Yi Cheng, and Marco Fumero. 2021. Clip-forge: Towards zero-shot text-to-shape generation. *arXiv preprint arXiv:2110.02624* (2021).

Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. 2021. LAION-400M: Open Dataset of CLIP-Filtered 400 Million Image-Text Pairs. *ArXiv* abs/2111.02114 (2021).

Jos Stam. 1998. Evaluation of loop subdivision surfaces. In *SIGGRAPH'98 CDROM Proceedings*. Citeseer.