2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Fast and Robust Normal Estimation for Sparse LiDAR Scans

Igor Bogoslavskyi Konstantinos Zampogiannis Raymond Phan

***Abstract*— Light Detection and Ranging (LiDAR) technology has proven to be an important part of many robotics systems. Surface normals estimated from LiDAR data are commonly used for a variety of tasks in such systems. As most of the today's mechanical LiDAR sensors produce sparse data, estimating normals from a single scan in a robust manner poses difficulties.**

In this paper, we address the problem of estimating normals for sparse LiDAR data avoiding the typical issues of smoothing out the normals in high curvature areas.

Mechanical LiDARs rotate a set of rigidly mounted lasers. One firing of such a set of lasers produces an array of points where each point's neighbor is known due to the known firing pattern of the scanner. We use this knowledge to connect these points to their neighbors and label them using the angles of the lines connecting them. When estimating normals at these points, we only consider points with the same label as neighbors. This allows us to avoid estimating normals in high curvature areas.

We evaluate our approach on various data, both self-recorded and publicly available, acquired using various sparse LiDAR sensors. We show that using our method for normal estimation leads to normals that are more robust in areas with high curvature which leads to maps of higher quality. We also show that our method only incurs a constant factor runtime overhead with respect to a lightweight baseline normal estimation procedure and is therefore suited for operation in computationally demanding environments.

I. Introduction

LiDAR sensors have proven their versatility in a number of domains, ranging from self-driving cars [11], [21], [23], [29], through agriculture [10], [26], and even on space missions [24]. In all of these environments, LiDAR sensors are routinely used for perception, localization and mapping among other use cases. Even though our contributions might be useful in all of these scenarios, in this paper we focus on applying our normal estimation method to the latter ones: localization and mapping tasks using LiDAR data.

A key step in these tasks is 3D data alignment and a key choice for data alignment is the choice of features. There is a significant amount of research into which features are best for the task. Today's SLAM systems use corner and planar features [35], surfels [5], or even raw points directly [30]. However, we believe that a point-to-plane metric remains an important staple in point cloud data alignment. In our view, it allows for better data associations, which simplifies and improves the alignment process, warranting the cost of additional computations needed for normal feature extraction.

There is a number of methods to compute normals needed for the point-to-plane metric to function. One common method is to fit a plane to a neighborhood of each point [15] and use the normal of that plane as an estimate of the surface orientation in that point. This method, however, has a number of drawbacks: it works poorly on sparse data, where larger neighborhoods must be used, and, being of $\mathcal{O}(\log n)$ complexity, it takes a significant amount of computation to find the neighborhood of any given point.

All authors are with Magic Leap.

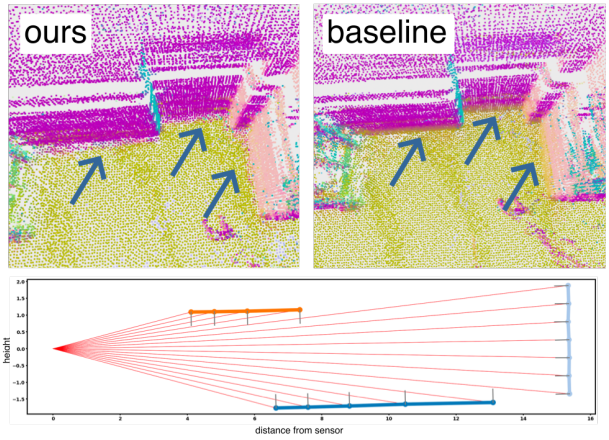

Fig. 1: Top two images show a part of an environment reconstructed by aligning multiple individual point clouds from the HILTI dataset [14]. Both point clouds are colored by the normals of their points. The right one uses a baseline implementation of normals and arrows indicate areas where the normals spill into an orthogonal surface. The left image shows that this does not happen when using the proposed method for normal estimation. The bottom image shows a single scan generated from a Velodyne VLP16 LiDAR, positioned at $(0, 0)$, with gray lines showing the normals estimated with the proposed method. Thick colored lines connecting the points indicate the connected components that the points belong to.

