SFU

Practices in visual computing II (Spring 2023)
Assignment 4
text driven mesh stylization (Text2Mesh)

Total points: 100

Due: Friday, 21 April, 2:25 PM

Introduction

3D asset creation is a challenging task that requires considerable expertise and computational resources. Traditionally, creating stylized meshes has been a time-consuming and labor-intensive process that requires skilled 3D artists to manually sculpt the mesh's topology and geometry. This process can be prohibitively expensive and may not always yield the desired results. As such, there has been growing interest in developing automated methods for generating stylized meshes.

In recent years several advances in machine learning and visual computing have paved the way for solving this problem. First, the appearance of vision-language models such as Clip [4] which enable using text as an intuitive handle for generating and manipulating visual assets, secondly the introduction of neural fields and coordinate based neural networks that as a novel representation of visual data have revolutionized the field, and finally introduction of differentiable renderers which allow us to optimize constraints on 3D meshes that are defined on 2d images.

In this assignment you explore the Text2Mesh[3] work which utilizes all of the three areas described above to solve the task of mesh stylization. Simply described, this work takes a mesh and a text-prompt as input, and changes the color and geometry of the mesh in a way that adheres to the text-prompt 1.

Part 1: The Kaolin library and differentiable renderers [30 points]

Introduced by Nvidia Kaolin [1] is a PyTorch API for 3D representation IO, manipulation, conversion and etc. It also provides multiple differentiable renderers which given a

