6 Supplementary Material

6.1 Normal and principal curvature estimation performance

Performance on real data. We qualitatively evaluate the performance of our method on the NYU Depth V2 dataset [19]. This dataset was captured using a Kinect v1 RGBD camera and contains indoor scene environment and includes missing data and a noise pattern that is significantly different than the PCPNet dataset. Specifically, the noise often has the same magnitude as some of the features. Most importantly, this dataset, much like other real-world datasets, does not have ground truth normals. Fig. 8 and Fig. 9 show the performance of DeepFit's normal and principal curvature estimation respectively compared to Jet. DeepFit was trained with 256 points, however, since the network's weights are shared between the points it can be used with any neighborhood size. In these results we show the performance for $128, 256, 512, 1024$ neighboring points. It shows that DeepFit is less sensitive to noise and is able to overcome the over-smoothing affect commonly attributed to using a large neighborhood while also preserving fine details.

We additionally qualittively evaluate the performance of our method on the KITTI dataset [8]. This dataset was captured using a Velodyne laser scanner mounted on a driving car.

Additional normal estimation results. We evaluate the normal estimation performance on the PCPNet datast using the percentage of good points (PGP α) metric. Fig 11 shows the results of different learning based methods for increasing α values . It shows that for low and medium noise levels, DeepFit is comparable to Lenssen et.al. [15] while in all other categories their performance is better. This is most likely attributed to the dataset bias towards flat and low curvature surfaces, in which case, our method does not pose an advantage. DeepFit main advantage is in curvy surfaces where an n-jet yields a better fit than a plane.

We evaluate DeepFit's normal estimation performance using RMSE for different n-jet orders and number of points in the neighborhood. The results are shown in Fig. 12. It shows that the increase in the number of neighboring points slightly decreases the performance in the no noise augmentation however it significantly improves the performance in high noise. This is mainly attributed to the weight estimation network that softly selects the most relevant points for the fit. It also shows that 1-jet (planes) perform well, however higher order jets have an advantage in the low and medium noise augmentation categories. In theory, the higher order jets have the capacity to fit planes, however in practice it is not always the case.

Fig. 13 depicts a visualization of DeepFit's results on PCPNet point clouds. Here the normal vectors are mapped to the RGB cube. Fig. 14 depicts a visualization of the angular error in each point for the PCPNet dataset. Here, the points' color correspond to angular difference, mapped to a heatmap ranging from 0-60 degrees. It shows that for complex shapes with high noise levels, the general direction of the normal vector is predicted correctly, but, the fine details

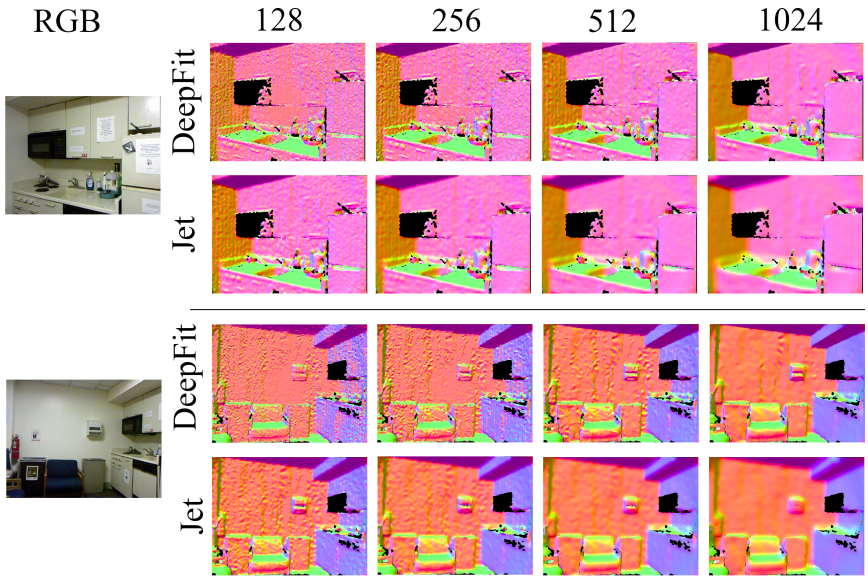


Fig. 8: Normal estimation results for DeepFit and Jet on NYU Depth V2 dataset for different neighborhoods sizes $(128, 256, 512, 1024)$. The colors of the points are normal vectors mapped to RGB and projected to the image plane.