Therefore, to reduce the amount of computation needed, we follow the approach used by Behley and Stachniss [5], which we will call the *baseline* method for normal computation. They utilize the fact that the data produced by a LiDAR sensor is structured due to the LiDAR data acquisition pattern. Mechanical LiDARs spin an array of rigidly fixed lasers, producing data that can be viewed in an organized fashion. This allows to query the neighbors of any point in $\mathcal{O}(1)$ time which makes the normal computation very efficient. This method is faster than the full plane-fitting method but suffers from the fact that the data can be sparse, especially between the points stemming from different lasers, due to the physical laser diode placement in the body of a LiDAR sensor. Thus baseline normal estimation fails to produce satisfying results when the curvature of the underlying surface is high.

The main contribution of this paper is a method that improves upon the baseline normal computation technique by clustering the points stemming from neighboring lasers into

components likely describing the same underlying surface and computing the normals within the clusters of these points, avoiding cross-surface normal computation. We evaluate our method with respect to our main two claims: (i) that our method produces normals that are more robust than the ones estimated by the baseline method and (ii) that our method only incurs a constant factor runtime overhead when compared to the baseline method.

II. Related Work

The topic of normal estimation enjoyed an extensive coverage throughout the years. As of this writing, methods for estimating surface normals from point clouds can be decomposed into two predominant methods: *traditional* methods and *learning-based* methods.

The traditional methods use the geometry and/or extracting hand-crafted features from the point cloud directly to find the best fitting plane for a subsequent a least-squares minimization. Our method also lies within this category.

One of the earliest methods for estimating surface normals in point clouds was to use PCA [15] which derives each point's normal by computing the eigenvalues and eigenvectors of the covariance matrix with the k-nearest neighbors. The eigenvector corresponding to the smallest eigenvalue is determined to be the normal at that point. Stemming from this method, several other approaches have been proposed [3], [22], [33] which consider noise, curvature and sampling density of the point clouds. However, these methods tend to smooth out sharp edges and corners in potential shapes that are apparent in the point cloud which is what we are trying to avoid with our method.

To combat these problems, other approaches like Voronoi-based methods [1], [2], [12], minimizing L_0 or L_1 norms [3], [28], adopting statistical-based methods [18], [34] and making use of low-rank matrices [20], [36], [37] have been proposed. These methods have a better ability to preserve sharp features for point clouds in the presence of noise. Similarly, in the work that is closest to ours, Badino et al. [4] reformulate the estimation problem by taking advantage of the organized nature of the data available from a LiDAR scanner, and use range images to compute normals. The normals then are obtained by calculating derivatives from the surface generated from a spherical range image. All of the methods above, though, work with denser data, while we explicitly target to improve normals computed from a single revolution of a very sparse LiDAR.

One typical way to estimate normals in a computationally-constrained environment is to perform a cross product between the vectors towards the points neighboring the query point. Behley and Stachniss [5] use the structured nature of the data provided by a LiDAR sensor to get the neighbors of such a point in constant time and compute a normal using the cross product as described above. Our method builds directly on top of this idea improving it by labeling the points in correspondence to the underlying surface in the environment that they stem from.

In recent years, learning-based normal estimation methods have become more prevalent with the proliferation of deep learning. Only a few of these methods have the ability to preserve sharp features, robustly handle noise and generalize across different shapes. Most of these approaches still require a GPU to run fast enough to be used effectively. We do not aim to compete with these methods and provide them here for a general overview.

