

Learning-based Estimation of Surface Normals in Unstructured Airborne LiDAR Point Clouds

Max Hermann^{1,2}, Martin Weinmann²

¹ Fraunhofer IOSB, Karlsruhe, Germany

² Institute of Photogrammetry and Remote Sensing, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany

Keywords: LiDAR, Point cloud, Surface reconstruction, Surface normals, Deep learning

Abstract

Estimating surface normals in airborne LiDAR point clouds is a critical step in mesh reconstruction and other downstream tasks such as semantic segmentation. Conventional methods often rely on local surface fitting combined with heuristic orientation strategies, which require careful parameter tuning and often yield inconsistent results, either globally, as in Minimum-Spanning-Trees, or locally on vertical structures, as in global axis-based orientation. In this work, we present two deep learning approaches based on a Point Transformer V3 encoder architecture for normal estimation in unstructured airborne LiDAR point clouds. The first approach treats normal orientation as a binary classification, flipping pre-computed normals to achieve consistent orientations. The second approach directly predicts the full normal vector. Both approaches estimate the results for all points simultaneously, which enables efficient processing of large-scale point clouds. Since no aerial LiDAR dataset provides reference normals for direct evaluation, we employ an indirect evaluation strategy based on Poisson surface reconstruction to assess the quality of the meshes generated from the estimated normals. Our method for full normal estimation outperforms the strongest baseline by up to 2 percentage points in terms of F1-Score, as shown by experiments with three reference LiDAR datasets: UseGeo, Hessigheim 3D, and our holdout dataset JB3D. However, our qualitative results also highlight the limitations of this approach. Since the training data consists of point clouds derived from photogrammetric point clouds and point clouds sampled from textured meshes, our approach for full normal vector estimation tends to result in noticeable smoothed edges.

1. Introduction

Reconstructing scene geometry is essential for many applications, including the creation of digital twins, preserving cultural heritage, visualizations in augmented and virtual reality, or surveying large outdoor areas. Established technologies include stereo cameras for indoor environments, photogrammetric multi-view stereo (MVS) for large-scale aerial reconstructions, and LiDAR for highly accurate 3D scanning. LiDAR data can be acquired either with stationary instruments like Terrestrial Laser Scanners (TLS) or from mobile mapping platforms such as robots or unmanned aerial vehicles (UAVs). Recorded data typically comprises spatial 3D coordinates, intensity of the backscattered signal, and return counts. In the following, we focus mainly on outdoor UAV-based LiDAR scanning.

To produce suitable 3D models for downstream tasks, point clouds are often triangulated to reconstruct a triangle mesh, which first requires estimating normal vectors that define the local surface orientation. Because normals are not directly measured during laser scanning, they are often estimated in post-processing using two steps: (1) selecting a neighborhood around each point and fitting a local surface function, and (2) orienting the resulting normal to distinguish inside from outside. Larger local neighborhoods often yield more consistent normals by averaging the surface but can smooth out sharp discontinuities. The normal vector is not only important for triangulating a point cloud but also facilitates other downstream tasks such as semantic segmentation in point clouds, since the normal vectors as additional input provide important information about the direction and orientation of the surface, thereby making it possible to distinguish between interior and exterior areas.

In this paper, we present two deep-learning approaches for air-

borne LiDAR scans aimed at improving the estimation of normal vectors. The first approach estimates only the orientation of a given pre-computed normal vector, while the second approach directly predicts the entire normal vector. Since there are no airborne LiDAR datasets that contain normal vectors for direct evaluation, we use an indirect evaluation method by first reconstructing a mesh from the point clouds using the estimated normal vectors and then assessing its quality compared to the LiDAR scans. Our main contributions are as follows:

- We propose two deep learning approaches for surface normal estimation in airborne LiDAR point clouds: one that predicts only the orientation of pre-computed normals, and one that directly estimates the full normal vector. Both approaches producing globally and locally consistent results without per scene parameter tuning.
- Unlike prior learning-based methods, our approach leverages a Point Transformer V3 (Wu et al., 2024) encoder to predict normals for all points simultaneously, enabling efficient processing of large-scale outdoor scenes.
- We utilize an indirect evaluation routine based on Poisson surface reconstruction (Kazhdan et al., 2006), addressing the lack of ground-truth normal datasets for airborne LiDAR, and demonstrate improvements of up to 2 percentage points in F1-Score over traditional methods while achieving significantly faster runtimes.

This paper is structured as follows. In Section 2, we present the current state-of-the-art and summarize related work. We then describe our general approach in Section 3 and how the reference methods are parameterized. Next, we describe the experimental setup with the employed datasets in Section 4 and we furthermore present both quantitative and qualitative results,

which are briefly discussed in Section 5. Finally, in Section 6, we summarize the results and offer an outlook on possible future work.