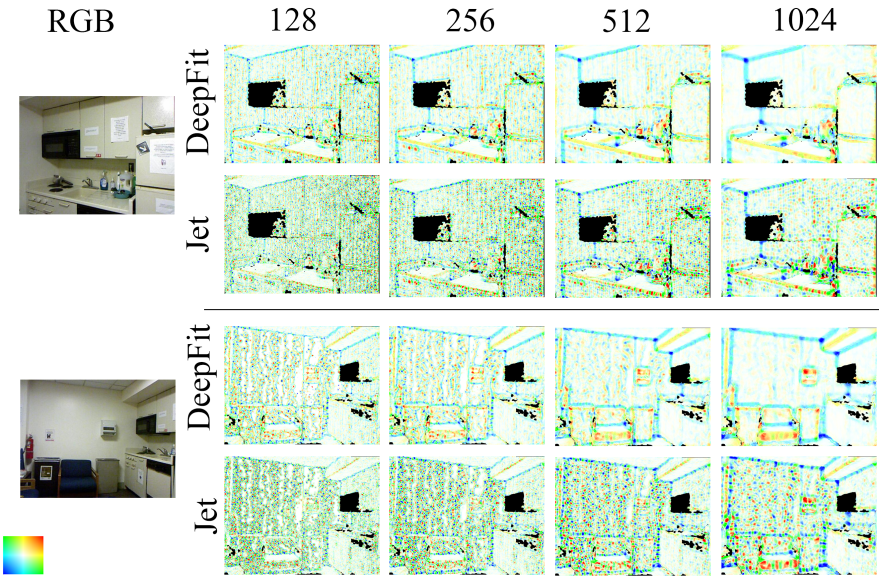
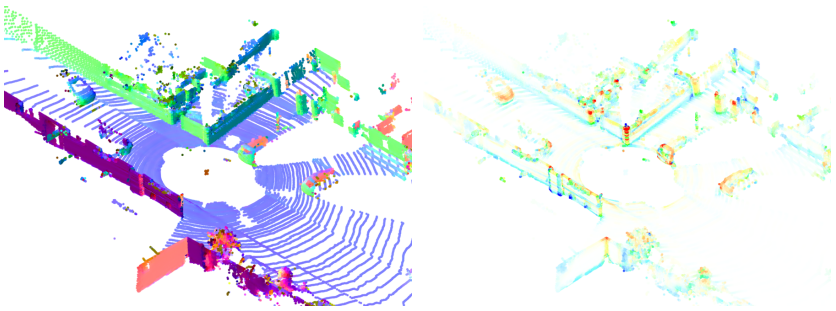


Fig. 9: Principal curvature estimation results for DeepFit and Jet on NYU Depth V2 dataset for different neighborhoods sizes $(128, 256, 512, 1024)$. The colors of the points correspond to their principal curvature values using the colormap in the bottom-left corner and projected to the image plane.



(a) (b)

Fig. 10: (a) Normal estimation results for DeepFit on KITTI dataset. The colors of the points are normal vectors mapped to RGB and projected to the image plane. (b) Principal curvature estimation results for DeepFit on KITTI dataset. The colors of the points correspond to their principal curvature values.