One of the first approaches was presented by Boulch and Marlet [9] where the HoughCNN architecture was introduced. In it 3D point clouds are mapped to a Hough space, and a CNN estimates normals in this representation. The transformation of the 3D points into a 2D Hough space is required as CNNs operate on 2D or image data. Because of this transformation, this method may discard vital geometrical details. Alternatively, in their paper, Qi et al. [25] introduce their seminal PointNet architecture that allows neural networks to directly consume 3D point clouds without transforming them into a 2D space. Using PointNet as a backbone, Guerrero et al. [13] introduce their PCP-Net for feature extraction to encode local neighborhoods of points to allow for direct regression of the normals. Similarly, Ben-Shabat et al. [6] propose a bagging approach that uses multiple networks in a mixture that observes the neighborhoods of each point at varying scales and selects the optimal network for predicting the normals. In [16], the authors propose a multi-scale k-nearest neighbors to robustly extract local features used as a pre-processing step to finally estimate the normals. Most of these and similar learning-based works [16], [17], [19], [31], [32] aim at finding the normals of dense and unstructured point clouds and due to the need of a GPU do not warrant a direct comparison to our method which aims to run fast on any CPU.

To the degree of our knowledge, our method is the first method to perform connected components on structured data coming from a sparse LiDAR sensor to aid the normal robustness while still maintaining the constant normal computation time per point.

III. Our Approach to Fast and Robust Normal Computation

A. Baseline Normal Computation

Our approach improves on the baseline normal estimation method as presented by Behley and Stachniss [5]. Likewise to their work, we focus on data generated by mechanical LiDARs. These LiDARs spin a physical array of lasers and provide the returns of these lasers as they are measured. These data can be viewed in an organized manner as illustrated in Fig. 2, where the shown points stem from lasers physically mounted on top of each other. Such data storage allows us to query the immediate neighbors of any given point in constant time. Consider that every point $\boldsymbol{p} = [x, y, z]^\top$ has up to four neighbors: $\boldsymbol{p}_{\text{left}}$, $\boldsymbol{p}_{\text{right}}$, $\boldsymbol{p}_{\text{top}}$, and $\boldsymbol{p}_{\text{bottom}}$. Given these neighbors we can compute the normal $\boldsymbol{n} = [n_x, n_y, n_z]^\top$ at point $\boldsymbol{p}$ as follows [5]:

$$\boldsymbol{n} = (\boldsymbol{p}_{\text{right}} - \boldsymbol{p}_{\text{left}}) \times (\boldsymbol{p}_{\text{top}} - \boldsymbol{p}_{\text{bottom}}), \tag{1}$$

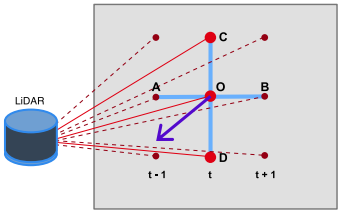

Fig. 2: Data that comes from a mechanical LiDAR is organized. Point O has 4 neighbors. Points A and B are generated by the same beam at a previous and next scan. Points C and D are generated by the beams neighboring the beam from which the point O stems within the same scan. Given these 4 neighbors a baseline way to compute a normal would be to compute a cross product between vectors $\overrightarrow{AB}$ and $\overrightarrow{CD}$ as shown in Eq. (1) that results in the normal shown in purple. This approach is noisy but serves as a good starting point for our approach.

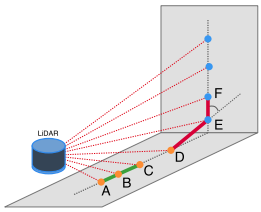

Fig. 3: Points generated from a single vertical stack of lasers. We interchangeably call such points a slice of LiDAR data or being "vertical neighbors". Points are colored orange and blue with respect to their label after clustering them with respect to the angles of the line segments connecting them. Some of these segments are shown too. The green segments AB and BC which form an angle $\angle ABC$ indicate that the points should belong to the same connected component, while the red lines forming the angle $\angle DEF$ indicate that the points D, E, and F should belong to different connected components.

here, $(\boldsymbol{p}_{\text{right}} - \boldsymbol{p}_{\text{left}})$ corresponds to $\overrightarrow{AB}$ in Fig. 2, while $(\boldsymbol{p}_{\text{top}} - \boldsymbol{p}_{\text{bottom}})$ corresponds to $\overrightarrow{CD}$, and the normal $\boldsymbol{n}$ being shown in purple. One thing to consider in this approach is that if one of either horizontal or vertical neighbors is missing, the point $\boldsymbol{p}$ can be used instead. Should both neighbors in one direction be missing, we consider such a point invalid and do not compute a normal for it.

