Supplemental Material: CLIP-based Neural Neighbor Style Transfer for 3D Assets

Shailesh Mishra[†1,2] and Jonathan Granskog[†3]

[1]German Research Center for Artificial Intelligence (DFKI), [2]NVIDIA, [3]Runway

Abstract

We present a method for transferring the style from a set of images to the texture of a 3D object. The texture of an asset is optimized with a differentiable renderer and losses using pretrained deep neural networks. More specifically, we utilize a nearest-neighbor feature matching (NNFM) loss with CLIP-ResNet50 that we extend to support multiple style images. We improve color accuracy and artistic control with an extra loss on user-provided or automatically extracted color palettes. Finally, we show that a CLIP-based NNFM loss provides a different appearance over a VGG-based one by focusing more on textural details over geometric shapes. However, we note that user preference is still subjective.

CCS Concepts

• ***Computing methodologies*** → *Appearance and texture representations; Rasterization; Supervised learning by regression;*

1. Optimization

In this section, we outline the details of our optimization process. All our results were produced on V100 GPUs where optimizing a single model on one GPU took three hours while using a node of eight GPUs brought the time down to 20 minutes per asset. The code was implemented in PyTorch [PGM*19] and for differentiable rendering we used nvdiffrast from Laine et al. [LHK*20]

We used batch size 8 with learning rate 10^{-2}. Unless otherwise mentioned, each batch element was randomized with camera distances based on the model and angles randomized in the sphere around the model. The power of the point light in the scene was fixed to 2 and its position randomized in the hemisphere with radiuses in the range of $[3,5]$. We also used a Phong material model with roughness set to 2. The render output resolution from the differentiable renderer was set to 512×512 and the texture resolution was set to 1024×1024 unless mentioned otherwise. We optimize a separate style texture that is added to the original texture for rendering. Then, the two textures can be controlled and modified independently afterwards.

For our main results using CLIP-ResNet50, we use the following loss weights; $\lambda_S = 10^4, \lambda_C = 22$ and the color loss weight λ_P in the beginning of optimization was set to 2000 but then reduced during the training duration. For other results, we had to tune the loss balance to adjust for missing losses or varying loss magnitudes. For VGG NNFM, we used $\lambda_S = 200, \lambda_C = 1.0$ and the initial weight for the color loss was also 2000.

† Work done while both were employed at NVIDIA

Figure 1: *Applying losses directly to the textures are another option. However, optimizing this way does not reduce discontinuities in the texture coordinates. Seams in the texture mapping are therefore more visible.*

2. Additional results

Here, we show additional results that might give more insight into how different training aspects affect the optimization result.

Direct texture optimization The differentiable renderer allows backpropagating image-space losses through the renderer to the 2D texture applied on the mesh. However, image-space losses could also be directly computed for the texture. In Figure 1, we show a comparison of the two approaches. Differentiable rendering allows optimizing textures while keeping learned patterns consistent and scaled similarly across seams. Discontinuities are more visible when optimizing textures directly (see the front leg of the cow).

Camera distance variation Camera distance plays a significant role in the size of patterns optimized into the texture. In Figure 2,

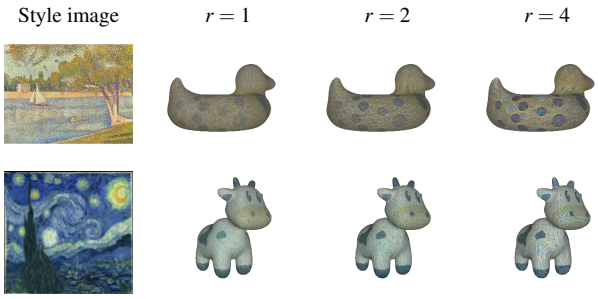

Figure 2: *Camera distance plays an important part in the size of the patterns found on the object. Here we compare different fixed camera distances r used during optimization while still rotating in the sphere around the object.*

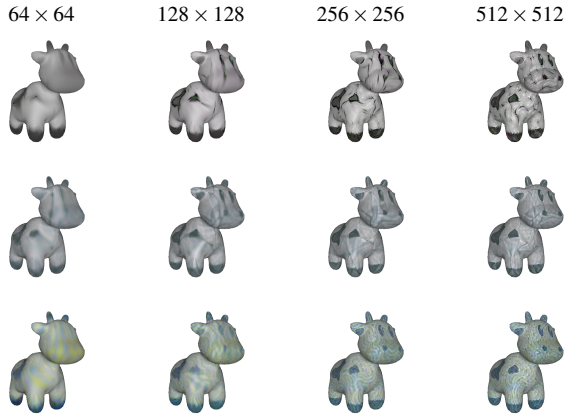

Figure 3: *Results using various texture resolutions.*

we show how different camera distances impact the resulting texture. Intuitively, closer camera distances result in smaller patterns while increased distance results in larger patterns. Finding the correct camera distance for a specific style requires some experimentation.