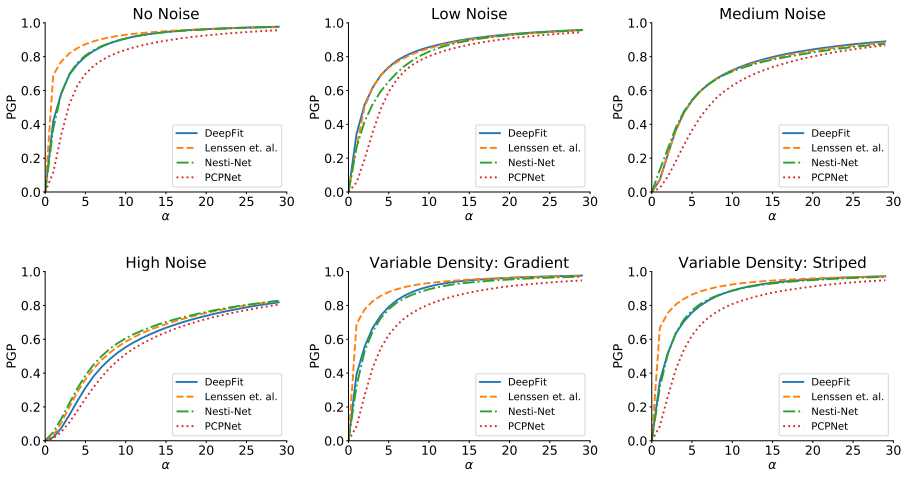


Fig. 11: Comparison of the percentage of good points (PGP) metric for unoriented normal estimation of the proposed DeepFit to other deep learning methods (PCPNet [10], Nesti-Net [3], Lenssen et. al. [15]). Here, α is the threshold for measuring the percentage of good points.

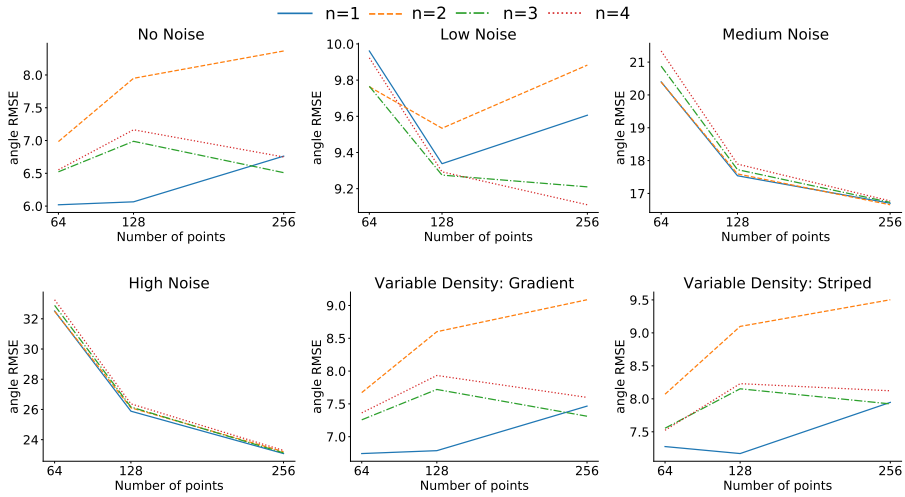


Fig. 12: Comparison of the angle RMSE metric for different DeepFit variants. Ablations include different n-jet order $(1, 2, 3, 4)$ and number of neighboring points $(64, 128, 256)$.

and exact normal vector are not obtained. For basic shapes the added noise does not affect the results substantially. Most notably, DeepFit shows robustness to point density corruptions.

Additional principal curvature estimation results. Fig. 15 qualitatively depicts DeepFit's results on the PCPNet dataset. For visualization, the principal curvatures are mapped to RGB values according to the commonly used mapping given in Fig. 5 i.e. both positive (dome) are red, both negative (bowl) are blue, one positive and one negative (saddle) are green, both zero (plane) are white, and one zero and one positive/negative (cylinder) are yellow/cyan. For consistency in color saturation we map each model differently according to the mean and standard deviation of the principal curvatures. Note that the curvature sign is determined by the ground truth normal orientation. DeepFit's normalized RMSE metric is visualized in Fig. 16 as the magnitude of the error vector mapped to a heatmap. It can be seen that more errors occur near edges, corners and small regions with a lot of detail and high curvature. Moreover, these visualizations show that for low noise levels, the principal curvature estimation is reliable, as expected, the reliability declines with the insertion of high magnitude noise.