This method has its advantages and disadvantages. Because the neighborhood of any single point is found in $\mathcal{O}(1)$ time, it is efficient and thus is commonly used when the computational resources are limited. At the same time, such small neighborhoods are a double-edged sword as this method is sensitive to the noise in the positions of the points. While white noise induced by the sensor noise does not usually pose big issues as the normal vectors can be averaged over multiple measurements, there is also a *systematic error caused by the sparsity of the LiDAR data*, especially in the data recorded by neighboring lasers which leads to points far apart being wrongly considered as neighbors.

The underlying assumption here is that the points lie close to each other and therefore stem from the same underlying surface patch. The distance between the vertical neighbors, however, grows quickly when the points are measured further away from the sensor. This reduces the likelihood that the neighboring points stem from the same underlying surface, breaking the assumption above. For an example of such a situation, see Fig. 3, where the points D and E lie on different planar patches of the environment while still being neighbors due to their acquisition order. Our main contribution is a heuristic-based method that deals with such situations better, increasing the chances of estimating normals within a single planar patch.

It is important to note that such data organization is an approximation due to the movement of the LiDAR sensor through the environment during data acquisition. The effect of this is typically low and various motion compensation techniques can be applied in order to mitigate this effect and we therefore neglect it in the remainder of the paper.

B. Create Run Length Encoding of Line Labels

In our previous work on LiDAR point cloud clustering [7], [8] we segmented LiDAR data using connected components analysis in order to detect a ground plane. In that work, we analyzed the line segments formed by connecting the vertical neighbors within the point cloud stemming from a single revolution of a mechanical LiDAR. Should the difference in the inclination of such line segments be small, the next point is labeled as ground.

In this work we modify and extend that idea while using a similar approach to separate points that stem from different planar segments in the underlying data. Ideally for a set of points stemming from a vertical array of lasers shown in Fig. 3, we would like to make sure that points A, B and C end up being in the same connected component, while the points D, E and F are split into separate components resulting in the final point labels to be as indicated by the colors of the points in the figure.

In this section, we focus on separating the line segments formed by the points within a single vertical LiDAR scan (similar to points in Fig. 3) into connected components based on the differences in the line segment orientation. Because we only consider a single stack of lasers here we can talk about such points in 2D. The same logic can, of course, be applied in the horizontal direction without the loss of generality. That being said, it is less important to apply our approach in the horizontal direction as the distances between the points generated by the same laser diode are much smaller than the distances between the points generated by neighboring laser diodes, making it less likely for the points to land on different surfaces, thus making the usage of a more complex method harder to motivate.

The approach we use for finding connected components can be seen as a trivial case of the depth first search (DFS) on a graph that degenerates into a chain. The nodes of such a graph are line segments between the neighboring points and the weights on the edges represent differences in the line segment orientation computed by the following equation:

$$\alpha = \arccos \frac{\boldsymbol{v}_1 \cdot \boldsymbol{v}_2}{\|\boldsymbol{v}_1\| \|\boldsymbol{v}_2\|}. \tag{2}$$