2. Related work

Estimating the normal vector, which describes the surface properties of the local neighborhood of a point, is a prerequisite for many applications. Traditional approaches rely on fitting a local surface, such as a plane for planar geometries or a quadratic function for curved surfaces (Hoppe et al., 1992). The size of the neighborhood is a critical factor, as it makes the result robust against local noise on the one hand, but also has a strong smoothing effect, which leads to errors, particularly at discontinuities. More robust approaches rely on local triangulation of using Voronoi diagrams (Dey et al., 2005; Amenta and Bern, 1998) or Delaunay triangulation (Dey and Goswami, 2004), but this can be computationally very expensive for large point clouds.

For orientation, various methods attempt to estimate the inside versus outside direction by propagating consistent normal orientations along the surface (König and Gumhold, 2009; Xie et al., 2003). For surface scans containing holes and occlusions, typical for airborne LiDAR heuristics like Minimum-Spanning-Trees (MST) (Hoppe et al., 1992) or global flips towards one major coordinate axis are often used. While Minimum-Spanning-Trees tend to produce locally consistent results, they can misorient entire regions. In contrast, flipping all normals towards the up-vector results in globally reasonable orientations, but may cause visible errors on vertical surfaces like building facades. Overall, these traditional approaches share the common limitation, that they rely on handcrafted heuristics that have to be adapted to the specific characteristics of different scene types or sensor configurations. The choice of neighborhood size, the propagation strategy for orientation, and the handling of complex geometries all require manual parameter tuning and domain expertise, often leading to suboptimal results when applied to diverse real-world datasets.

These limitations have motivated the exploration of learning-based methods, which can potentially capture complex geometric relationships directly from data. Deep learning has become increasingly prevalent in point cloud processing, offering the ability to learn task-specific features without explicit geometric modeling. While early approaches such as PointNet were initially only capable of extracting global features (Qi et al., 2017a), this capability has been expanded to local features later through hierarchical clustering in PointNet++ (Qi et al., 2017b). While these approaches are primarily based on multilayer perceptrons (MLPs), subsequent approaches such as KPConv rely on learned convolution kernels (Thomas et al., 2019). Coming from natural language processing, transformer-based models for point cloud data like shown by Zhao et al. (2021) and Wu et al. (2022, 2024) achieve highly competitive results by converting a point cloud into a sequence through curve fitting in 3D space. Based on these general network architectures, there are also specialized variants for estimating local geometries, such as surface curvature or normal vectors. Guerrero et al. (2018) demonstrate with PCPNet that information about local geometric properties can be extracted even from noisy point cloud data. Their model relies on a modified PointNet encoder and processes RGBD data. Ben-Shabat et al. (2019) present Nesti-Net, an approach that uses a mixture-of-experts method based on 3D convolutions to estimate the normal vector for individual points.

Ben-Shabat and Gould (2020) also use PointNet as their architecture but instead predict the point-wise weights for weighted least squares polynomial surface fitting.

In the following, we aim to show that, unlike previous approaches using point-wise queries, the normal vector can also be estimated in a single step. Analogous to semantic segmentation, we predict a result for each point simultaneously, estimating either just the orientation of a normal vector or the full vector directly. Since our target domain is airborne LiDAR scans, the point clouds are already correctly oriented and contain low noise. However, as these are surface scans, they inherently contain holes and occlusions and also cannot be processed into watertight meshes.

3. Methodology

In the following, we present our two different approaches. In the first approach, we focus solely on determining the orientation of the normal vector, while in the second approach, we estimate the complete surface normal vector. Both use the same datasets as training data with different data preparation strategies. Both use the same Point Transformer V3 encoder architecture presented by Wu et al. (2024) and only differ in the decoder head.

3.1 Normal orientation estimation

For many applications, particularly those involving planar surfaces, the normal vector itself is well-defined, but its orientation is difficult to estimate consistently. Our first model variant, in the following referred to as *Ours (orient.)*, addresses this issue by assuming a valid normal vector and simply estimating whether it needs to be flipped to the correct orientation. To do this, we treat the task as a two-class problem involving a cross-entropy loss function and flip the normal vectors during post-processing according to the output of our model. The input for our model consists of the XYZ coordinates, RGB values, and the preexisting normal vector. During training, we randomly flip 50 % of the normal vectors and use this as the binary label for the training pipeline. In this way, our model processes a meaningful surface vector but must learn its correct orientation. During testing, we use the same normal estimation as the reference approaches but fully randomize the orientations. We then flip the normals according to the output of our network.

3.2 Full normal estimation

For our second model *Ours (full)*, we aim to predict the entire normal vector by directly estimating its XYZ values. The advantage here is that the normal vector does not need to be known in advance, so only point coordinates and RGB color values are used as input. The output vector is first normalized and then compared to the ground truth normal vector. For this, we use the root angular loss as the loss function $L(\theta) = \sqrt{\theta}$ with $\theta = \arccos(n_{est} \cdot n_{gt})$ being the angle between the estimated normal vector n_{est} and ground truth normal vectors n_{gt} , taking the orientation into account. This is equivalent to the more general loss function $L_p(\theta) = |\theta|^p$, $0 < p < 1$ using a p of 0.5. In our experiments, this loss term converges better than alternatives such as the cosine loss or the L_1 angle loss. We find this particularly valuable in the final iterations, when small angular differences of just a few degrees have to be adjusted, as the corresponding loss curve is significantly steeper in this value range. The value ranges of different loss functions for angle estimation are visualized in Figure 1.

