View Synthesis via Sculpted Neural Points

Julien Gaubil, julien.gaubil@ens-paris-saclay.fr
M2 MVA, ENS Paris-Saclay
NPM3D 2022-2023 – Project Report

1. Introduction

The article View Synthesis via Sculpted Neural Points [7] tackles the problem of view synthesis. This consists in generating observations of a scene from unseen view points given input images of the scene from different view points. It uses a Point-Cloud based approach: unlike many recent works on Neural Radiance Fields (NeRFs) [5] that use an implicit representation of the scene, this work uses an explicit representation that is a Point Cloud. This point cloud is then featurized to enable rendering.

In NeRF-based approaches, a set of 3D points is generated by marching camera rays through the scene. These locations, along with view directions, are then provided to a neural network that outputs colors and volume density. Volume rendering is then applied to these points and their predicted colors and volume densities to render the scene. It accumulates the color and volume density along a ray to render a pixel. Such a process is computationally costly since it requires a lot of points to render an image, while the 3D space is mostly unoccupied. Using an explicit point cloud representation enables to be computationally more efficient by only considering points that are close to the actual surfaces. It also enables fine-grained scene editing, which is hard with NeRFs.

In this project, we will study the Sculpted Neural Points approach proposed in [7]. We will first describe it and then evaluate by performing experiments and ablations. Finally, we will discuss the limits identified through our study and will point out potential improvements.

2. Sculpted Neural Points

Notations: A set of elements $\{a_1, ..., a_n\}$ will be denoted as $a_{1:n}$ for simplicity.

Overview The approach consists in creating a point cloud representation of the scene from a set of M images $I_{1:M}$ via a Multi-View Stereo System (MVS). The MVS however produces a lot of errors. To correct such errors, this work adopts a two-step approach that is called Sculpted Neural Points. First, points are removed in a step of point pruning according to a photometric consistency criterion. This first step can result in large holes created in the point cloud. To overcome this issue, a second step of point adding is used, that adds points that will help improving the rendering of the point cloud in a global way that enables to fill important holes. This approach enables to increase the robustness of the representation to errors and large holes.

The Point Cloud obtained is then featurized and its features are optimized by gradient descent to minimize a rendering loss. The rendering is point-based, executed in a differentiable way.

MVS System The MVS system used is CER-MVS [3]. It predicts depth maps for every input image $I_{1:M}$ and then merges them in a fusion step to create the point cloud. Its original version uses a geometric consistency check, Dynamic Consistency Checking [6] to eliminate outliers. SNP however uses the raw depth maps and imposes its own consistency criterion that is photometric and therefore more suitable to the downstream task.

Point pruning During the point pruning task, points are removed by imposing a photometric consistency criterion. Its intuition consists in keeping a point if it is not significantly closer to any of the source views than the original surface. Let Π be the projection that maps a 3D point P and the parameters of a camera C to the corresponding pixel p in this view, and Π^{-1} be the mapping between a pixel p, its depth d_p, the camera parameters of the view C and to its corresponding 3D point P. A pixel p from the depth map of the source view i is kept if the following condition is met:

$$\bigcap_{m=1}^{M} \left(D^m(\Pi^{-1}(p, d_p^i, C^i)) \geq \delta_d . d_q^m \right) \tag{1}$$

where $\Pi^{-1}(p, d_p^i, C^i)$ is the 3D point with depth d_p^i in view i predicted by the MVS corresponding to the pixel p. D^m is the depth (i.e. third coordinate) of this 3D point in the referential of the view m, q its corresponding pixel in the view m and d_q^j the depth predicted by the MVS for q in the depth map of the view m. δ_d is an hyperparameter that handles the tolerance of the criterion.

This enables to filter points floating freely between the cameras and the surface, and can keep points that are seen only in a limited number of views, which is important since the number of source views M is relatively low.

Features and rendering The point cloud is featurized i.e. a learnable feature $f_p \in \mathbb{R}^D$ and a learnable opacity parameter $o_p \in \mathbb{R}$ are attached to every point p. Given a view v to synthetize, these features are transformed by spherical harmonics of degree 2 that modelize view-dependant effects. The spherical harmonics features of every points are then converted into a 2D feature map by soft-rasterization [2] that mixes the features of points aligned on a same ray by weighting them with their depth and opacity. The 2D feature map is then fed to a U-Net to generate the RGB image for the view v. The whole pipeline is differentiable and parameterized.