Algorithm 1 Point Labeling

function LABELPOINTS(P, $\alpha_{\text{threshold}}$)
  $L^{\text{rle}} \leftarrow$ `EncodeLines`$(P, \alpha_{\text{threshold}})$
  $L^{\text{points}} \leftarrow$ `ExpandLabels`(P, L^{rle})
  return L^{points}
function ENCODELINES(P, $\alpha_{\text{threshold}}$)
  $\boldsymbol{v}_{\text{prev}} \leftarrow P_1 - P_0$
  $L^{\text{rle}} \leftarrow \{\}$
  $c \leftarrow 1$
  for $i = 1 \dots \|P\|$ **do**
    $\boldsymbol{v}_{\text{next}} \leftarrow P_{i+1} - P_i$
    $\alpha \leftarrow \angle(\boldsymbol{v}_{\text{prev}}, \boldsymbol{v}_{\text{next}})$ using Eq. (2).
    if $\alpha > \alpha_{\text{threshold}}$ **then**
      $L^{\text{rle}} \leftarrow L^{\text{rle}} \cup \{c\}$, $c \leftarrow 0$;
    $c \leftarrow c + 1$
    $\boldsymbol{v}_{\text{prev}} \leftarrow \boldsymbol{v}_{\text{next}}$
  $L^{\text{rle}} \leftarrow L^{\text{rle}} \cup \{c\}$
  return L^{rle}
function EXPANDLABELS(P, L^{rle})
  Initialize point labels: $L^{\text{points}} \leftarrow \{0\}$
  Previous component strength: $s_{\text{prev}} = 1$
  for $l = 0 \dots \|L^{\text{rle}}\|$ **do**
    Current component strength: $s \leftarrow L^{\text{rle}}_l$
    Index of disputed point: $i_d \leftarrow \|L^{\text{points}}\| - 1$
    if $s > 1$ **and** $s_{\text{prev}} == 1$ **then**
      $L^{\text{points}}_{i_d} \leftarrow l$
    else if $s > 1$ **and** $s_{\text{prev}} > 1$ **then**
      if $\|P_{i_d} - P_{i_d+1}\| < \|P_{i_d} - P_{i_d-1}\|$ **then**
        $L^{\text{points}}_{i_d} \leftarrow l$
    $L^{\text{points}} \leftarrow L^{\text{points}} \cup \{l\}^s$
    $s_{\text{prev}} \leftarrow s$
  return L^{points}

The connected component algorithm walks over a sequence of lines and compares if the angle difference between them is larger than a given threshold. If the angle difference is below the threshold, the next line is labeled with the same label as the previous one, otherwise it gets the next consecutive label.

To store the line labels along with the size of their component, that we call "strength", we make use of run length encoding (RLE) [27]. Let us assume for the sake of example that while performing the connected component labeling we ended up seeing such a string of line labels:

$$L^{\text{lines}} = \{0, 0, 0, 0, 1, 2, 2, 2, 3, 4, 5, 5, 5\}. \tag{3}$$

This list can be compactly represented with the RLE as a sequence of entries of the form $\{^n l\}$, where n is the number of times a label l appears in the label list. For an example shown above the RLE will look as follows:

$$L^{\text{rle}} = \{^4 0, ^1 1, ^3 2, ^1 3, ^1 4, ^3 5\}. \tag{4}$$

Utilizing the fact that our labels are consecutive integer numbers, we can further simplify the representation of such a list as an array of numbers indicating the "strength" of the component with index l : $[4, 1, 3, 1, 1, 3]$. Not only is this representation more compact than the full list of labels, it also makes the next step of labeling points easier.

Function `EncodeLines` in Alg. 1 shows this algorithm in detail. Here, P represents a single vertical stack of LiDAR points, with P_i representing the i-th point in it, $\alpha_{\text{threshold}}$ is the angle difference threshold, and c indicates the current component strength.

C. Point Labeling Using RLE

To compute the normals that do not cross the border of the underlying planar patches, we need to extend the labels of the line segments onto the points. We do this by walking over the L^{rle} values and filling the L^{points} as described in the function `ExpandLabels` in Alg. 1. The main idea of such label expansion is that we want to assign points that are adjacent to a "strong" component the label of that component. We call a component "strong" if its strength recorded in the L^{rle} is bigger than 1. Points that find themselves neighboring on two strong components get assigned to the one closest to them in Euclidean space. Points stuck between two weak components are considered to have high curvature. These points get an arbitrary label of one of their adjacent line labels and can be handled down the line. We can choose not to compute a normal in such points or mark them as points with high curvature and handle them later. The underlying assumption behind this method is that the points that form a "strong" components lie on smooth surfaces.