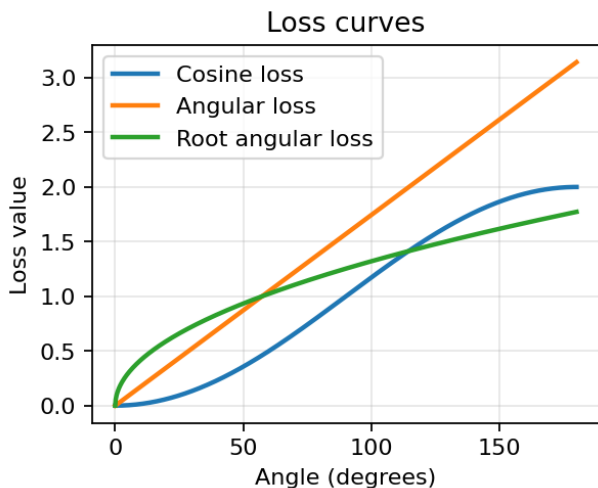


Figure 1. Value ranges of commonly used loss functions to estimate angles.

4. Evaluation

In this section, we present the results of our two methods and compare them with the reference approaches. We first describe the datasets, implementation details and the used reference approaches. Subsequently, we explain our evaluation procedure, since a conventional assessment based on ground-truth normal vectors is not feasible due to the absence of suitable datasets. To nevertheless assess the advantages of our approach, we use Poisson meshing as an indirect quality measure (Kazhdan et al., 2006).

4.1 Datasets

To the best of our knowledge, no aerial LiDAR dataset directly provides normal vectors. Therefore, we build our training dataset from a combination of photogrammetric 3D models with normals and processed LiDAR datasets. For the LiDAR data, we estimate normals via local plane fitting and orient them towards the sensor position at acquisition time. Additionally, we use datasets with textured meshes, from which we sample point clouds on the surface together with their corresponding normal vectors. Overall, our training data consists primarily of three publicly available benchmark datasets. We use photogrammetric 3D models reconstructed from the UseGeo dataset (Nex et al., 2024; Hermann et al., 2024), as well as the processed LiDAR scans for which we have carefully estimated and oriented the normal vectors. For this purpose, we split the dataset into training and test sets, using an 80/20 split by area. We also use point clouds sampled from the textured meshes of the training sets SUM and H3D datasets (Gao et al., 2021; Kölle et al., 2021).

Although the training data is obtained from multiple sources, the evaluation is carried out exclusively on LiDAR data. For this, we primarily use our test set from UseGeo and the official test set of H3D. In addition, we include a holdout dataset that is not part of the training pipeline to evaluate generalization to entirely unseen scenes. The JB3D dataset used for this purpose is similar to UseGeo and H3D, consisting of LiDAR scans of outdoor environments. For H3D, we split the LiDAR scans into 50 m × 50 m tiles, analogous to the mesh tiles used during training. The UseGeo and JB3D datasets are cut into 100 m × 100 m tiles. In total, this results in 38 individual point

clouds for testing. While the UseGeo dataset is predominantly urban, with many closely spaced buildings and patches of vegetation, the H3D dataset contains more isolated buildings and vegetation. In contrast, the JB3D dataset features a greater proportion of tall and low vegetation, together with urban geometry, like buildings, paths, and roads.

4.2 Implementation details

We train both models with a grid size of 5 cm and samples of up to 100,000 points for 1,000 epochs, using a batch size of 8. For data augmentation, we apply random rotation around the Z-axis and small rotations of up to 5 degrees around the X and Y axes, as well as random scaling, random shifting, and random point and color dropout. At test time, we use a combination of random rotation around the Z-axis and random scaling for augmentation.

4.3 Reference approaches

As a reference method, we first use a simple heuristic that aligns all normal vectors upwards and is referred to as *Towards Z+*. This implicitly assumes that we are only dealing with surface scans without any interior spaces. In addition, we employ Minimum-Spanning-Trees, as implemented in the open source software CloudCompare (CloudCompare, 2026), as a second group of baseline heuristics. We distinguish three variants: *MST-6*, *MST-10*, and *MST-20 combined*, which differ mainly in the number of neighbors connected to each node in the Minimum-Spanning-Tree. In general, increasing this number improves quality but also raises computation time. We use a radius of 0.2 m for normal estimation, since a larger radius leads to excessive smoothing of discontinuities. *MST-20 combined* further enhances quality by first estimating and orienting the normals with a radius of 3 m, yielding very smoothed edges but consistently oriented normals. This orientation is then fixed, and the normals are re-estimated with a radius of 0.2 m. In our experiments, this produces better results, but at a substantially higher computational cost.