Point adding The goal of the point adding step is to find a set of points $\mathcal{X}$ such that after optimising the features of the point cloud: $PC = PC_{pruned} \cup \mathcal{X}$, the photometric error decreases compared to only optimizing on PC_{pruned}. This procedure is done in two steps:

1. The features of the pruned point cloud PC_{pruned} are optimized with gradient descent with a rendering loss: $L = \sum_{m=1}^{M} \left(\|I_m - I_m^{render}\|_1 + TV(I_m^{render}) \right)$ penalized with a total variation penalty to help generalizing to novel views. After optimization, an error map $E_m = |I_m - I_m^{render}|$ can then be obtained for each source view m.

2. For each view m, if a pixel of index (u, v) has an error $E_m(u, v) \geq \delta$ (δ being a fixed threshold) K points are sampled uniformly on the camera-pixel ray through the scene. Only the points that do not occlude any of the points of PC_{pruned} in any view are kept.

The features of the point cloud obtained with these additional points are then optimized with the same objective as in step 1 to render the source images $I_{1:M}$. The point cloud obtained after adding points can therefore be a superset of the original point cloud, since its features are further optimized.

Technical tricks SNP uses a few technical tricks that enable to boost its performances. First, they introduce point dropout layers in order to boost the generalization capacities of the model. Second, they remove Batch Normalizations in the U-Net that are useless in this case of small number of images.

3. Experiments

We will evaluate the training procedure on three different setups: the first does not use point adding and point pruning, the second only uses point pruning and the last uses both. We use the same training settings as the default Sculpted Neural Points [7]: the optimizer AdamW with a learning rate of 0.00025, a point dropout rate of 0.5 and two a two-layer U-Net.

We perform experimentations on datasets DTU [1] and LLFF [4] datasets and evaluate our results with metrics PSNR, SSIM and LPIPS. We train SNP from scratch on a scene of LLFF, performing 5000 steps for optimization of features for point adding and 50 000 steps for optimizing features for the final training. We train our models on a single Nvidia V100, which lasts around 12 hours in total training time.

3.1. Results

3.1.1 Point Cloud visualization

We first start by visualizing the effects of the point sculpting procedure on point cloud reconstructed from the depth maps from the MVS. To do so, we use a scene of the dataset DTU (the scan 24) that contains a single object with a close surface, which enables to see better the effect of the procedure on the point cloud. We evaluate the following settings:

- point cloud created from the depth maps obtained from the MVS that are not refined by any consistency check (neither DCC, used in [3] nor point pruning) and no point adding is performed: 6 479 996 points,
- point cloud created after performing point pruning only: 4 411 851 points,
- point cloud created using point pruning and point adding: 4 546 223 points.

We first compare the point pruning procedure with the raw point cloud to visualize the points removed, figure 1;

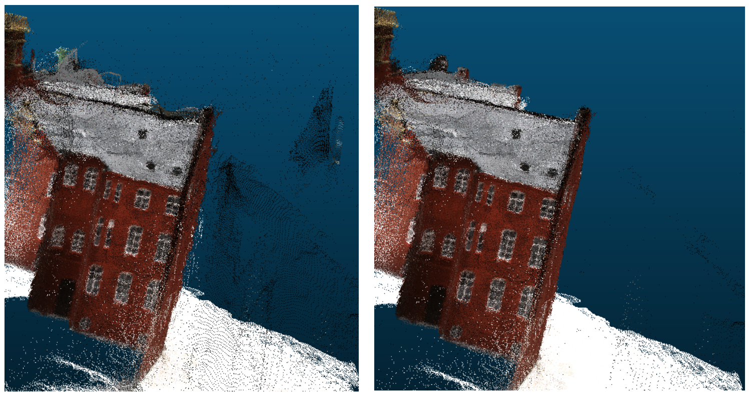

Figure 1: Free outlier points removed by point pruning (right) compared to raw point cloud (left).

