EUROGRAPHICS 2022 / R. Chaine and M. H. Kim
(Guest Editors)

Volume 41 (2022), Number 2

Supplementary Material for Z2P: Instant Visualization of Point Clouds

G. Metzer[1], R. Hanocka[4], R. Giryes[1], N. J. Mitra[2,3], D. Cohen-Or[1]

[1]Tel Aviv University
[2] University College London
[3] Adobe Research
[4] University of Chicago

1. Experiments and Evaluations

We evaluate the ability of our method to cope with point clouds at different scales, which results in a different point density on the z-buffer image. In Figure 1, we show that our method provides the most consistent results across the different methods we compared to.

1.1. Fine Details

One limitation of our method is the attention to fine details, this is due to three main reasons. First, representing shapes as points is inherently prone to miss fine details, as points are a type of discrete sampling of the continuous surface. This can be seen in the surface reconstruction results in Figure 2, compared to the ground truth mesh. Second, convolutional neural networks tend to output smooth results, this effect is also encouraged by the MSE loss our network is trained with, as this loss aims at achieving the *average* visualization result. Third, the natural shapes our method was trained on mainly contain low frequency surfaces, while high frequency fine details are rare. In Figure 3 we test our method on four different spheres. Each sphere is modulated with spherical harmonics with increasing frequency. The increasing frequency of the spherical harmonics represents details with decreasing granularity, and demonstrate our method's robustness towards fine details. As expected, when the frequency is high enough, our method visualizes the sphere as a smooth surface due to the above reasons.

2. Architecture

Our image2image model is a modified U-NET fully convolutional network. The modifications are as follows:

AdaIN layers. We swap out the batch-norm layers in favor of AdaIN layers. Through the AdaIN layers we inject the rendering controls such as color and light position. First, the control vector is mapped through the *mapping network* to a latent representation of size 512. Then, the latent representation is passed through a **learnable** affine transformations that maps the 512 latent representation to the AdaIN primary input feature size, which depends on the depth of the network as well as the number of convolution kernels in the previous layer. The affine transformation is a required such that the resulting vector can be passed as a secondary AdaIN parameter dimension-wise. This flow is depicted in Figure 5.

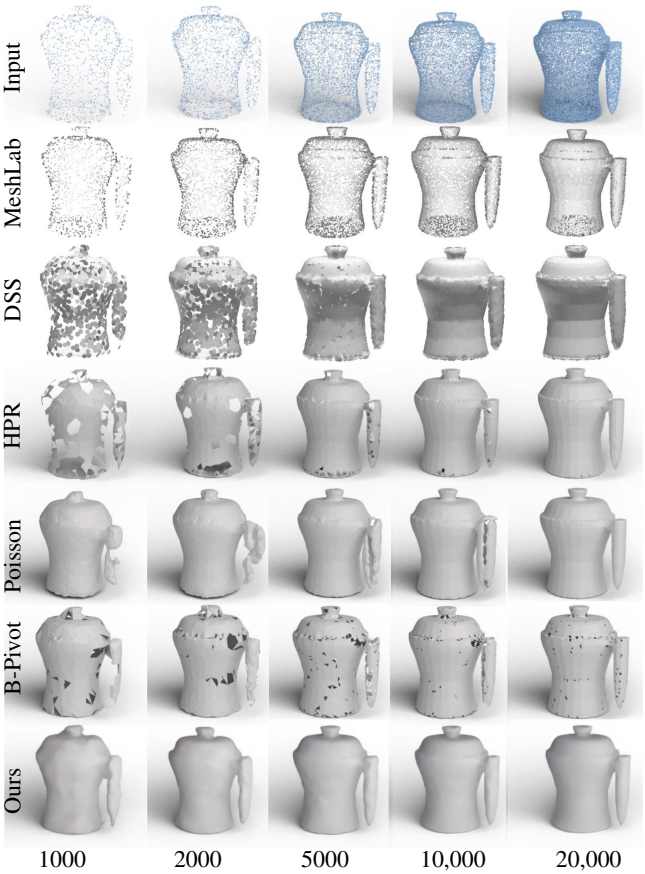

Figure 1: *Robustness to number of point samples. For an increasing number of point samples in the point cloud, our method is most consistent in the final rendered result.*

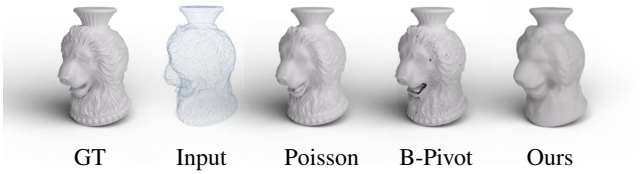

Figure 2: *Comparison on the Lion head model. Fine details are smoothed out, a similar effect as seen in Figure 3.*

