SURFACE MODELING FOR AIRBORNE LIDAR

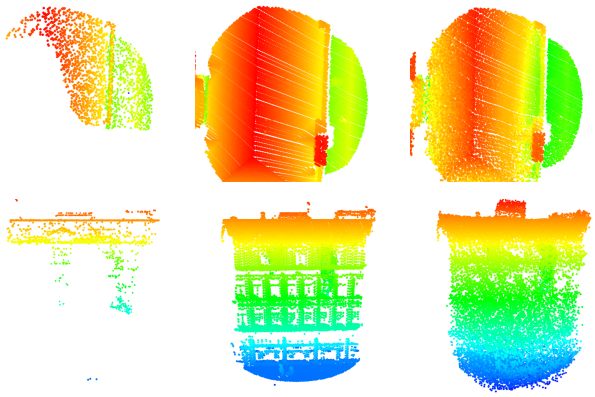
Hunter Blanton, Sean Grate, Nathan Jacobs

University of Kentucky

ABSTRACT

Repeat-visit airborne lidar is a powerful tool for change detection in urban and rural environments. In this work, we present a learning-based approach that addresses one of the key challenges in comparing point cloud scans of the same region: handling geometric differences caused by varying sensor position. Our approach is to perform shape modeling through ray casting with a point cloud neural network. Recent work on learning-based shape modeling has been based on the assumption that an explicit surface representation is available, which is not the case for airborne lidar datasets. Our key insight is that by using a ray casting approach we can perform shape modeling directly with lidar measurements. We evaluate our method both quantitatively and qualitatively on learned surface accuracy and show that our method correctly predicts surface intersection even in sparse regions of the input cloud.

***Index Terms*—** lidar, machine learning, ray casting, point cloud synthesis



(a) Input (b) Ground Truth (c) Prediction

Fig. 1: Qualitative results of our method for two separate inputs. Color represents the height. Our method produces accurate results even when sampling in sparse regions.

1. INTRODUCTION

Airborne lidar allows for extremely accurate, but sparse, measurements of surface geometry. The specific points captured by the lidar depend on the underlying surface as well as the position of the sensor and the angle of the lidar beam. Some points are only visible from particular viewing rays due to occlusion by other objects, leading to missing points. To gather more detailed measurement of a region, multiple scans from different flight routes are combined to increase point sampling density. However, due to changes in the environment over time, there will be regions of the merged point cloud that are not geometrically consistent, such as measurements of a building at different stages of construction. A detailed surface model from a single scan would aid in determining these inconsistencies and allow for better fusion of scans over time.

Neural networks have had great success in surface modeling in recent years. Given sufficient training data, they have been able to accurately and efficiently represent even complex shapes. Recent approaches have focused on networks that make predictions for each point in space [1, 2, 3]. Given a point, these models explicitly classify the point as being inside or outside the surface, or regress a Signed Distance Function (SDF) value for the point. While they allow for flexibility in shape complexity and resolution, these methods require known, detailed surface information at training time, limiting them mostly to synthetic data with known geometry.

In this work, we propose an alternate approach to surface modeling by predicting surface intersections. Our method, based on ray casting, offers many benefits over the point sampling approaches mentioned above. Unlike point sampling which may require several iterations to find a point on the surface, ray casting produces a precise surface estimate in a single step, simplifying inference significantly. The major benefit, however, lies in the training data requirements. For typical methods, a complete mesh is required for generating training samples. Also, due to the point-based nature, care must be taken to sample a sufficient number of points near the object boundary. Because we instead rely on ray-surface intersections, the data necessary to train our system is provided directly from lidar sensors. Despite the relaxed data requirement, our method still allows for predicting point probabilities over all points in space. Our contributions include a novel approach for surface representation by ray casting and qualitative and quantitative evaluation of our approach applied to real-world airborne lidar.