6.2 Efficiency

Table 3 shows a comparison between the number of parameters and run time between different deep learning based normal estimation methods. It can be

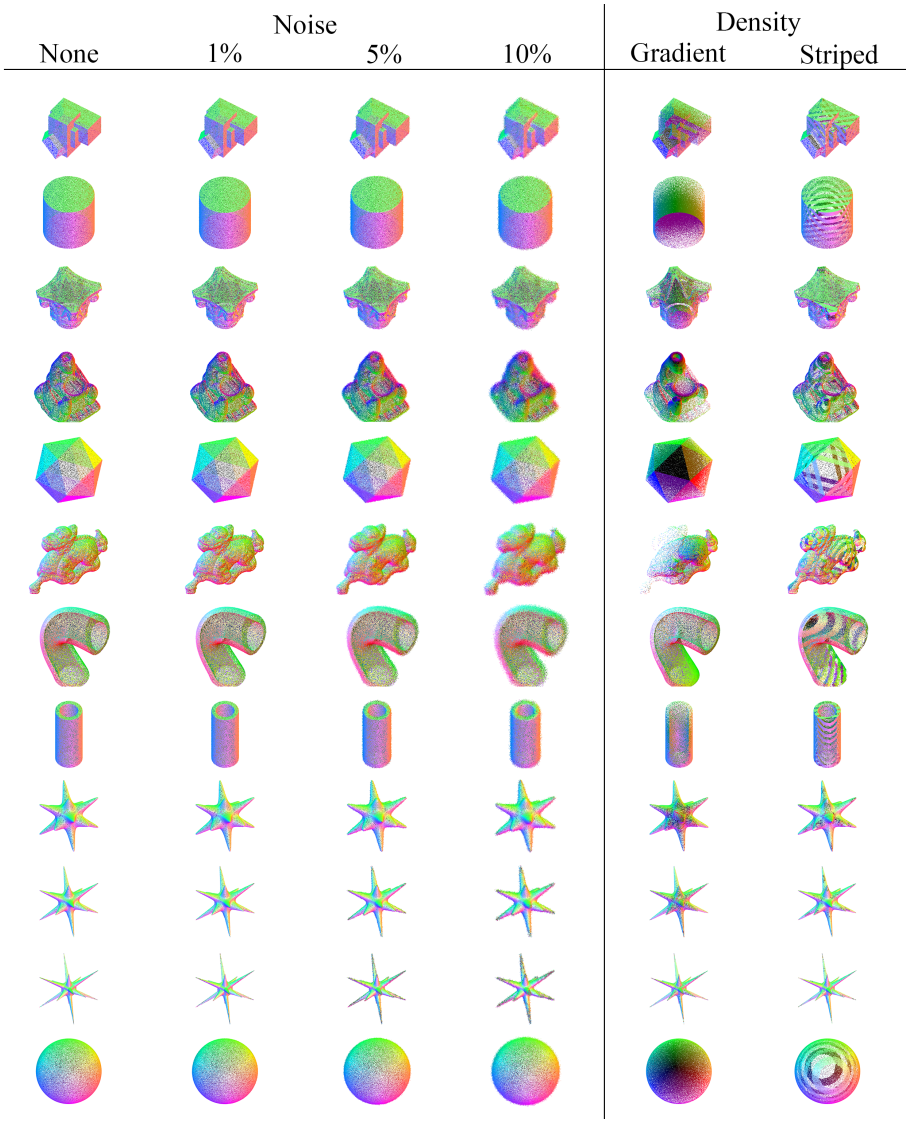


Fig. 13: DeepFit's normal estimation results for different noise levels (columns 1-4), and density distortions (columns 5-6). The colors of the points are normal vectors mapped to RGB.