D. Normal Computation using Point Labels

Once the points are labeled, computing their normals follows the baseline approach described in Sec. III-A closely. We compute the normal with Eq. (1) using the cross product between the vectors spanning between the neighboring points. The main difference to the baseline approach, however, is that for a point with label l we only consider neighbors that have the same label l. This results in points that lie on the border between two labels to only consider their one neighbor that has the same label as themselves, improving the normal estimation in areas with high curvature. One corner case in such an approach appears when a point has both of its vertical neighbors having different label from itself. In such a case, we argue that no reliable normal can be computed and avoid normal computation of such a point in our experiments. That being said, these points can instead be marked as points exhibiting high curvature and dealt with at a later point in the pipeline.

The overall approach is shown in Alg. 1 but can be optimized when being efficiently implemented. While we show a two-step approach to aid explanation, further optimizations can be made that reduce the approach to a single pass over the data, yielding minimal overhead with respect to the baseline approach.

Our approach is a heuristic-based approach that relies on an assumption that an environment consists of multiple smooth components which might not be the case at all times. However, as we show in our experimental evaluation, our approach tends to work well in practice in a variety of scenarios.

IV. EXPERIMENTAL EVALUATION

The main focus of this work is a system that detects normals on sparse data produced by a single LiDAR. It is

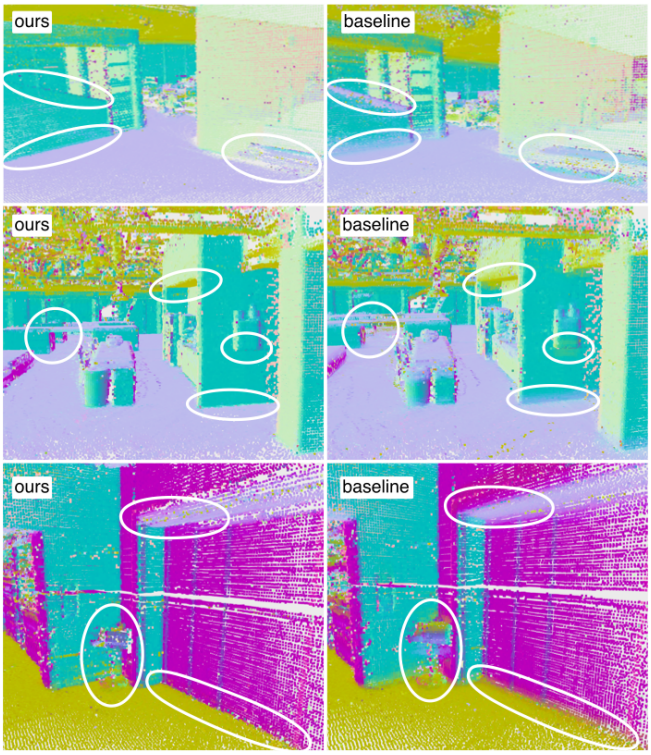

Fig. 4: A comparison of the robustness of the normals on various datasets. All of the images show point clouds as aggregated by our SLAM system described in Sec. IV-A with the colors representing the normals. Every row of images shows the map with our normals on the left and a map with baseline normals on the right. The top two rows are recorded with a single Velodyne VLP16 LiDAR through a NavVis M6 rig in our office, while the bottom row is reconstructed from the BASEMENT_3 sequence of the HILTI 2021 dataset [14] recorded with a single Ouster OS0 64 beam LiDAR that we subsample to 32 beams to simulate sparsity. The white ellipses highlight parts of the environment that are of interest and exhibit a substantial change in normal robustness. Note how our approach consistently provides sharper gradient in normals.

able to avoid a typical pitfall of baseline implementations by clustering the points that stem from a single vertical array of lasers and only considering the points with the same label for normal estimation.