4.4 Evaluation routine

While some LiDAR datasets may offer normals derived from photogrammetric reconstruction, these are inherently imperfect and may introduce systematic biases. Methods trained on similar photogrammetric data may appear artificially superior in direct comparisons, while traditional approaches like Minimum-Spanning-Trees, which lack domain knowledge about global orientation, can be unfairly penalized even when producing locally consistent results. We therefore report direct angular error on the JB3D dataset only as a supplementary analysis and instead adopt an indirect evaluation strategy based on mesh reconstruction quality.

Our primary evaluation uses Poisson surface reconstruction (Kazhdan et al., 2006), which is highly sensitive to normal quality. Inconsistent orientations between neighboring regions produce visible artifacts at boundaries, and inaccurate normal directions lead to geometric distortions, which are then both reflected in the reconstruction metrics. We perform Poisson meshing with an octree depth of 12 and remove regions below the 3 % density quantile to eliminate boundary artifacts. We then sample a point cloud from the reconstructed mesh surface and compare it against the original LiDAR scan.

As error metrics, we use the absolute error L1-abs and root-mean-square error (RMSE), which quantify the distance

Dataset	Method	Cpl. $\uparrow$	L1-abs $\downarrow$	RMSE $\downarrow$	CD $\downarrow$	% F1-Score $\uparrow$
H3D	Towards Z+	0.9190	0.0317	0.0395	0.0584	73.57
	MST-6	0.9246	0.0292	0.0357	0.0541	75.47
	MST-10	<u>0.9244</u>	0.0289	0.0353	0.0538	75.72
	MST-20 combined	0.9223	<u>0.0265</u>	<u>0.0319</u>	<u>0.0514</u>	<u>76.99</u>
	Ours (orient.)	0.9242	0.0280	0.0341	0.0525	76.23
	Ours (full)	0.9197	0.0259	0.0313	0.0510	77.91
UseGeo	Towards Z+	0.5997	0.0467	0.0531	0.0993	43.87
	MST-6	0.6014	0.0471	0.0538	0.0982	44.36
	MST-10	0.6022	0.0466	0.0530	0.0977	44.47
	MST-20 combined	0.6044	<u>0.0447</u>	<u>0.0504</u>	0.0966	<u>44.60</u>
	Ours (orient.)	<u>0.6035</u>	0.0464	0.0526	<u>0.0968</u>	43.85
	Ours (full)	0.5939	0.0442	0.0498	0.0991	45.00
JB3D	Towards Z+	0.7217	0.0520	0.0618	0.0971	41.92
	MST-6	0.7252	0.0522	0.0620	0.0956	41.84
	MST-10	<u>0.7251</u>	0.0508	0.0604	0.0943	42.74
	MST-20 combined	0.7235	<u>0.0494</u>	<u>0.0585</u>	<u>0.0955</u>	<u>43.21</u>
	Ours (orient.)	0.7207	0.0544	0.0655	0.1005	41.14
	Ours (full)	0.7055	0.0466	0.0552	0.0956	45.40

Table 1. Quantitative results comparing the reference methods with our approach for estimating only the orientation *Ours (orient.)* and our second approach *Ours (full.)* to predict the full normal vector. The results are ordered by dataset, with the second-best result per column and dataset highlighted underlined and the best result additionally by bold print.

between the sampled points and the ground truth LiDAR scan, as well as the completeness, which measures the extent to which the ground truth is covered. Following Seitz et al. (2006), we exclude outliers from the L1-abs and RMSE calculations, but apply a stricter threshold of 1 %, as the LiDAR scans contain very few outliers.

In addition, we calculate the Chamfer distance defined as the symmetric distance between two point clouds and the F1 score, which also jointly expresses the accuracy and completeness in a single metric. Following Knapitsch et al. (2017), we compute the precision by determining, for each point in the estimated point cloud, whether its nearest neighbor in the ground truth point cloud lies within 0.03 m. The fraction of such points, multiplied by 100, yields the precision in percent. Recall is computed analogously but starting from the ground truth. The completeness metric is equivalent to recall but is typically reported as a percentage rather than a score between 0 and 100.

4.5 Experimental results

As a supplementary analysis, we first evaluate the angles directly on the JB3D holdout dataset, which provides photogrammetrically derived normal vectors. To account for consistent but oppositely oriented results from Minimum-Spanning-Trees methods, we compute the angular error using once the estimated normal and its opposite orientation, taking the minimum. Our full normal estimation approach achieves a median angular error of 10.77 degrees, compared to 13.31 degrees for the next-best method *MST-20 combined*. When considering only unsigned angles, thus disregarding orientation, the gap narrows to 9.49 degrees versus 10.46 degrees. However, as noted above, these results should be interpreted cautiously due to the bias towards photogrammetric training data.