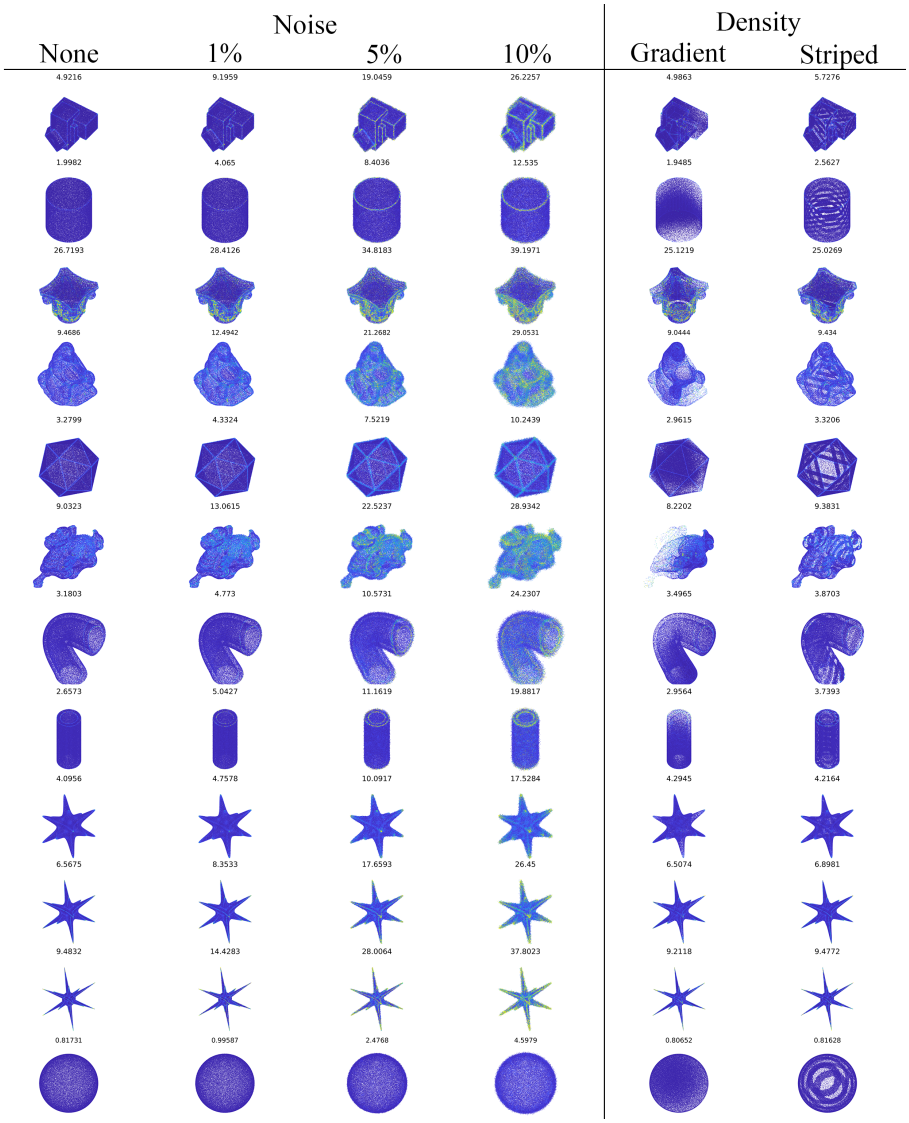


Fig. 14: Normal estimation error visualization for different noise levels (columns 1-4), and density distortions (columns 5-6). The colors of the points correspond to angular difference, mapped to a heatmap ranging from 0-60 degrees. The number above each point cloud is the RMSE.