To this end, the results of our experiments support our key claims, which are: (i) The proposed method for normal computation produces more robust normals in the areas of high curvature; (ii) The normals computation only incurs a linear runtime overhead with respect to the baseline method;

A. Experimental Setup

Before we dive into the experiments, we would like to outline a SLAM system used in some of these experiments. While this SLAM system is not part of the contributions of this work, we believe it is helpful to understand how it works to interpret the presented results better. We employ this system in all experiments that support our first claim of our method producing normals with better robustness in the high curvature parts of the environment.

Our SLAM system acquires data from a LiDAR sensor, reconstructs a point cloud, performs motion correction if necessary and works with such point clouds from that point on. As a first step, this point cloud is integrated into a LiDAR odometry system that produces a relative transformation with respect to the previous data. This transformation is then used to add this point cloud to the current submap represented as a voxel grid. Once such a submap has reached a certain size it gets post-processed and added to a node with this submap is added to a pose graph. The submap after the post-processing does not change. We also perform loop closures on the pose graph.

We omit many details here as we believe that they are not fully relevant for this work. What *is* important is that we estimate normals exclusively with the proposed method on single point clouds as soon as they get reconstructed. When these point clouds are processed with our LiDAR odometry system or are integrated into a submap, the normals of all the points that fall into a single voxel get averaged. Therefore, all the dense point clouds that we show in this section indicate the robustness of the normals as detected by our method.

B. Normal Robustness

The first experiment evaluates the performance of our approach and its outcome supports the claim that the proposed method of computing normals outperforms the baseline in terms of robustness of the produced normals. We evaluate the robustness of the normals in three key ways.

First, we assess the maps built with our SLAM system outlined in Sec. IV-A visually. Fig. 4 showcases a number of places in various environments. We recorded data at the Magic Leap office with a NavVis M6 system that sports a Velodyne VLP16 LiDAR, positioned horizontally. This LiDAR has 16 vertical laser beams that produce sparse range data. In addition to that we make use of publicly available BASEMENT_3 dataset from HILTI [14]. This dataset uses an Ouster OS0 64 beam LiDAR which produces much denser data. We take every second beam in order to showcase the performance of our method with sparse data. The places shown in Fig. 4 depict points colored by their normal. A sharper edge between two surfaces indicates less noise in the normals. The images clearly show that the edge between different surfaces is sharper when using the normals computed with the method presented in this paper and so shows their higher robustness.

Another way to evaluate the robustness of the normals is by utilizing them in the SLAM system. If a SLAM system uses normals extensively to align various range data then better normals should lead to better alignment. We performed a SLAM run on the BASEMENT_3 dataset from HILTI [14] using the standard normals as well as the proposed method. The results in Fig. 5 show that the error incurred by our SLAM system when using the proposed normal estimation method is lower than the one using the standard method, which proves that the normals generated with the proposed algorithm are more robust.

Last but not least, in Fig. 6 we show slices of data produced by a single array of vertical lasers from an Ouster OS0 64 LiDAR reduced to 32 beams (left side) and from