Texture resolution In Figure 3, we show the impact of texture resolution on the stylization.

Additionally, we tried optimizing a hierarchy of textures at different resolutions. Optimization would begin at the lowest resolution and the resolution would be gradually increased. Whenever the resolution was increased, we upsampled the current lower resolution texture to the new resolution and continued optimization. However, this provided minimal benefit in most cases and finding adequate hyperparameters became too challenging. But we believe there might be something interesting to discover if further investigations are made.

CLIP scores The CLIP score is a standard metric for evaluating how well a generated image aligns with a text prompt. It is computed by taking the cosine similarity between the computed CLIP embeddings. We evaluate it to determine how well our stylized 3D assets match the style image. We take the ViT-B/32 predicted image embedding of the style image and compare it to the embedding of the rendered image. We show the results in the table below. In terms of CLIP score, our method performs well compared to the others; ranking highest for two and on par for one of the five style images we tested. The score is computed as an average over a few different objects for each style image. However, one should be careful to draw conclusions from these results. In this case, CLIP scores are slightly flawed as a metric as our pretrained network CLIP-ResNet50 is trained on the same objective and dataset as the network used to compute the score. We deliberately select a different CLIP network, CLIP-ViT-B/32, to reduce bias but this can still lead to a preference for CLIP-based methods over VGG16. We note that stylization is still a highly subjective area. For example, certain people might prefer the geometric features of VGG NNFM over CLIP NNFM or vice versa. However, combining VGG and CLIP NNFM or using transformer-based CLIP architectures might lead to improved results.

Models	Elephant	Picasso	River	Starry	Scream
CLIP NNFM	0.676	**0.594**	**0.536**	0.565	**0.534**
VGG NNFM	0.652	0.534	0.459	0.517	0.461
CLIP Gram	**0.678**	0.549	0.497	0.554	0.528
VGG Gram	0.655	0.592	**0.536**	**0.580**	0.510

Teaser image The teaser image in the main paper, also shown in Figure 4, was created by stylizing objects individually using our CLIP-based NNFM method combined with content and color palette losses. The textures were then saved and added to the correct objects in Blender 3.0 where we layed out the scene. To account for our addition of the style texture and the original texture, we built a shader graph for each object performing this addition and adjusted some of the texture appearances through some manual tweaking of the blending. The scene was then rendered using Cycles.

The teaser unfortunately also reveals a challenge with independent optimization of objects. Varying object sizes and texture coordinates makes it difficult to get similarly sized patterns appearing on scene objects. One could instead learn to map position to texture instead and optimize entire scenes simultaneously.

References

[LHK*20] LAINE S., HELLSTEN J., KARRAS T., SEOL Y., LEHTINEN J., AILA T.: Modular primitives for high-performance differentiable rendering. *ACM Trans. Graph. 39*, 6 (nov 2020). URL: https://doi.org/10.1145/3414685.3417861, doi:10.1145/3414685.3417861. 1

[PGM*19] PASZKE A., GROSS S., MASSA F., LERER A., BRADBURY J., CHANAN G., KILLEEN T., LIN Z., GIMELSHEIN N., ANTIGA L., DESMAISON A., KOPF A., YANG E., DEVITO Z., RAISON M., TEJANI A., CHILAMKURTHY S., STEINER B., FANG L., BAI J., CHINTALA S.: Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, Wallach H., Larochelle H., Beygelzimer A., d'Alché-Buc F., Fox E., Garnett R., (Eds.). Curran Associates, Inc., 2019, pp. 8024–8035. 1