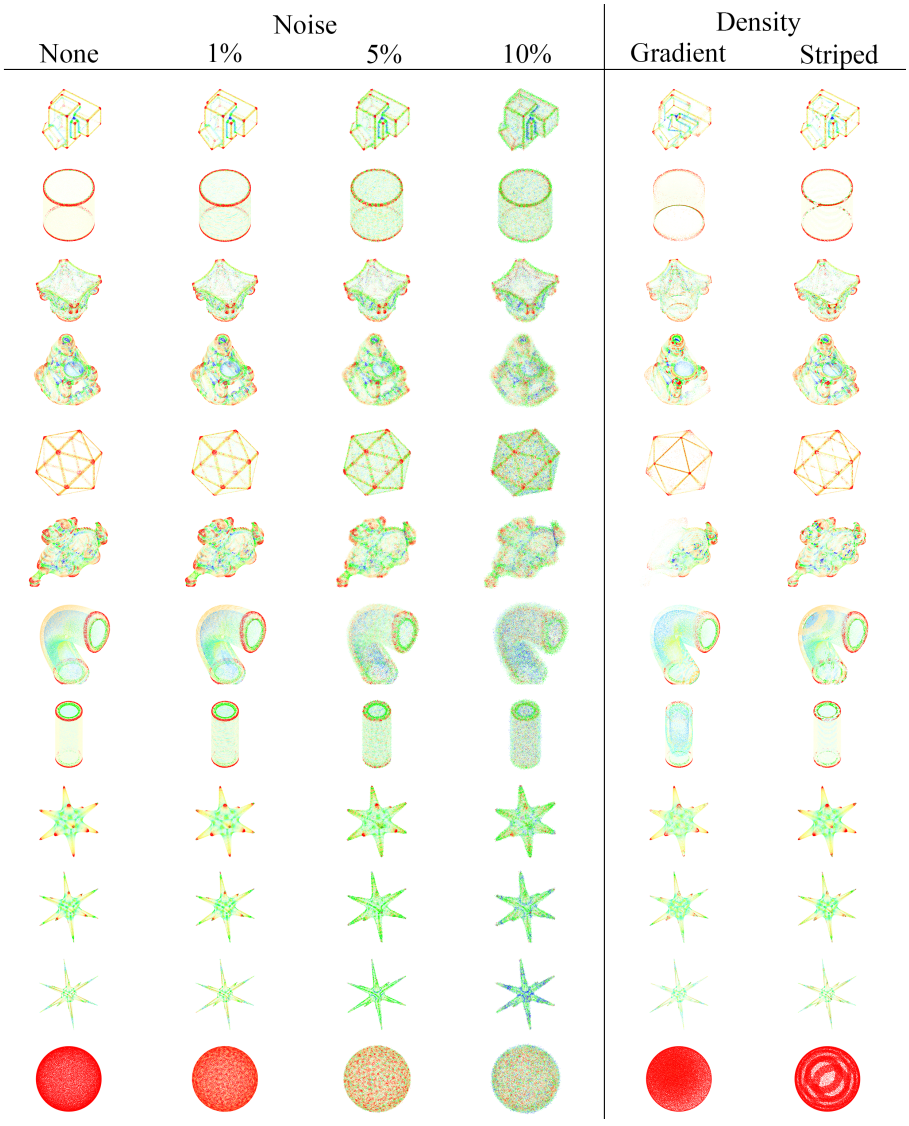


Fig. 15: Curvature estimation results visualization. The colors of the points corresponds to the mapping of k_1, k_2 to the color map given in Fig 5. Values in the range $[-(\mu(|k_i|) + \sigma(|k_i|)), \mu(|k_i|) + \sigma(|k_i|)]|_{i=1,2}$.

