Text2PointCloud: Text-Driven Stylization for Sparse PointCloud

Inwoo Hwang[1], Hyeonwoo Kim[1], Donggeun Lim[1], Inbum Park[1] and Young Min Kim[†][1,2]

[1]Seoul National University, Department of Electrical and Computer Engineering, South Korea
[2]Seoul National University, Interdisciplinary Program in Artificial Intelligence and INMC, South Korea

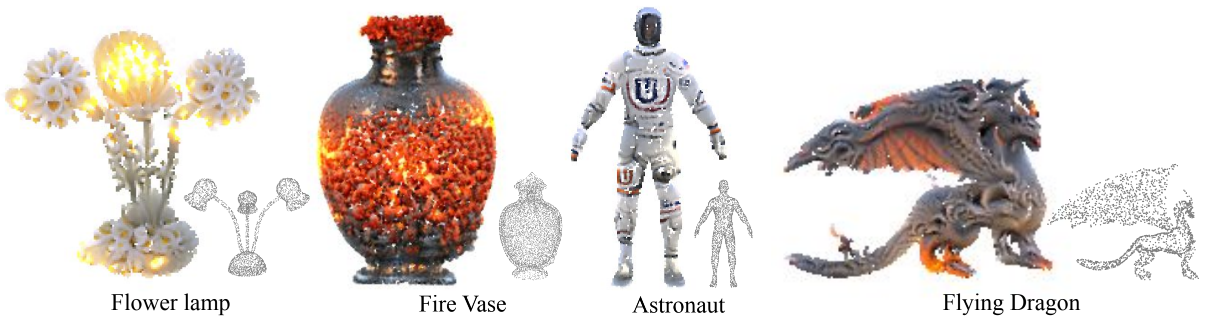

Figure 1: *From sparse, noisy point cloud and corresponding text, our method generates high-quality stylized output.*

Abstract

We present Text2PointCloud, a method to process sparse, noisy point cloud input and generate high-quality stylized output. Given point cloud data, our iterative pipeline stylizes and deforms points guided by a text description and gradually densifies the point cloud. As our framework utilizes the existing resources of image and text embedding, it does not require dedicated 3D datasets with high-quality textures, which are produced by skillful artists or high-resolution colored 3D models. Also, since we represent 3D shapes as a point cloud, we can visualize fine-grained geometric variations with a complex topology such as flowers or fire. To the best of our knowledge, it is the first approach for directly stylizing the uncolored, sparse point cloud input without converting it into a mesh or implicit representation, which might fail to express the original information in the measurements, especially when the object exhibits complex topology.

CCS Concepts
• ***Imaging and Video*** → *Computational Photography; Multi-View and 3D; Paint Systems;*

1. Introduction

In this paper, we present a method to stylize 3D objects provided as point cloud representation. 3D measurements of real-world are provided in a point cloud format, including LiDAR, RGB-D cameras, and multi-view stereo. Despite such accessibility, there are few practical frameworks to directly edit the representation with additional geometric or textural details on point cloud data. Point clouds often have to be transformed into other representations (e.g., mesh, implicit representation) to allow high-quality visualization or further geometric edits as it lacks structure or neighborhood information. However, there still exist cases where it is hard to faithfully retrieve a manifold representation of a scanned object. For example, when the object exhibits complex topologies, such as flowers, wrinkled clothes, or flames of fire, the conversion assuming a water-tight surface often fails to express and visualize the objects.

Inspired by the remarkable results of recent text-driven manipulations, we suggest a text-based tool to directly stylize point clouds which can eventually visualize complicated topology with realistic colors. Leveraging the powerful joint embeddings of large-scale images and texts provided by CLIP [RKH*21], some pioneer works [WCH*22, MBOL*22] successfully transform mesh or implicit geometries into stylized objects without requiring 3D datasets for training. We similarly stylize a noisy and sparse scan of an object without manual inputs or transforming points into other representations. Additionally, as point cloud can deform to detailed

† Young Min Kim is the corresponding author.