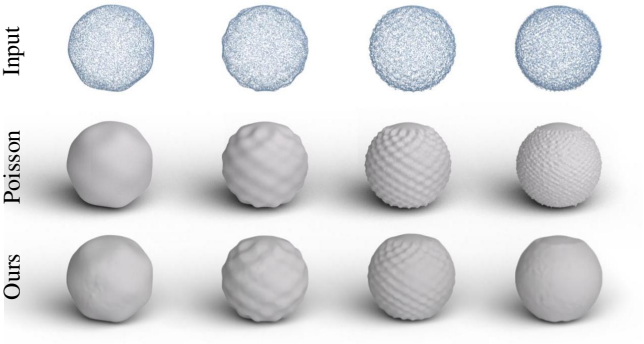

Figure 3: *Results on point clouds with varying amounts of details, to demonstrate detail robustness limitation. Each sphere is modulated with spherical harmonics with increasing frequency. As the frequency increases, more details smooth out, up until the highest frequency where the harmonic modulation is not visible in our visualization.*

Mapping Network. The mapping network is a simple fully connected network, that maps between the input control vector to a latent representation of size 512. The control vector is nominally of size 6 *i.e.* 3 elements for color in RGB format, and 3 elements for light position in the form of r, θ, ϕ with respect to the camera. In the extra material control experiment in the paper, the control vector was extended to be of size 8, with two extra input features between $0-1$ for *Metallic* and *Roughness* attributes.

Positional Encoding. We also include extra input features to each pixel, that correspond to the pixel's position in the input image. This requires making the input feature size larger, specifically

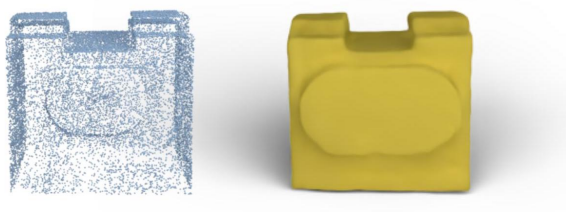

Figure 4: *Our result on a real Lidar scanned cube, scan provided by AIM@SHAPE-VISIONAIR.*

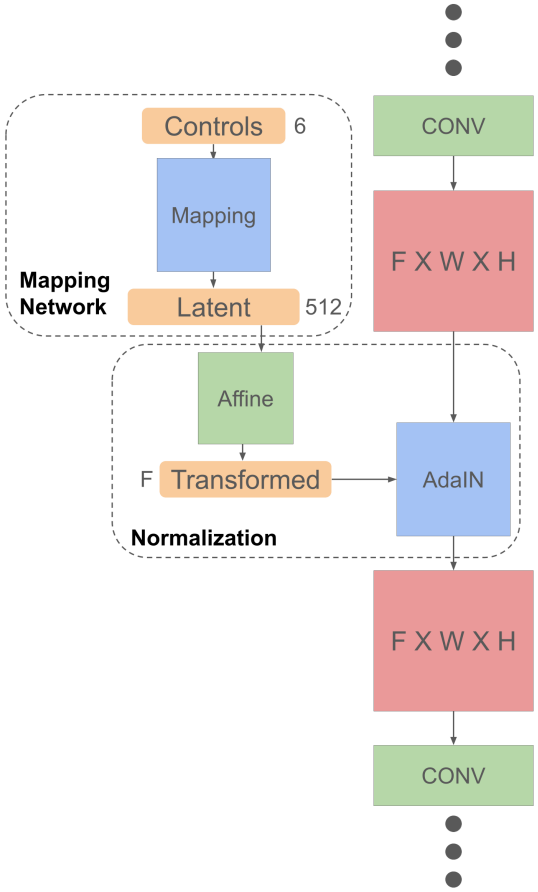

Figure 5: *Diagram depicting the flow of information from the controls vector, through the mapping and affine layers, and lastly to the AdaIN layer. Each normalization layer has a different target feature size, determined by the previous convolution layer, and accordingly a matching affine mapping layer to that target size.*

the number of positional encoding arguments plus one for the point z-buffer.

3. Training Details

Our model is implemented in Pytorch. Both training and testing were done on a single Nvidia RTX 2080 GPU. Training was performed with the ADAM [KB14] optimizer, and a learning-rate of $2e-4$.

References

[HMGC20] HANOCKA, RANA, METZER, GAL, GIRYES, RAJA, and COHEN-OR, DANIEL. "Point2Mesh: A Self-Prior for Deformable Meshes". *ACM Trans. Graph.* 39.4 (July 2020). ISSN: 0730-0301. DOI: 10.1145/3386569.3392415. URL: https://doi.org/10.1145/3386569.3392415 3.