We can observe that the point pruning indeed enabled to remove outliers points created by the MVS system. These points (in black on the right part of the left image) were floating freely between the surface and camera views and were accurately removed from the point cloud without loosing details on the surface. This illustrates the capacity of the consistency check that relies on the criterion 1 to remove such outliers.

The source views in the training set only cover a part of the architecture, which leads to an open surface representation, almost half of the object not being displayed in any views. The figure 2 details the effect of point pruning on a zone with low point density corresponding to parts of the object present in few views:

The right part of the images represents a zone of the scene that is seen in very few views, at the limit of the unseen parts of the object. This implies a low point density on this part of the scene, which could lead to these points being treated as outlier points freely floating between the cameras and the surface. This example shows that the criterion defined indeed enables to keep these points in the pruned point cloud. There are indeed very few qualitative differences in this low density zone between the pruned and the raw point cloud.

Finally, we compare the pruned point cloud and the pruned point cloud with point adding to visualize (in red) the zones where points are added, figure 3:

Although no significant hole has been created by point pruning, the structure of the point cloud is still incomplete due to missing parts of the scene in the training views. The points added on this figure show that the point adding procedure focuses on these parts of the scene by adding points that fill the inside of the structure. This example is interesting since these points are not really removed from the actual surface by the point pruning procedure.

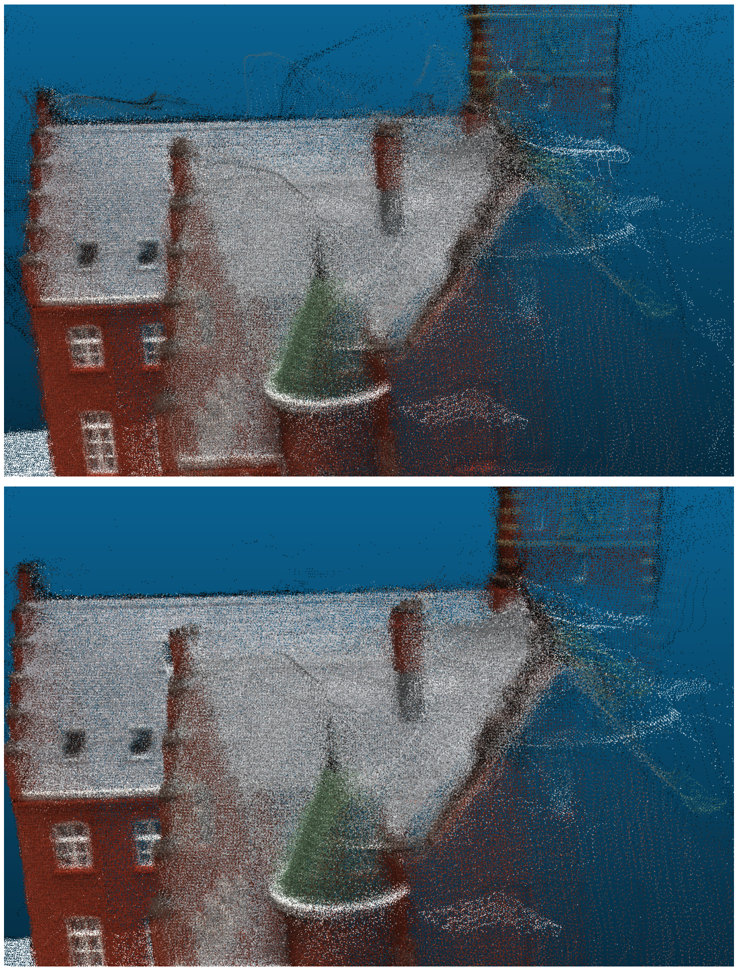
Figure 2: Pruning procedure (right) keeps points that are seen in few source views in the training images compared to raw point cloud (left).