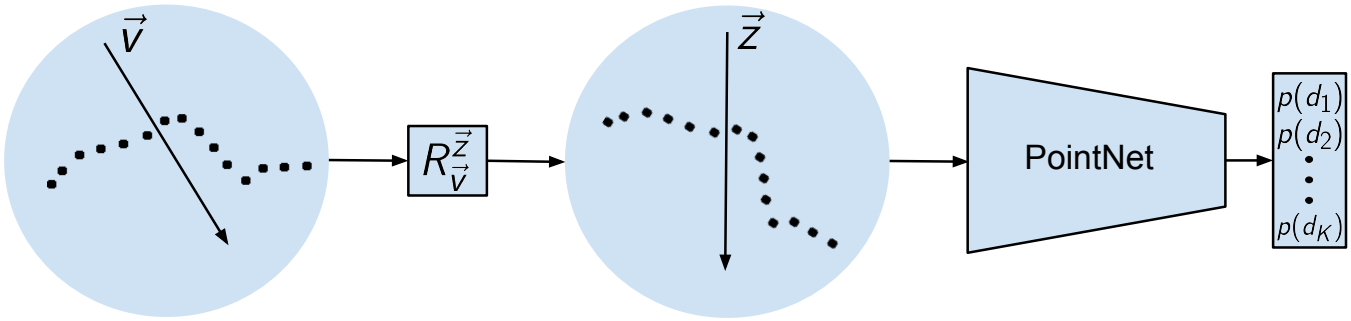


Fig. 2: Our architecture. The point cloud is first rotated into a fixed orientation and then passed through a PointNet that outputs a probability distribution over K discretized surface intersection distances.

2. RELATED WORK

In recent years, much progress has been made in understanding point clouds using learning-based methods. PointNet [4] pioneered using point clouds as direct input to neural networks. PointNet++ [5] extends PointNet using multi-scale grouping to capture both local and global features extracted from the raw input. Similar networks have become increasingly popular in remote sensing tasks, such as segmentation of large-scale lidar point clouds [6, 7].Many methods have been developed for generating new point clouds [8, 9] or filling in missing data [10, 11]. These methods can generate accurate and spatially consistent point clouds, but do not allow for fine-grained control over the location of the new points. Our approach generates new point samples along a specific view ray, providing a level of control that is not obtainable with other methods.

Neural networks have achieved remarkable success for implicit surface modeling. Early approaches typically focused on translating from one representation to another, e.g. image to point cloud [12], multi-image to occupancy grid [13], point cloud to mesh [14], etc. The current state-of-the-art learning-based approaches involve predicting information about a point's relation to the surface boundary for each point in space. Chen and Zhang [2] classify points as inside or outside of the surface, effectively learning an implicit field of the shape. Mescheder et al. [3] propose a similar idea and show that the shape embedding can be predicted from RGB image input. DeepSDF [1] instead opts to regress the SDF value for each point. This allows their model to predict both the inside/outside prediction as before, but also the minimum distance from a point to the surface boundary. The distance information is used to simplify finding the surface boundary.

The issue with the above learning-based methods is the requirement for known surface geometry to construct training data. To know if a point is inside or outside of the surface, or how far away from the surface it lies, you must have precise knowledge of the surface itself. This is typically only possible with synthetic meshes. On the other hand, our approach relies solely on direct sensor measurements, allowing it to be applied to real-world data such as airborne lidar.

Williams et al. [15] optimize an ensemble of overlapping surface patch maps for accurate and consistent surface regression. This approach is different from most others in that it requires no prior surface knowledge other than the input point cloud and the individual patch models are optimized only for the current input. This means that the method can be applied to arbitrary point clouds with no need for pretraining. However, this has only been shown for dense point clouds of single objects, and must be optimized separately for each point cloud. Our approach on the other hand can be applied to unseen airborne lidar measurements.

3. LIDAR RESAMPLING BY RAY CASTING

We address the problem of predicting likely points from novel scan positions in a previously captured lidar point cloud. Given a point cloud, P, of an underlying surface, S, and sampling ray, $\vec{v}$, we want to determine the intersection of $\vec{v}$ and S. That is to say, if a lidar sensor were to measure along $\vec{v}$, we wish to find at what distance the sensor would detect a point. To do this, we propose a neural network based classification system that predicts a probability distribution over surface intersection distances along the ray. Since our method only depends on an input point cloud and a view ray, our method is capable of synthesizing entirely new lidar scans from different scan positions and orientations.