DOI: 10.2312/egs.20231007

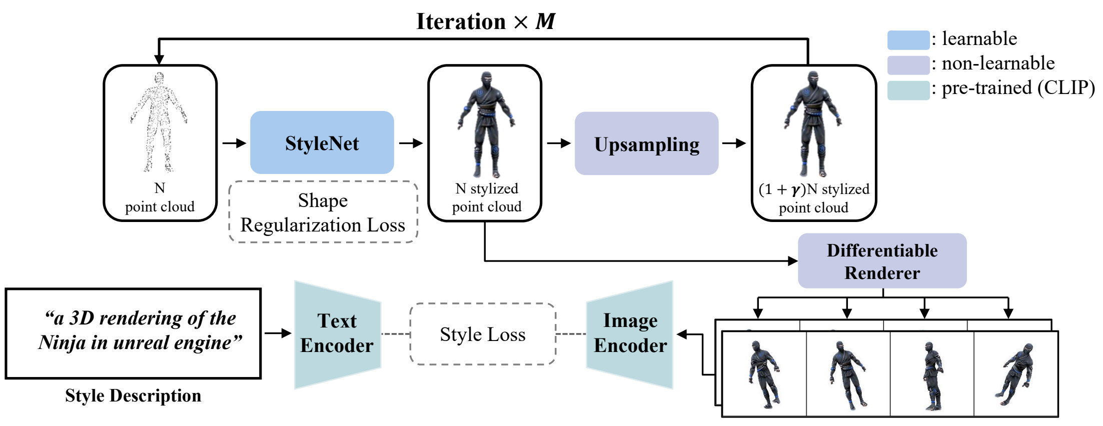

Figure 2: *Overview of our framework.*

structure variations free from manifold assumptions, our formulation can further extend the stylization into objects exhibiting complex topologies or entangled with narrow parts.

To improve the quality of style and geometry, we iterate between stylization and upsampling processes. In every iteration, the upsampling process densifies point samples and the stylization process gradually improves style by concurrently denoising possible random perturbations or upsampling artifacts of the unstructured point clouds. To briefly elaborate, we represent the style as displacement vectors and colors for individual points, which are queried from neural style fields. The neural style fields are trained to enforce style information with CLIP loss, which minimizes the feature-space distance between the input style text and rendered images. Additionally, it is trained to weakly bind the neighborhood information using additional losses, such that the results prefer regular point distribution forming locally coherent surfaces. Nonetheless, the point clouds are free from fixed topology and our framework allows non-manifold structures when preferred for the desired style.

In summary, we propose a method for the detailed stylization of point cloud scans without converting the representations. Using the large-scale language model, our method practically and easily stylizes the given point cloud and does not require a dedicated 3D dataset. Our framework can express fine-grained ornaments with complex topology thanks to the property of point cloud, and stably converges to high-quality geometry by an iterative method despite sparse and noisy input.

2. Method

Given an uncolored point cloud $P_{\text{in}} \in \mathbb{R}^{N\times 3}$ and a simple text description of the desired style T, our approach stylizes the 3D point cloud such that it can be rendered into colorful objects with fine-grained details. The pipeline iterates between stylization and upsampling, as described in Fig. 2. The stylization utilizes *StyleNet*, which is trained to deform and add colors to individual points such that together they can be visualized with style defined in the input text (Sec. 2.1). While StyleNet is updated in each iteration, the subsequent upsampling is a pre-defined procedure. Upsampling adds points respecting the current geometry of the object such that the final results eventually produce a dense rendering of the object (Sec. 2.2). After M iterations of learning StyleNet and upsampling, we finally obtain a stylized, dense geometry derived from the input.

2.1. StyleNet

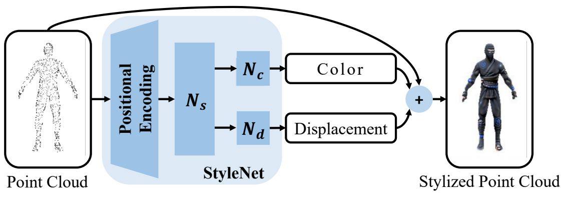

Figure 3: *Architecture of StyleNet*

StyleNet is a neural network composed of two branches, where they generate the color and displacement of each point, respectively. The overall architecture is illustrated in Fig. 3. It first applies the positional encoding of the xyz coordinates of the input points to capture high-frequency details. Similar to Text2Mesh [MBOL*22], each point p passes through a shared network N_s and then goes through separate branches that predict color N_c and displacement N_d. We assume that the input color is neutral grey $(0.5, 0.5, 0.5)$ and the displacement is initialized as zero. The output of the network adds an estimated color $c_p \in [-0.5, 0.5]^3$ to the initial gray color. Also, the displacement vector $d_p \in [-d, d]^3$ shifts the current position. At each iteration step, the range of displacement d gradually decreases to half from the initial range d_0. Together the entire point cloud is deformed with vivid colors.