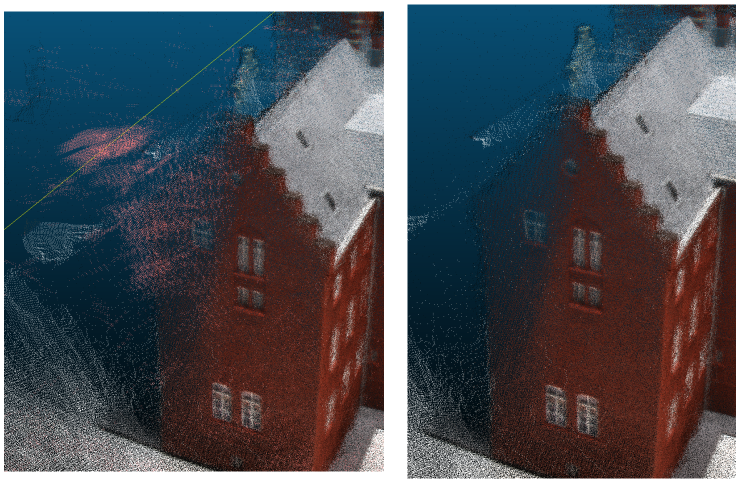
Figure 3: Points added in unseen zones after pruning in red (left) compared to pruned point cloud (right).

The point adding procedure here plays another role that is filling the missing parts of the representation. Although it can be counter-balanced by the following optimization of the features attached to each point, we can therefore wonder to which extent point adding brings hallucinations of unseen parts of the scene.

3.1.2 Entire training

We now evaluate the point adding method on a scene from LLFF dataset that contains more variability in lightning and more objects than DTU scenes. We first evaluate the same setups as in the quantitative section: raw point clouds, point pruning and point pruning + point adding. We stress out that due to lack of computation ressources, we were able to perform experiments on a single scene only, while training on different scenes would have been interesting to have an exhaustive evaluation of the model on different conditions. We presentthe quantitative results for these different setups table 1:

Point sculpting method	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$
Raw	0.847	0.248	25.85
Point Pruning	**0.852**	**0.245**	26.09
Point Pruning+Adding	**0.852**	0.247	**26.13**

Table 1: Quantitative evaluation on View Synthesis on scan Horns of dataset LLFF.

This table shows that on this specific scene, the effect of point adding and point pruning are almost negligible on quantitative metrics. Similarly to results reported in the original article [7], we find non-significant differences between sculpting pipeline using only point pruning and using pruning + adding. However, unlike the results of the article, we do not observe a significant improvement when adding the point pruning consistency check over using the raw depth maps. These quantitative results are coherent with the view synthesis results displayed figure 4. Visualizing the point clouds for this example shows that the removed outliers using point pruning are mostly noisy envelops of the surfaces on the scene, which can be further corrected by feature optimization. This could explain the lack of difference between the two setups in both quantitative and qualitative evaluation when training from scratch. Another possible explanation for this lack of difference would be the use of Point Dropout that would enable to correct the use of these outliers for rendering when using the raw depth maps.

Other claims from the paper are supported by our experiments. First, non-Lambertian effects are displayed on the view synthesis figure 4, which authors attribute to the use of Spherical Harmonic functions. We also measure a total rendering time around 0.2s for a single image, which is on par with the 5 FPS inference time presented by the authors.

4. Discussion

From our experiments, it seems that the raw depth maps inferred by the Multi-View Stereo System are good enough to be directly used without further refining. The examples of point clouds displayed illustrate this behaviour by showing that point pruning does not automatically creates holes on point clouds. It also shows that in the case of scans with unseen parts of the