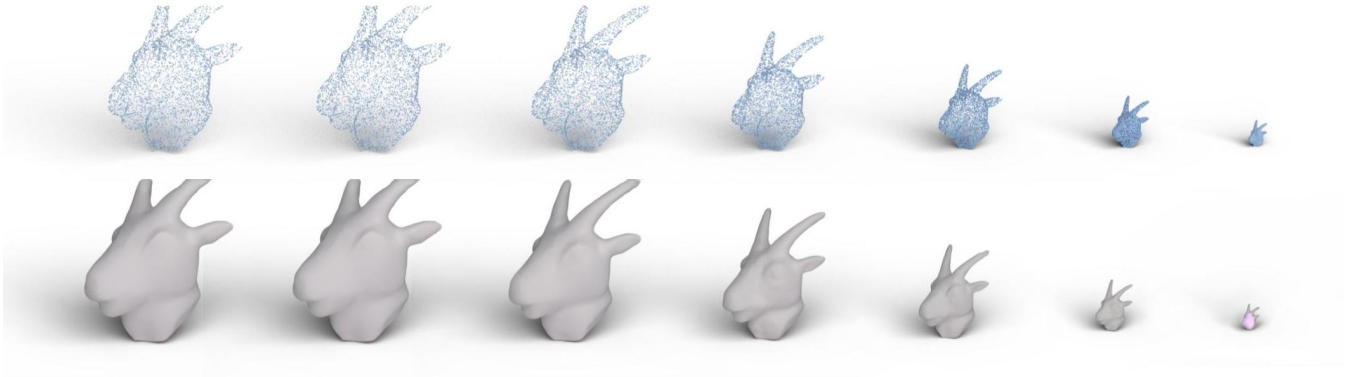

Figure 6: *Our method rendering results at different scales of the input. Note that our method was trained on meshes normalized to a single scale.*

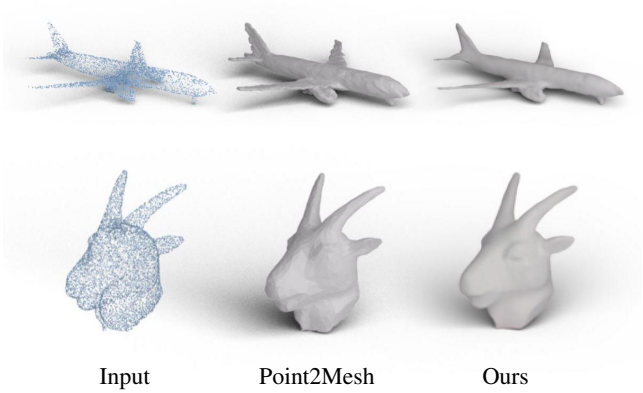

Figure 7: *Comparison to [HMGC20]. As point to mesh is also able to produce good results, it requires a long time to converge (30-60 minutes), compared to our result that is obtained in seconds.*

Figure 8: *Results of rendering different planes and cars generated by [YHH*19].*

[KB14] KINGMA, DIEDERIK P and BA, JIMMY. "Adam: A method for stochastic optimization". *arXiv preprint arXiv:1412.6980* (2014) 2.

[YHH*19] YANG, GUANDAO, HUANG, XUN, HAO, ZEKUN, et al. "Pointflow: 3d point cloud generation with continuous normalizing flows". *Proceedings of the IEEE International Conference on Computer Vision*. 2019, 4541–4550 3.

[YKH*18] YUAN, WENTAO, KHOT, TEJAS, HELD, DAVID, et al. "Pcn: Point completion network". *2018 International Conference on 3D Vision (3DV)*. IEEE. 2018, 728–737 4.

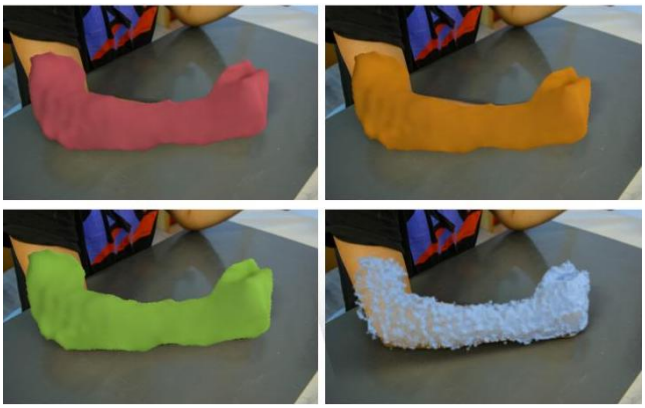

Figure 9: *As our network is able to render single view point clouds as they were meshes fast, it can be used together with a lidar scanner to visualize a reconstruction of a scan live. In this project, human hands were scanned with a lidar scanner to make specialized casts. Our method allows for a fast visualization of the cast in different color for the patient to choose from. In blue are the actual points that were scanned from the arm.*