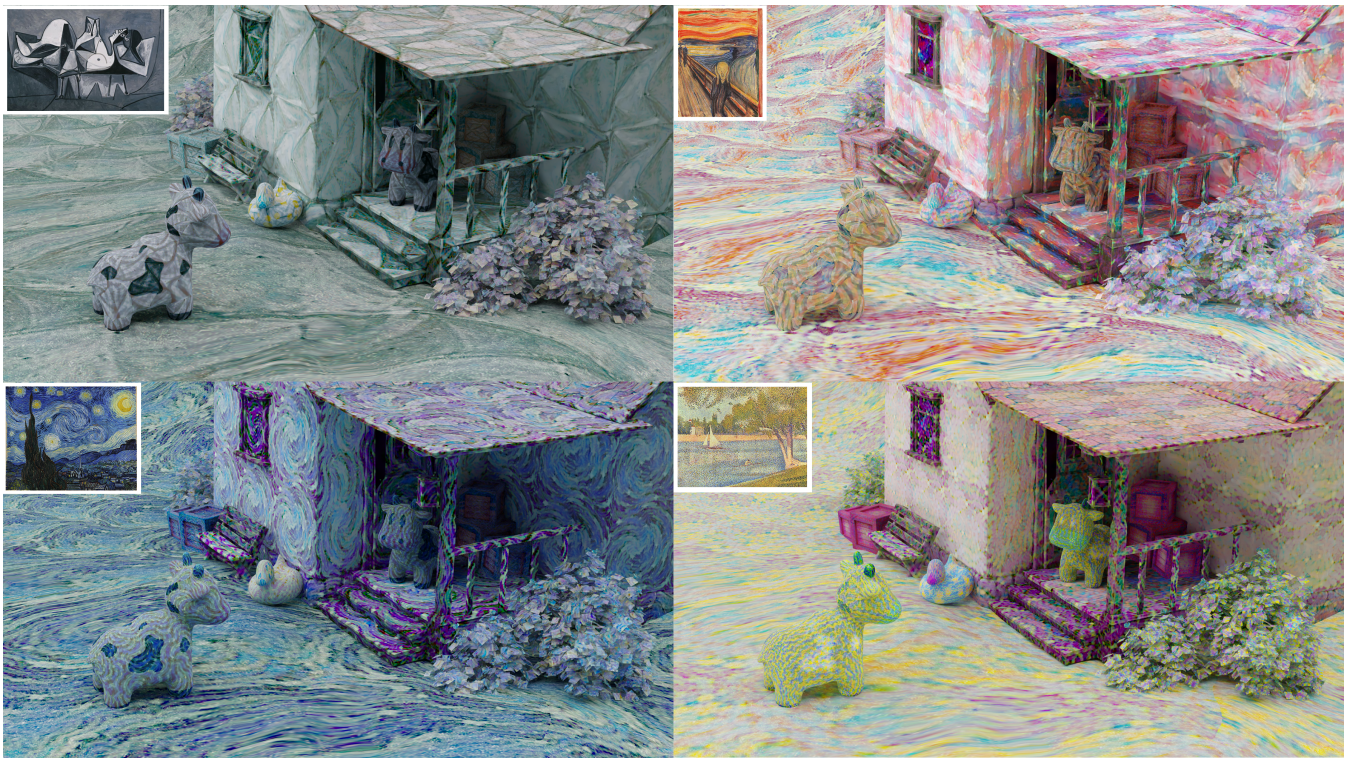

Figure 4: *Uncropped renders of the teaser figure scene.*

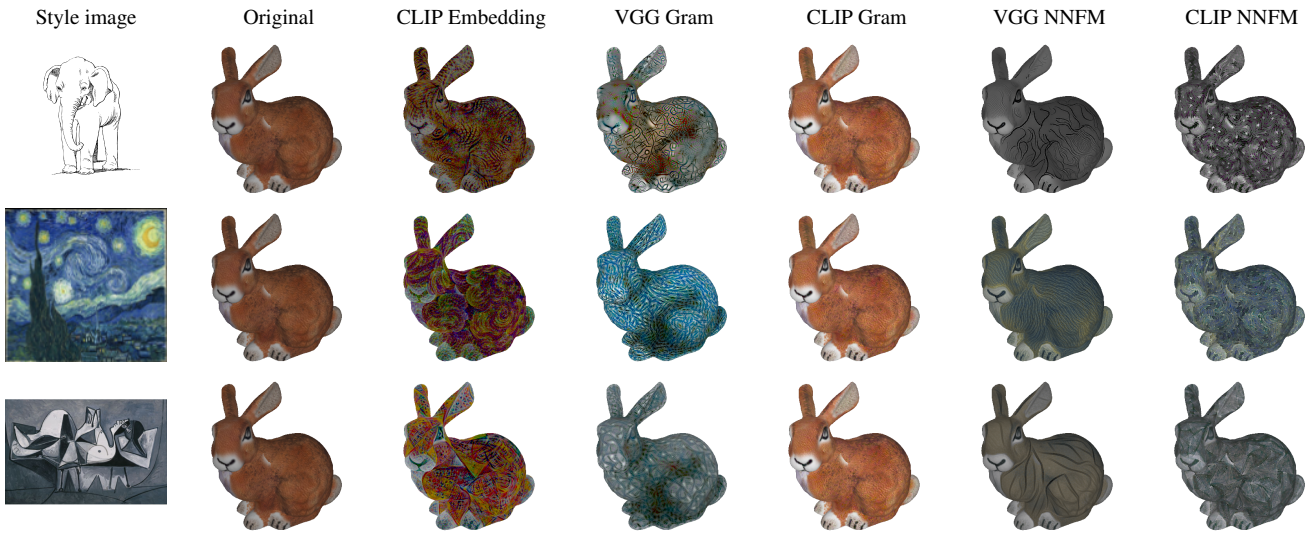

Figure 5: *Additional comparisons of different approaches on a Stanford bunny from the Stanford University Computer Graphics Laboratory. As seen here, the CLIP Gram approach did not work most of the time.*