In the following, we present the quantitative and qualitative results based on Poisson surface reconstruction of our two methods and compare them with the reference approaches. All res-

ults are based on the analysis of the reconstructed triangular meshes. Regarding our network to estimate only the orientation we achieve up to 99% accuracy on our test set in terms of correcting the normal orientation. However, this means that large scenes can still contain thousands of misoriented normals. These errors are typically sparse and can be removed with a simple post-processing step like a median filter. In our case, the inherent smoothing in using Poisson meshing to reconstruct the triangle meshes does not necessarily require this. Table 1 shows our quantitative results across all three datasets. Figures 2 and 3 show qualitative results of the normal estimation and the following surface reconstruction. The first row shows the input point cloud. This is followed by two reference methods *MST-10* and *Towards Z+* as well as our two approaches to only estimate the orientation *Ours (orient.)* and our model, which estimates the full normal vector *Ours (full)*. For each approach, the first row visualizes the normal angle in HSV color space. It encodes the inclination and azimuth angles of each point’s normal vector as HSV colors, so that surfaces with similar angular orientation appear in similar colors. The second row depicts the mesh reconstruction results, with the error highlighted from values close to zero in blue to 0.03 m and higher in red.

The qualitative results suggest that Minimum-Spanning-Trees tend to struggle with estimating a globally consistent normal orientation on our datasets. This is visible at the chimney in the left column of Figure 2 and on the church tower roof in the middle column of Figure 3. In both cases, the geometric mesh quality suffers, because artifacts appear at the boundaries between oppositely oriented regions, and the meshes cannot be connected seamlessly. In vegetation, these locally varying orientations cause especially large errors, which are clearly noticeable in the JB3D dataset in the right column of both figures. The *Towards Z+* heuristic achieves a globally consistent orientation but fails on vertical structures such as facades especially visible in the first column of Figure 2 and the second column of Figure 3. By contrast, our methods appear to predict the

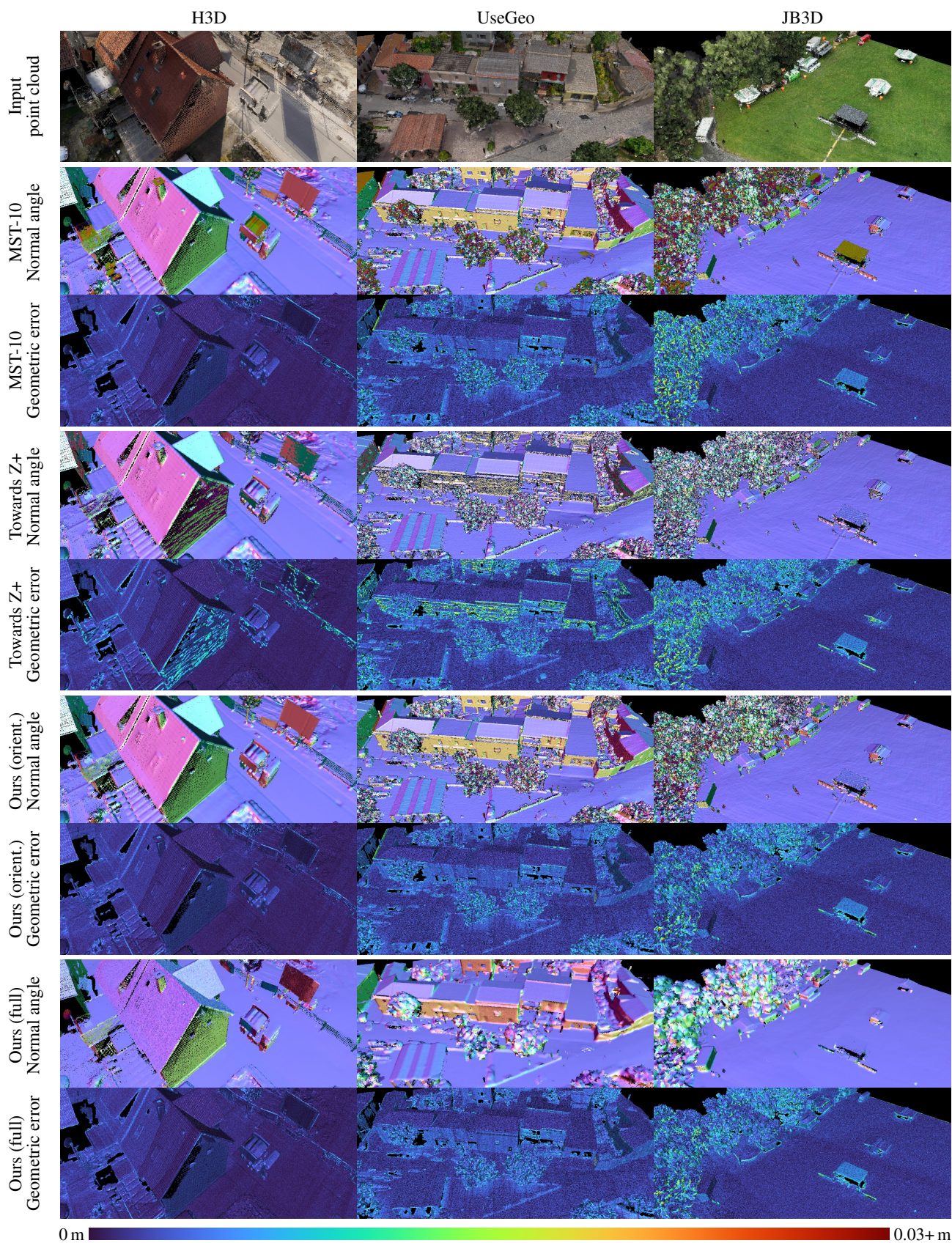


Figure 2. Qualitative comparison of two reference methods *MST-10* and *Towards Z+*, followed by our approach that only estimates the orientation *Ours (orient.)* and then our method for full normal estimation *Ours (full)*. In the first row, each method's normal angle is visualized using the HSV color space. The second row shows the geometric error of the mesh reconstructed from these normals, with errors ranging from near zero in blue towards 0.03 m and above in red.