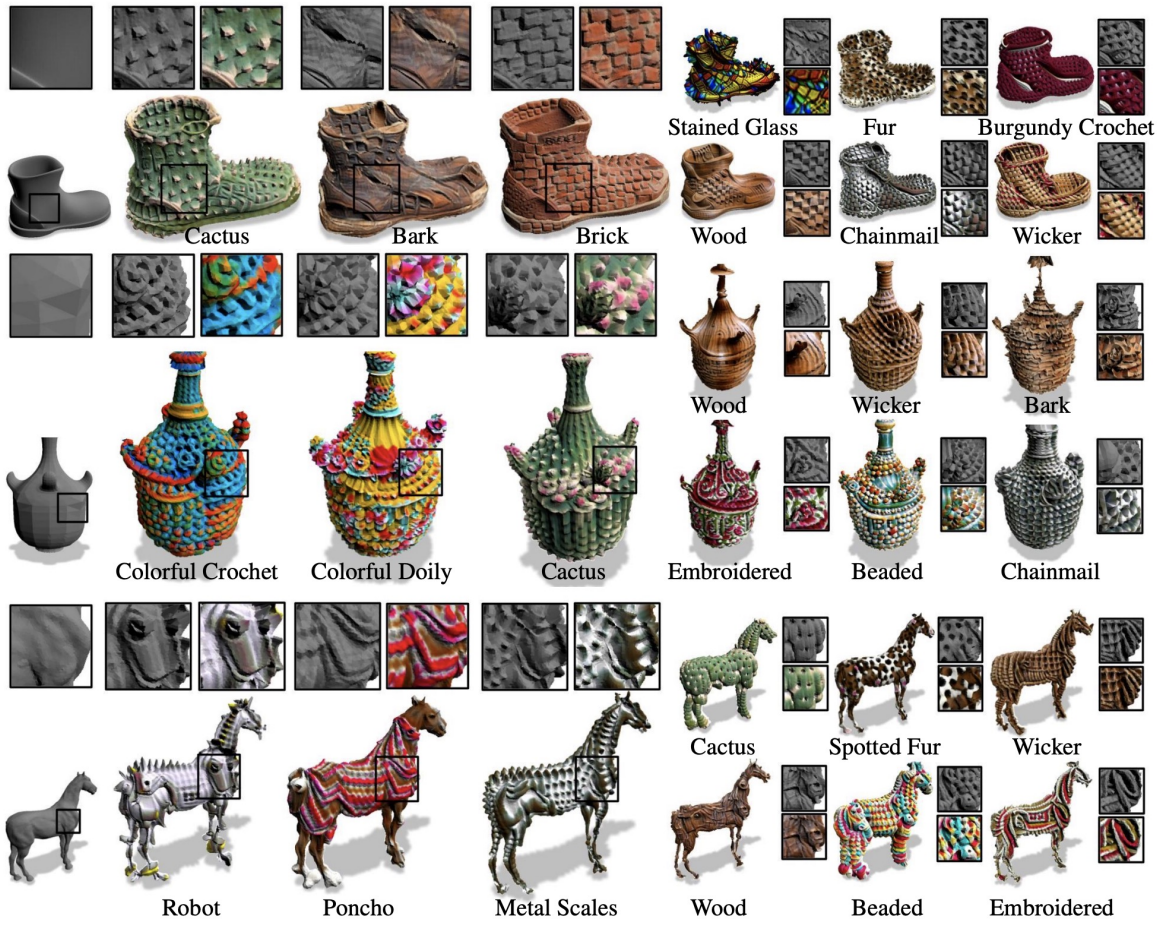

Figure 1: Examples of the text2mesh input/output

3D representation such as a triangle mesh, renders the mesh and outputs a 2D image. Differentiable renderers are a class of renderers which consist of series of operations which are differentiable and therefore are very suited for integration in deep learning settings. These renderers can be used to backpropagate a loss that is defined in the image space such as the Clip similarity loss to the parameters of a 3D representation, like the color and vertex displacements of a triangle mesh.

In this part of the assignment, you should extend what we did in the lab session for rendering a single image using Kaolin and move a camera in a circular motion around the object and output a video.

Part 2: Text2Mesh [35 points]

In this part of the assignment you should read the Text2Mesh [3] paper and fully understand its pipeline. Furthermore, you can use the source code of the work available at this link to run the code. In the demo you should provide at least two examples of mesh-text pair results. Use the previous section to get circular renders of the output of the model and demonstrate the result in the demo session.

Part 3: Positional encoding [35 points]

Like every recent coordinate based neural network, Text2Mesh uses positional encoding to mitigate the frequency bias of neural networks. The frequency bias is a phenomenon discovered recently which states that neural networks are biased towards learning low frequency representations and therefore cannot represent highly detailed and high frequency signals. Positional encoding is a simple method to solve this problem by simply concatenating the input of the network with different frequency bands [5]. More precisely given point p as input of the neural network, we change it to:

$$\gamma(p) = [cos(2\pi Bp), sin(2\pi Bp)], \tag{1}$$

in which B is a random Gaussian matrix which entries are drawn from $\mathcal{N}(0, \sigma^2)$. Additionally Text2Mesh uses a trick that high frequency inputs are gradually introduced to the network by using a mask which uncovers the high frequency parts of $\gamma(p)$ gradually as training progresses. This trick was introduced by [2] and it has been shown to imporve the result of most neural field and coordinate based neural networks. This part of the network is implemented in the **ProgressiveEncoding** class in the `neural_style_field.py` file in the source code.

The paper claims that changing the σ value in the random matrix affects the details generated by the network. (Figure 7 in the paper). In this part you should experiment this claim and run examples with different σs and report the result in the demo session.

References

[1] Clement Fuji Tsang, Maria Shugrina, Jean Francois Lafleche, Towaki Takikawa, Jiehan Wang, Charles Loop, Wenzheng Chen, Krishna Murthy Jatavallabhula, Edward Smith, Artem Rozantsev, Or Perel, Tianchang Shen, Jun Gao, Sanja Fi-

dler, Gavriel State, Jason Gorski, Tommy Xiang, Jianing Li, Michael Li, and Rev Lebaredian. Kaolin: A pytorch library for accelerating 3d deep learning research. `https://github.com/NVIDIAGameWorks/kaolin`, 2022.

[2] Amir Hertz, Or Perel, Raja Giryes, Olga Sorkine-Hornung, and Daniel Cohen-Or. Sape: Spatially-adaptive progressive encoding for neural optimization. *arXiv preprint arXiv:2104.09125*, 2021.

[3] 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.

[4] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[5] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020.