Figure 10: *As our network predicts an alpha map that models shadows for each image, we can blend the generated results with arbitrary backgrounds. The point cloud in this figure was generated by [YKH*18].*

Figure 11: *Sweep over the metallic and roughness controls, compared to the ground truth rendering of the mesh with blender.*

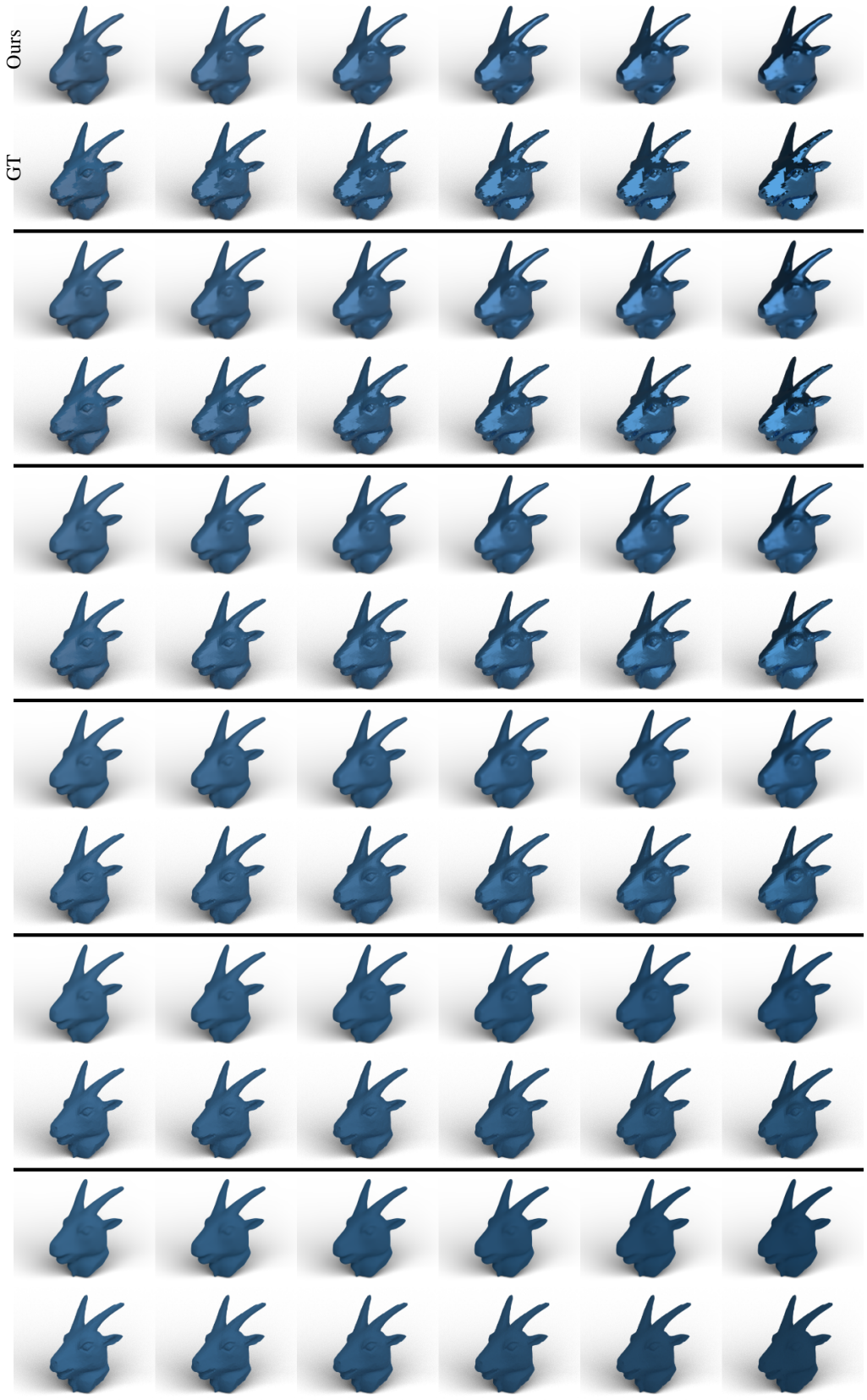

Figure 12: *Sweep over the metallic and roughness controls, compared to the ground truth rendering of the mesh with blender.*

Figure 13: *Examples of the dataset used for training our method. The dataset is composed of these meshes only, rendered from different views, shaded with different colors, and lit from different directions. The corresponding point cloud z-buffer, which is used as input to the network, is visualized next to each rendered target mesh. Even though our framework was only trained on these meshes, it is able to generalize and work on the various different types of point cloud shown in the paper.*