3.1. Architecture

An overview of our ray casting approach is shown in Figure 2. Given a point cloud and a view ray $\vec{v}$, we want to predict where along $\vec{v}$ the surface is likely to intersect. A transformation $R^{\vec{z}}_{\vec{v}}$ is first applied to the point cloud such that $\vec{v}$ aligns with the z-axis, $\vec{z}$, ensuring that sampling is performed with a fixed orientation. The aligned point cloud is then fed into a standard PointNet [4]. The PointNet operates by extracting features for each point then applying average pooling to get a single feature vector to feed into a multi-layer perceptron. The final output is a probability distribution over possi-

Table 1: Surface resampling with varying input cloud sizes for training and inference. Results are given as (mean/median) pairs for MSE (in meters).

Train Size	Prediction	Inference Size			
		512	1024	2048	4096
-	baseline	44.208/18.990	43.890/18.739	43.638/18.541	43.438/18.387
512	argmax	1.128/0.662	1.123/0.657	1.117/0.655	1.293/0.640
	mean	1.115/0.653	1.109/0.647	1.105/0.646	1.276/0.621
1024	argmax	1.015/0.631	1.008/0.628	1.003/0.624	1.002/0.623
	mean	0.984/0.614	0.977/0.611	0.973/0.606	0.971/0.606
2048	argmax	0.931/0.590	0.924/0.586	0.921/0.583	**0.919**/0.583
	mean	0.933/0.578	0.925/0.572	0.922/0.569	0.920/**0.569**
4096	argmax	0.986/0.646	0.980/0.643	0.977/0.641	0.975/0.640
	mean	0.948/0.613	0.941/0.610	0.939/0.609	0.936/0.608

ble surface intersection distances along the ray. We use the cross-entropy loss between the network output and ground truth label to optimize the network. A single point location can be produced from this distribution in a number of ways, for example taking the highest probability distance.

3.2. Dataset

Training requires points with associated view ray information, which is immediately available in lidar waveform data. We use a large dataset of airborne lidar scans to construct a dataset for training and evaluation. The NYU Dublin dataset [16] is an extremely high resolution point cloud containing 41 overlapping flight paths over a 2 km^2 area of Dublin, Ireland. The large number of overlapping scans results in a point density of roughly 300 points per m^2. All scans were collected in March 2015 so there is little variation in landscape and structures.

Though the sensor location is unknown, each point in the dataset has an associated location,(x, y, z), and scan direction, (v_x, v_y, v_z). We use this information to generate a large number of point cloud regions. Each region contains thousands of points within a 10 meter radius cylinder centered around a random point inside the entire Dublin point cloud. We construct virtual sensors using the original point's scan directions to give each ray a starting position. We generate 5000 regions for training and 100 for testing. The testing regions are pulled from a held out area of the Dublin point cloud that was not used for training.

3.3. Implementation Details

For any location in the dataset, there are potentially several scans of the area. We only consider regions that contain points from multiple scans such that we can test the ability of the network to synthesize unseen points. To generate a single training example, we choose one scan to be the input source scan and pick a target sample point from the remaining scans. Once the input scan has been selected, we randomly sample N points to be passed into the network. The output label is computed by finding the location of the target point along the view ray and placing the distance into one of K bins. The bins represent discretized distances between the minimum and maximum distance values. In our case, the min and max distances are 0 and 20 meters, respectively.

We implemented our method using the PyTorch [17] framework. We trained each model with the Adam [18] optimizer, a learning rate of $1e^{-4}$, a weight decay of $1e^{-3}$, and a batch size of 128 for 100 epochs. Points are scaled to fit within a unit sphere so that the input to the network is in the range $[-1, 1]$. We apply random rotations and shifts about the view ray as data augmentation during training.

4. EVALUATION

We evaluate our ray casting method for novel scan synthesis. For each test, we sample points along view rays that are not measured in the input scan. As we compute points along a specific view ray, we have direct correspondence between ground truth and predicted points. We report the mean and median Euclidean distance in meters between corresponding ground truth and predicted points. For computing metrics, the output of the network is compared to the true, real-valued distance, not the discretized value used for training.

4.1. Quantitative Analysis of Ray Casting

We evaluate our method on a wide variety of input point cloud sizes for both training and inference. For each test, the output probability distribution has $K = 256$ bins, which was found the be best for our experiments. Given the output probability distribution from the network, we can calculate distance in several ways. For our experiments, we compute both the distance with the highest probability as well as the weighted average over all distances.