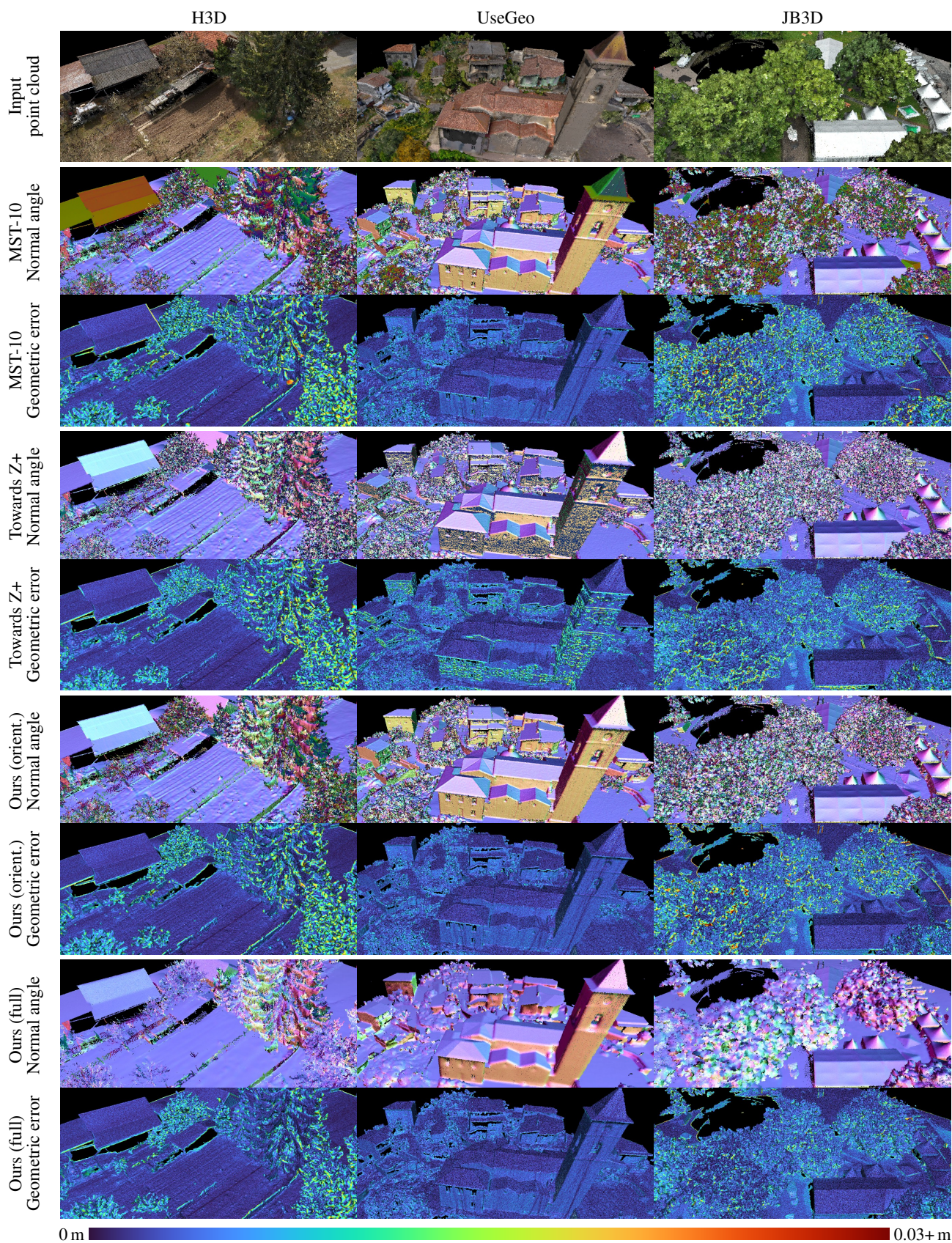


Figure 3. Qualitative comparison of two reference methods *MST-10* and *Towards Z+*, followed by our approach that only estimates the orientation *Ours (orient.)* and then our method for full normal estimation *Ours (full)*. In the first row, each method's normal angle is visualized using the HSV color space. The second row shows the geometric error of the mesh reconstructed from these normals, with errors ranging from near zero in blue towards 0.03 m and above in red.

orientation consistent both globally and locally. However, the model for full normal estimation tends to inherit strong biases from the training data and over-smooths discontinuities such as edges or corners, as well as vegetation. This is particularly apparent in the left column of Figure 3 when compared to the *MST-10* result, since the terrain looks very smooth. This effect is also visible when comparing these results with the output of our approach for only orientation estimation. Because the underlying normal vectors are estimated using a radius of 0.2 m, the edges appear significantly sharper.