2.1.1. Training Objective

For each iteration, StyleNet is trained with the combined loss $\mathcal{L}_{\text{style}} + \mathcal{L}_{\text{shape}}$, where each term is the style loss $\mathcal{L}_{\text{style}}$ and the shape regularization loss $\mathcal{L}_{\text{shape}}$, respectively. The style loss leverages the powerful CLIP feature space [RKH*21] to induce the point cloud to exhibit the desired style. Specifically, we render the point cloud P with a differentiable renderer and obtain multi-view images I. The style loss compares the similarity in the joint embedding space of the image I and text T

$$\mathcal{L}_{\text{style}}(I, T) = 1 - sim(E_1(I), E_2(T)), \tag{1}$$

where E_1 and E_2 denote the pre-trained encoders for image and text, respectively, and $sim(\mathbf{x},\mathbf{y}) = \frac{\mathbf{x}^\top \mathbf{y}}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2}$ is the cosine similarity. We boost capturing details of various scales by generating multiple images with random perspective transformation and crop augmentation. The final $\mathcal{L}_{\text{style}}$ is calculated by the mean of clip similarities of the input text compared against various images, including the multi-view renderings and augmentation.

Additionally, we apply shape regularization, which holds the points together into a coherent shape. The individual displacements, as well as the subsequent stage of upsampling, result in noisy distribution of points rather than a clean manifold surface. While we intentionally allow independent displacements to visualize shapes with complex topology, the loss eventually results in a more realistic visualization of the stylized object by encouraging the neighboring points to form a locally smooth surface. The shape regularization loss is composed of three terms:

$$\mathcal{L}_{\text{shape}}(P,P_{\text{in}}) = \lambda_{\text{uni}}\mathcal{L}_{\text{uni}}(P) + \lambda_{\text{pro}}\mathcal{L}_{\text{pro}}(P) + \lambda_{\text{in}}\mathcal{L}_{\text{in}}(P,P_{\text{in}}). \quad (2)$$

The first two terms, the uniform loss $\mathcal{L}_{\text{uni}}$ and the projection loss $\mathcal{L}_{\text{pro}}$ analyze the quality of the current point cloud being stylized P. The uniform loss encourages the point cloud to be uniformly distributed and avoids irregular or sparse samples of points [LLF*19]. Specifically, it minimizes the variation of the local point density, which is estimated by comparing the numbers of points within patches with the same projected area. The projection loss minimizes the distance from a point to the estimated local surface such that the points form a manifold-like distribution [YSW*19]. The last term of the shape regularization loss is the input loss, which maintains the deformed points close to the original input measurement. Specifically, it compares the Hausdorff distance [BLN*13], which calculates the distance of closest points between the deformed point cloud P and the input point cloud P_{in}.

Together, the shape regularization loss in Eq. (2) encourages regular samples of points that can be locally approximated as manifold surfaces such that the rendering of the points is visually pleasing. At the same time, the resulting shape does not severely deviate from the original input shape. In the result section, we demonstrate that the loss plays a critical role in faithfully stylizing the fine details using unstructured point clouds.

2.2. Non-learnable Upsampling

The upsampling module creates additional points that preserve the current distribution of points to further enhance the capacity for the subsequent stylization step. The number of points is proportional to the degree of freedom, and at the same time, it is the resolution that decides the level of expressiveness for rendering geometric details. Orthogonal to our approach, we can obtain dense and complete point samples of clean geometry by incorporating abundant previous works on point cloud upsampling or shape completion [].

For upsampling, we first sample uniform points from the current set of point P using the farthest point sampling. For each sampled point, we find the K nearest points and create new points at the centers of the K points. Although this upsampling method may generate a spray of points with noise, the subsequent StyleNet effectively corrects the distribution by modifying the displacements of points with shape regularization loss. Our upsampling method mixes the current point locations to generate additional points around them, which is very fast. However, the naïve method can deteriorate the quality of approximated geometry beyond the level that StyleNet can handle when a significant number of new points are added. Therefore, we progressively increase the number of points by γ ratio at each iteration. We demonstrate the effect of our iterative upsampling in the result.