Results are given in Table 1. We compare to a baseline method where new points are sampled based on the closest points in the input scan. This is performed by finding the

$C = 10$ nearest points and projecting their average onto the view ray. We tested with several different $C \in [0, 10]$ and found they performed similarly. While this methods performs well when there are nearby points, many test points lie in sparse regions of the input point cloud where there are few or zero points within a reasonable neighborhood, leading to sub-optimal performance. However, our approach works well in these cases, accurately filling in large gaps.

Interestingly, we show that the weighted average distance performs better than the maximum in general. This implies that the network is somewhat handling uncertainty in the prediction when the true point falls in between two bin centers. Overall, our best models are able to synthesize new points with sub-meter mean and median error. Note that at around 2048 input points performance is more-or-less stable. We believe that due to the average pooling in the network, 2048 points is a sufficient number of samples to get close to the true point cloud feature. The benefits of using at least this many points are seen both in training and inference, where performance significantly improves up to this size.

4.2. Qualitative Evaluation

As seen in Figure 1, our approach works surprisingly well in sparse regions where there is very little information provided by the input point cloud. We believe that this can be explained by the somewhat uniform nature of Dublin. The network is able to capture the structure of the buildings in the dataset and apply it to unseen data. While fine details are not able to be completely generated, the model is able to fill in missing data remarkably well. This provides evidence that a sophisticated surface representation is being learned even with the sparse surface measurements provided for training.

5. CONCLUSION

We presented a method for implicit surface modeling of airborne lidar through direct point cloud ray casting. Our method allows for accurate and simple to generate surface predictions. Because it depends only on easily obtainable samples for training, it can be applied to real-world data. When applied to real airborne lidar data, we show that our approach is able to accurately sample new points even in areas where the input cloud is extremely sparse. We believe our method is useful for lidar point cloud change detection, and hope to evaluate it for this task in future work.

Acknowledgements We gratefully acknowledge the support of the National Science Foundation (IIS-1553116).

6. REFERENCES

[1] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *CVPR*, 2019. 1, 2

[2] Zhiqin Chen and Hao Zhang, "Learning implicit fields for generative shape modeling," in *CVPR*, 2019. 1, 2

[3] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *CVPR*, 2019. 1, 2

[4] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *CVPR*, 2017. 2

[5] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *NIPS*, 2017. 2

[6] J. Contreras and J. Denzler, "Edge-convolution point net for semantic segmentation of large-scale point clouds," in *IGARSS*, 2019. 2

[7] Y. Lian, T. Feng, and J. Zhou, "A dense pointnet++ architecture for 3d point cloud semantic segmentation," in *IGARSS 2019*, 2019. 2

[8] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum, "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling," in *NIPS*, 2016. 2

[9] Heli Ben-Hamu, Haggai Maron, Itay Kezurer, Gal Avineri, and Yaron Lipman, "Multi-chart generative surface modeling," *ACM Transactions on Graphics*, vol. 37, no. 6, pp. 1–15, 2018. 2

[10] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner, "Shape completion using 3d-encoder-predictor cnns and shape synthesis," in *CVPR*, 2017. 2

[11] David Stutz and Andreas Geiger, "Learning 3d shape completion under weak supervision," *International Journal of Computer Vision*, pp. 1–20, 2018. 2

[12] Haoqiang Fan, Hao Su, and Leonidas J Guibas, "A point set generation network for 3d object reconstruction from a single image," in *CVPR*, 2017. 2

[13] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese, "3d-r2n2: A unified approach for single and multi-view 3d object reconstruction," in *ECCV*, 2016. 2

[14] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry, "AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation," in *CVPR*, 2018. 2

[15] Francis Williams, Teseo Schneider, Claudio Silva, Denis Zorin, Joan Bruna, and Daniele Panozzo, "Deep geometric prior for surface reconstruction," in *CVPR*, 2019. 2

[16] Debra F. Laefer, Saleh Abuwarda, Anh-Vu Vo, Linh Truong-Hong, and Hamid Gharibi, "2015 aerial laser and photogrammetry datasets for dublin, ireland's city center," 2017. 3

[17] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer, "Automatic differentiation in PyTorch," in *NIPS Autodiff Workshop*, 2017. 3

[18] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. 3