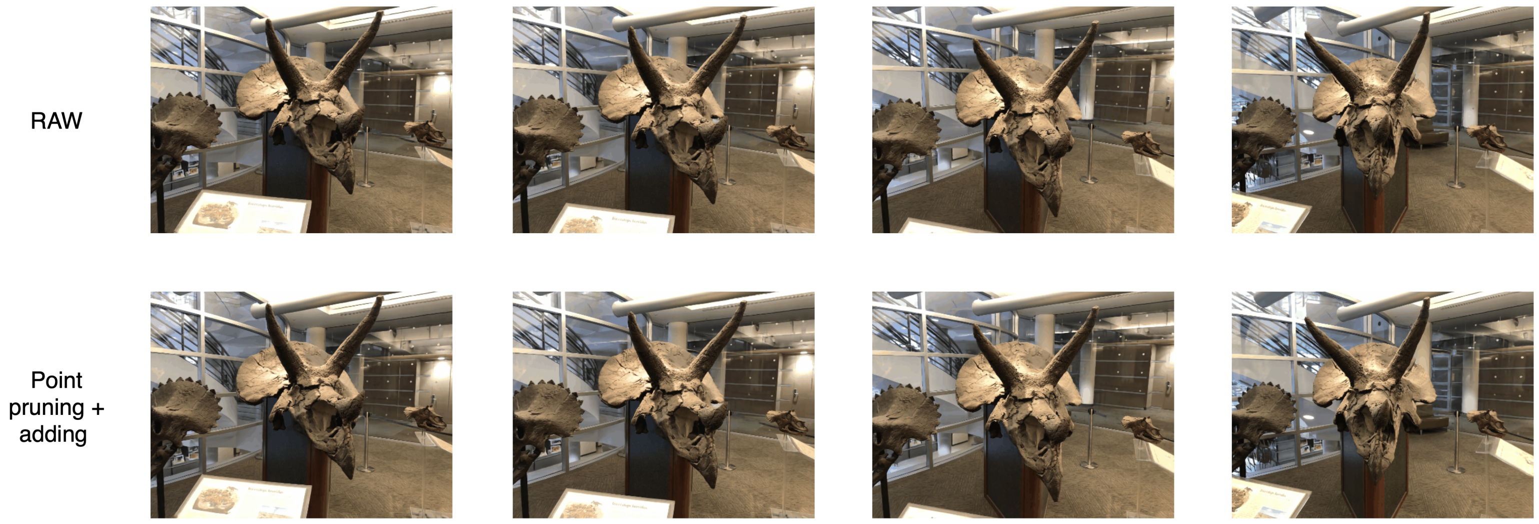

Figure 4: View synthesis on Horns scan of LLFF dataset, achieving almost similar results for sculpting pipeline using raw depth maps and SNP sculpting pipeline.

scenes, the point adding step can hallucinate points in the empty unseen zones instead of covering points removal due to pruning on the actual surfaces. This behaviour is useless for view synthesis and can even translate into hallucinations. A simplified pipeline without Point Sculpting and point adding could therefore be used, which would both save training time and hyperparameters tuning.

Authors also state a limitation of this approach that lies in the explicit point cloud representation: the model is unable to deal with scenes with background textures that are arbitrarily far away from the camera (e.g. sky). A solution could be to explicitly model the background in the point cloud representation by sampling points on a sphere that contains the whole scene. This could enable to have a background model and to model objects at the infinite, but would in turn probably increase by a significant factor the number of parameters, which could slow down inference and training.

Finally, authors highlight the lack of 3D consistency of their method that is due to the hallucinations of the U-Net. This results in flickering effects that are visible on the videos in the Github associated to the code for this project. They leave as a future work to remove this dependency to a 2D generator, which would probably require to use a different rendering pipeline and would imply important changes in the architecture.

5. Conclusion

Our experiments enabled to confirm interesting properties of the point-based view synthesis model Sculpted Neural Points, such as the efficiency of the point pruning procedure to remove outliers produced by the MVS system, or the capture of non-Lambertian effects. However, they did not demonstrate the effects of one of the main mechanisms of the approach, that is sculpting neural points. Using this procedure indeed did not translate into significant improvements on the specific scene on which we have been able to perform our experiments.

References

[1] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 406–413, 2014. 2

[2] Christoph Lassner and Michael Zollhofer. Pulsar: Efficient sphere-based neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1440–1449, June 2021. 1

[3] Zeyu Ma, Zachary Teed, and Jia Deng. Multiview stereo with cascaded epipolar raft. In *Proceedings of the European conference on computer vision (ECCV)*, 2022. 1, 2

[4] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Trans. Graph.*, 38(4), jul 2019. 2

[5] 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. 1

[6] Jianfeng Yan, Zizhuang Wei, Hongwei Yi, Mingyu Ding, Runze Zhang, Yisong Chen, Guoping Wang, and Yu-Wing Tai. Dense hybrid recurrent multi-view stereo net with dynamic consistency checking. In *ECCV*, 2020. 1

[7] Yiming Zuo and Jia Deng. View synthesis with sculpted neural points. *arXiv preprint arXiv:2205.05869*, 2022. 1, 2, 3