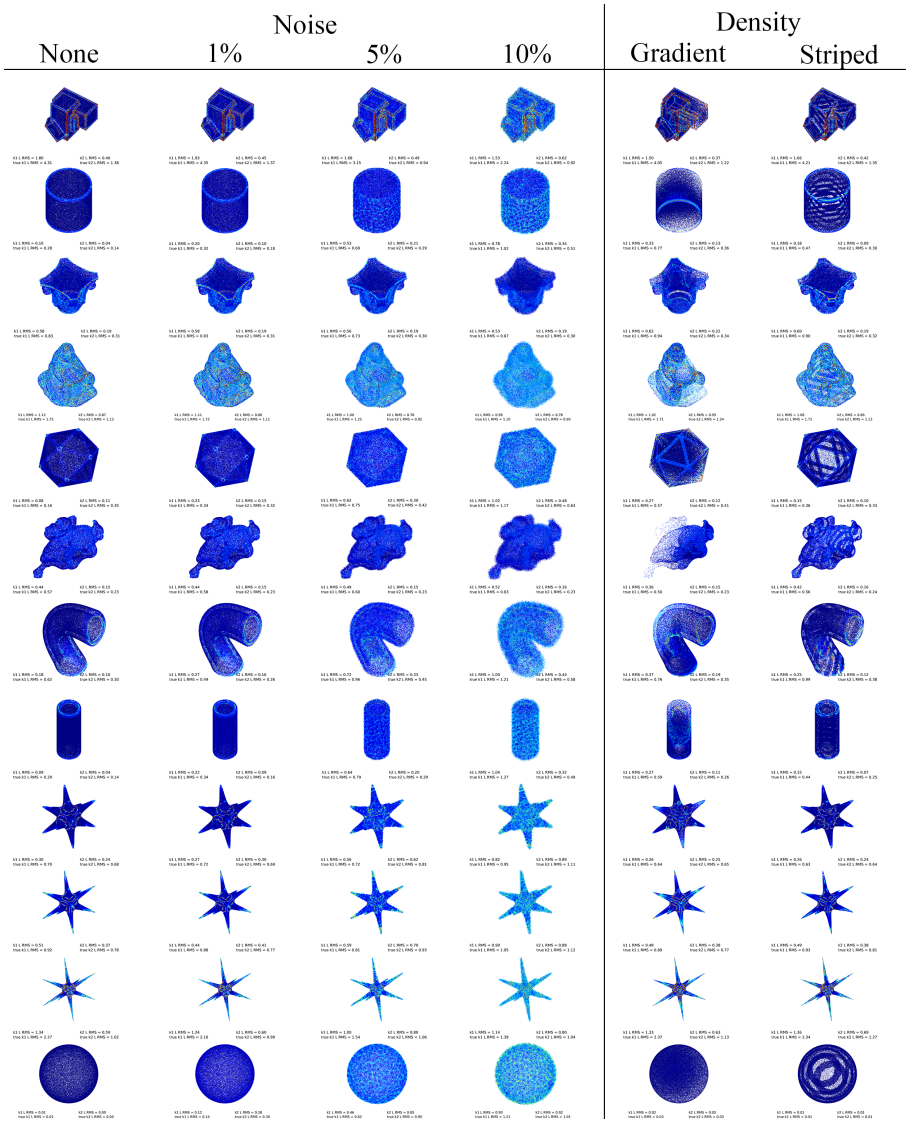


Fig. 16: Curvature estimation error results. The numbers under each point cloud are its RMSE and normalized RMSE. The color corresponds to the L2 norm of the error vector mapped to a heatmap ranging from 0-5.

seen that DeepFit has a significantly lower number of parameters compared to Nesti-Net and PCPNet and more parameters than Lenssen et. al.. This gap in the number of parameters can be explained by the lack of point structure in our method while Lessen et. al. construct a graph. Constructing a graph introduces a limitation with respects to the number of neighboring points, i.e. training and testing has to be done on the same neighborhood size, using a PointNet architecture allows to train and test on different sizes of neighborhoods. DeepFit's , number of parameters is mainly attributed to the PointNet transformation subnetworks. The reported run time is the average run time for a batch of size 64 (i.e. computing normals for 64 points simultaneously). We chose a batch of 64 in order to fairly compare to the more resource intensive methods (Nesti-Net). Most methods, including ours, can compute in larger batches for faster performance, particularly Lenssen et. al. that are able to fit a full size point cloud (100k points) in a single batch on the GPU.

Method	Our DeepFit	Nesti-Net [3]	PCPNet [10]	Lenssen et. al. [15]
Parameters	3.5M	179M	22M	7981
Exec time (per point)	0.35ms	266ms	0.61ms	0.13ms

Table 3: Number of parameters and execution time performance for deep learning normals estimation methods. Run time is averaged for batches of size 64.