The inaccurate orientation estimation using Minimum-Spanning-Trees seems to have only a minor effect on the quantitative results in Table 1, probably because the errors primarily occur at the boundaries between inconsistently oriented regions. Although substantial artifacts appear, they are typically low-density and are removed using the density threshold. However, a major drawback for downstream applications might be that the mesh is not connected at these locations. In general increasing the nodes in the Minimum-Spanning-Trees improves the quantitative results. As shown in Table 1, our approach for estimating the full normal vector outperforms the best reference method, *MST-20 combined*, by up to 2 percentage points in F1-Score while being significantly faster. However, our approach, which estimates only the orientation, results in slightly worse results, which may be due to suboptimal pre-computed normals.

The runtime of our two approaches depends heavily on the amount of test time augmentation. For a segment with approximately 3,000,000 points, our approach takes 4 min on an NVIDIA L40 GPU with test time augmentation and 0.25 min without. Without test time augmentation, the results drop on average by 0.2 percentage points in terms of F1-Score, which suggests that the model benefits little and already produces robust results without test time augmentation. The reference approach *MST-10* takes 2 min, and the multi-stage approach *MST-20 combined* requires a total of 25 min. Of these, 15 min are spent on the normal estimation with a 3 m radius, and 7 min on the orientation using *MST-20*.

5. Discussion

For the model that estimates only the orientation, our experiments have shown that at test time the initial orientation of the normal vectors in the input data is very important. If larger areas such as entire buildings are consistently mis-oriented, our approach cannot correct this. This can be easily resolved by randomizing the orientation of all normal vectors in a pre-processing step, which prevents such consistently incorrect regions. Our model is still able to predict the correct orientation afterwards. Furthermore, the results improve slightly when multiple consecutive runs are performed. Hereby, the input starts with randomized normal orientations and is then further refined as the model corrects some of its errors made in the first run. In this case, the input of the second inference step is the output of the first inference step further denoising the results. We also tested various types of data augmentation designed to simulate the different kinds of miss-oriented normal vectors that occur in practice, ranging from varying degrees of noise to systematic defects affecting entire regions. However, these did little to improve the model's robustness, and it turned out that simply eliminating any systematic effects by completely randomizing the normal vectors is already highly effective.

While predicting only the orientation is valuable when high-quality normals with undefined orientation are available, our experiments indicate that estimating the full normal vector is also promising. This not only removes the need for such a pre-processing step, but our quantitative evaluation in Table 1 also demonstrates that it yields better Poisson meshing quality. However, the training has proven susceptible to getting stuck in a local minimum once not only the unsigned normal vector but also its orientation has to be estimated. It is reasonable to assume that data preparation and model parameters play a crucial role here, since a local context that is too small, where for example only parts of a facade are visible, can produce an incorrect training signal due to the undefined normal orientation.

As shown in Figure 2 and Figure 3, our method is capable of producing consistent results. Nonetheless, the qualitative examples also clearly show that the deep learning model is constrained by the quality of the training data. In particular, the smooth edges of the mesh datasets and the photogrammetric point clouds are reflected in the predictions. This effect is especially noticeable in vegetation, such as the trees in the JB3D dataset. At the same time, these observations confirm that the model can both accurately reproduce properties of the training data and apply this knowledge to unseen scenes. Consequently, if training data with higher-quality normals were available, the model would likely achieve significantly better performance. The domain gap between photogrammetric and mesh-sampled training data on the one hand, and the LiDAR scans used for evaluation on the other, may also be limiting. Exploring synthetic datasets could be a promising direction for future work.

Another advantage of our method is that it requires no parameter tuning, since neither a radius for normal estimation nor a Minimum-Spanning-Tree size must be chosen. The only required domain knowledge is that the 3D model must be aligned with the Z+ axis, consistent with the training data.

6. Conclusion

In summary, we introduce two methods for estimating surface normals in unordered point clouds from airborne LiDAR scans. The first predicts only the orientation, while the second directly estimates the full normal vector. Across multiple LiDAR datasets, both variants consistently reduce artifacts and improve the quality of reconstructed triangle meshes compared to the reference methods.

Our approaches not only produce consistently oriented normals but also deliver competitive Poisson meshing results. Our approach for full normal estimation outperforms the strongest baseline *MST-20 combined* by up to 2 percentage points in F1-Score while being significantly faster. A key advantage is that our normals are consistent both locally and globally, whereas the reference primarily produces local consistency leading to artifacts at boundaries between incorrectly oriented regions.

A further benefit is the relative ease of gathering training data, since several public datasets either provide images for photogrammetric reconstructions or directly contain textured meshes from which normals can be sampled. However, this setup introduces a clear bias toward smoother normals. Looking ahead, improving the training data appears particularly promising, since the model can accurately reproduce dataset characteristics but tends to oversmooth discontinuities. It is therefore reasonable to expect that more suitable training data would yield substantially better results.