3. Experiments

Our method can stylize and densify various sparse point cloud. For all of our experiments, we render a stylized point cloud with point cloud differentiable renderer [JRR*20]. The point cloud P is scaled to fit a unit box and is observed from a camera on a hemisphere with a radius 2.0. The viewing angles are sampled from Gaussian distribution with $\sigma = \pi/4$ around the front view of objects within the elevation angle of $[10^\circ, 80^\circ]$ and the azimuth angle of $[0^\circ, 360^\circ]$. We set $M = 5, \gamma = 0.3, d_0 = 0.05$ and $\lambda_{\text{uni}} = 1.0, \lambda_{\text{pro}} = 0.001, \lambda_{\text{in}}$=0.1 as default. The hyperparameters can be set differently depending on the degree of change desired. For all experiments, a text description is set as "a 3D rendering of the *[object]* in unreal engine", where *[object]* is replaced into the word specifying the input point cloud, as shown in Fig. 2.

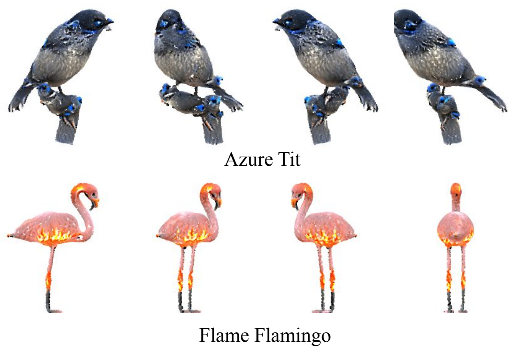

Figure 4: *Multi-view rendering results.*

Figure 1 shows samples of stylization results of diverse objects using our method. The input point cloud and the specified objects of the text prompt are in the figure for reference. The input point cloud is 4000 uncolored points sampled from meshes collected from existing 3D datasets [Tur22, ZJ16]. Our method is capable of generating fine-grained geometric variations that possess complex topologies. Concurrently, we densify point cloud and generate local details which reflect text descriptions while preserving original geometric information. Our final output contains about 15000 colored points, which can be rendered to show a stylized 3D object. Figure 4 shows multi-view renderings of the stylized outputs.

The benefits of using the point cloud are prominent when we stylize objects to exhibit complicated topology. We compare our stylization result against [MBOL*22] where the initial 3D model is in mesh representation in Fig. 5. While the fixed connectivity of mesh inherently results in the predefined manifold structure, our

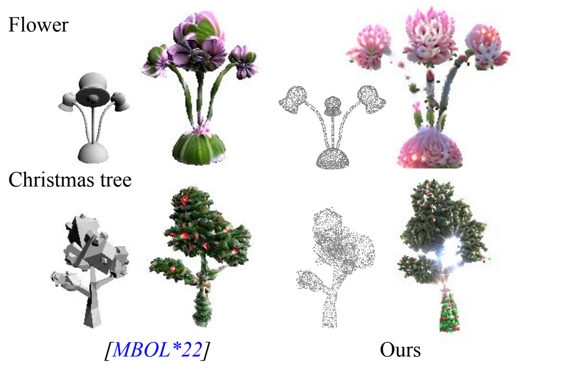

Figure 5: *Visual comparison with Text2Mesh [MBOL*22].*

representation is free to exhibit diverse topology of the surface details while maintaining locally smooth surfaces. Additionally, our explicit point cloud representation enables faster rendering compared to implicit representations.

3.1. Ablation Study

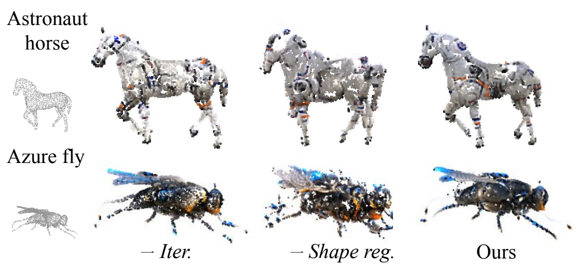

Figure 6: *Results of ablation study.*

While it is difficult to faithfully generate object geometry from unordered point cloud, our pipeline adapts iterative upsampling and shape regularization for simple yet effective edits on the point cloud without converting the representation. We compare the visual quality of stylized objects against ablated versions in Fig. 6. By comparing against the results without the iterative upsampling pipeline (*-iter.*), we can deduce that our progressive upsampling module effectively creates dense points of the sparse input, which improves the visual quality. Also, without the shape regularization loss for StyleNet (*-shape reg.*), the overall shape of the object is severally deteriorated as the points largely deviate from plausible geometry.