References

- Amenta, N., Bern, M., 1998. Surface reconstruction by voronoi filtering. *Proc. 14th Annual Symposium on Computational Geometry*, 39–48.
- Ben-Shabat, Y., Gould, S., 2020. Deepfit: 3d surface fitting via neural network weighted least squares. *Proc. European Conference on Computer Vision*, Springer, 20–34.
- Ben-Shabat, Y., Lindenbaum, M., Fischer, A., 2019. Nestinet: Normal estimation for unstructured 3d point clouds using convolutional neural networks. *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10112–10120.
- CloudCompare, 2026. CloudCompare (version 2.13.2): 3d point cloud and mesh processing software. <https://www.cloudcompare.org/>. Open-source software, accessed 2026-03-24.
- Dey, T. K., Goswami, S., 2004. Provable surface reconstruction from noisy samples. *Proc. 20th Annual Symposium on Computational Geometry*, 330–339.
- Dey, T. K., Li, G., Sun, J., 2005. Normal estimation for point clouds: A comparison study for a voronoi based method. *Proc. Eurographics/IEEE VGTC Symposium Point-Based Graphics*, 39–46.
- Gao, W., Nan, L., Boom, B., Ledoux, H., 2021. SUM: A Benchmark Dataset of Semantic Urban Meshes. *ISPRS Journal of Photogrammetry and Remote Sensing*, 179, 108–120.
- Guerrero, P., Kleiman, Y., Ovsjanikov, M., Mitra, N. J., 2018. PCPNet: Learning Local Shape Properties from Raw Point Clouds. *Computer Graphics Forum*, 37(2), 75–85.
- Hermann, M., Weinmann, M., Nex, F., Stathopoulou, E., Remondino, F., Jutzi, B., Ruf, B., 2024. Depth estimation and 3D reconstruction from UAV-borne imagery: Evaluation on the UseGeo dataset. *ISPRS Open Journal of Photogrammetry and Remote Sensing*, 100065.
- Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W., 1992. Surface reconstruction from unorganized points. *Proc. 19th Annual Conference on Computer Graphics and Interactive Techniques*, 71–78.
- Kazhdan, M., Bolitho, M., Hoppe, H., 2006. Poisson surface reconstruction. *Proc. Eurographics Symposium on Geometry Processing*, 61–70.
- Knapitsch, A., Park, J., Zhou, Q.-Y., Koltun, V., 2017. Tanks and Temples: Benchmarking Large-Scale Scene Reconstruction. *ACM Transactions on Graphics*, 36(4).
- König, S., Gumhold, S., 2009. Consistent propagation of normal orientations in point clouds. *International Symposium on Vision, Modeling, and Visualization*, 83, 92.
- Kölle, M., Laupheimer, D., Schmohl, S., Haala, N., Rottensteiner, F., Wegner, J. D., Ledoux, H., 2021. The Hessigheim 3D (H3D) benchmark on semantic segmentation of high-resolution 3D point clouds and textured meshes from UAV LiDAR and Multi-View-Stereo. *ISPRS Open Journal of Photogrammetry and Remote Sensing*, 1, 11.
- Nex, F., Stathopoulou, E., Remondino, F., Yang, M., Madhuanand, L., Yogender, Y., Alsadik, B., Weinmann, M., Jutzi, B., Qin, R., 2024. UseGeo – A UAV-based multi-sensor dataset for geospatial research. *ISPRS Open Journal of Photogrammetry and Remote Sensing*, 13, 100070.
- Qi, C. R., Su, H., Mo, K., Guibas, L. J., 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 652–660.
- Qi, C. R., Yi, L., Su, H., Guibas, L. J., 2017b. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, 30.
- Seitz, S., Curless, B., Diebel, J., Scharstein, D., Szeliski, R., 2006. A comparison and evaluation of multi-view stereo reconstruction algorithms. *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 519–528.
- Thomas, H., Qi, C. R., Deschaud, J.-E., Marcotegui, B., Goulette, F., Guibas, L. J., 2019. Kpconv: Flexible and deformable convolution for point clouds. *Proc. International Conference on Computer Vision*, 6411–6420.
- Wu, X., Jiang, L., Wang, P.-S., Liu, Z., Liu, X., Qiao, Y., Ouyang, W., He, T., Zhao, H., 2024. Point Transformer V3: Simpler, Faster, Stronger. *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4840–4851.
- Wu, X., Lao, Y., Jiang, L., Liu, X., Zhao, H., 2022. Point transformer v2: Grouped vector attention and partition-based pooling. *Advances in Neural Information Processing Systems*, 35, 33330–33342.
- Xie, H., Wang, J., Hua, J., Qin, H., Kaufman, A., 2003. Piecewise c1 continuous surface reconstruction of noisy point clouds via local implicit quadric regression. *Proc. 14th IEEE Visualization*, VIS '03, 13.
- Zhao, H., Jiang, L., Jia, J., Torr, P. H., Koltun, V., 2021. Point transformer. *Proc. International Conference on Computer Vision*, 16259–16268.