3.2. Results on Incomplete Data

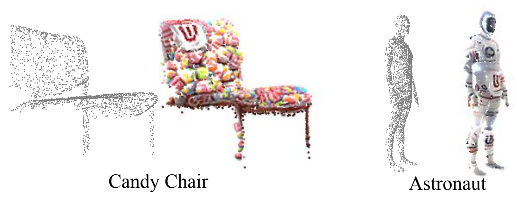

Figure 7: *Experiments on incomplete data.*

Because our method directly stylizes point cloud input, it can be applied to real-world 3D measurements. Figure 7 contains the stylization results on real scanned data and manually mangled incomplete data. The input data for the left consists of noisy, sparse real scanned points [UPH*19], and the right consists of partial points from the front side of the human [VRM*17]. Starting from the corrupted data, our method can still generate faithful styles. Note that our upsampling module merely creates spurious points near the boundaries of the missing parts, and we largely rely on the shape regularization loss for high-fidelity details.

4. Conclusions

In this paper, we introduce Text2Pointcloud, a framework to stylize and upsample the uncolored, sparse point cloud such that we can render high-quality 3D rendering of the given text description. While the point cloud can handle a wide variety of geometric details without connectivity constraints, it is challenging to maintain a geometrically plausible structure. Our pipeline iterates between gradually adding points and stylizing the object with the shape regularization loss. As a result, we can easily create dense samples of objects that exhibit diverse and sophisticated decorations.

References

[BLN*13] BERGER M., LEVINE J. A., NONATO L. G., TAUBIN G., SILVA C. T.: A benchmark for surface reconstruction. *ACM Trans. Graph. 32* (2013), 20:1–20:17. 3

[JRR*20] JOHNSON J., RAVI N., REIZENSTEIN J., NOVOTNY D., TULSIANI S., LASSNER C., BRANSON S.: Accelerating 3d deep learning with pytorch3d. In *SIGGRAPH Asia 2020 Courses* (New York, NY, USA, 2020), SA '20. 3

[LLF*19] LI R., LI X., FU C.-W., COHEN-OR D., HENG P.-A.: Pu-gan: a point cloud upsampling adversarial network. In *IEEE International Conference on Computer Vision (ICCV)* (2019). 3

[MBOL*22] MICHEL O., BAR-ON R., LIU R., BENAIM S., HANOCKA R.: Text2mesh: Text-driven neural stylization for meshes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2022). 1, 2, 3, 4

[RKH*21] RADFORD A., KIM J. W., HALLACY C., RAMESH A., GOH G., AGARWAL S., SASTRY G., ASKELL A., MISHKIN P., CLARK J., KRUEGER G., SUTSKEVER I.: Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning* (18–24 Jul 2021), vol. 139 of *Proceedings of Machine Learning Research*, PMLR. 1, 2

[Tur22] TURBOSQUID: Turbosquid 3d model repository. In *https://www.turbosquid.com/* (2022). 3

[UPH*19] UY M. A., PHAM Q.-H., HUA B.-S., NGUYEN D. T., YEUNG S.-K.: Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *International Conference on Computer Vision (ICCV)* (2019). 4

[VRM*17] VAROL G., ROMERO J., MARTIN X., MAHMOOD N., BLACK M. J., LAPTEV I., SCHMID C.: Learning from synthetic humans. In *CVPR* (2017). 4

[WCH*22] WANG C., CHAI M., HE M., CHEN D., LIAO J.: Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2022). 1

[YSW*19] YIFAN W., SERENA F., WU S., ÖZTIRELI C., SORKINE-HORNUNG O.: Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH ASIA) 38*, 6 (2019). 3

[ZJ16] ZHOU Q., JACOBSON A.: Thingi10k: A dataset of 10, 000 3d-printing models. *ArXiv abs/1605.04797* (2016). 3