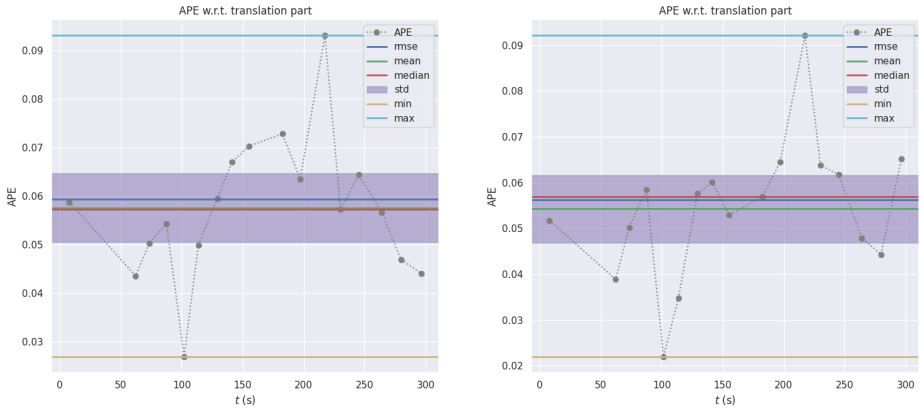

Fig. 5: Comparison of the APE (average percentage error) on poses estimated with our SLAM system described in Sec. IV-A as reported by HILTI challenge evaluation system on the BASEMENT_3 dataset. This dataset contains scans from Ouster OS0 64 beam LiDAR, which we subsample to 32 beams to increase data sparsity. Left image shows the APE computed from the poses using the baseline normals, while the right image shows the same APE computed from the poses estimated using our proposed method for normal estimation. Using our robust normals in our SLAM system that relies on normal estimation for point cloud alignment improves the quality of the resulting poses as can be seen by comparing the position of the lines. All the lines are lower in terms of APE when using our method for normal estimation (right image).

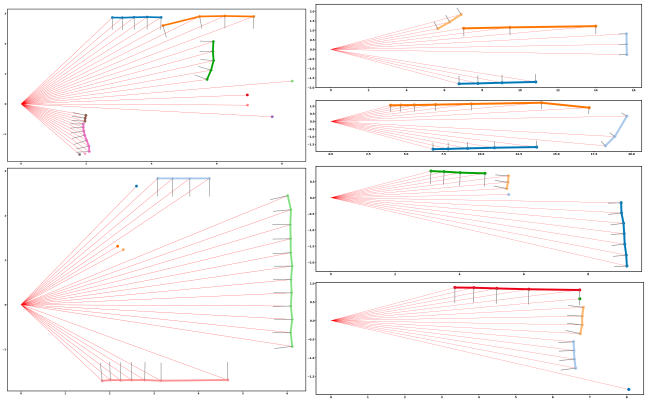

Fig. 6: A qualitative assessment of normals computed from 2D slices of 3D data. Every slice is generated from points produced by a single firing of the sensor array of lasers, similar to the illustration in Fig. 3. Left side shows 32 beams sampled from a 64 beam Ouster OS0 LiDAR, while the right side shows scans from a 16 beam Velodyne VLP16 LiDAR. The colors of the lines indicate their connected component label, gray lines show the normals estimated at points and the thin red lines show laser beams that generated the points of the scan. Note that some points are missing as the corresponding laser did not return. These results clearly show that our method is able to separate points on different underlying surfaces keeping the normals at fringe points sharp. Note also that standalone points do not have a normal.

Velodyne VLP16 LiDAR (right side). These slices are colored by the connected component to which the points and lines correspond and show the computed normals with gray lines. It is easy to see that the proposed algorithm is able to detect normals both inside the "strong" components as well as on their borders.

C. Runtime

Finally we analyze the runtime incurred by our method in comparison to the baseline method. While computing the normals with the proposed method is slower as more actions need to take place, we believe that this cost is warranted in order to achieve higher normals robustness. We report timing for both 16 beam as well as 32 beam LiDARs to show that our method only incurs a constant factor of around 2 as it does not change the $\mathcal{O}(n)$ complexity of the baseline method that computes normals. The results of this comparison can be found in Tab. I that shows the time it takes to compute normals of a single point cloud representing as a single revolution of either a 16 beam or a 32 beam sensor on an Intel® Xeon® 2.10GHz CPU.

TABLE I: Average runtime and std. dev.

Method	16 beams	32 beams
Our normals	1.02 ms ± 0.14 ms	2.54 ms ± 0.61 ms
Baseline normals	0.56 ms ± 0.02 ms	1.3 ms ± 0.26 ms

V. Conclusion

In this paper, we present a novel approach to compute normals on sparse LiDAR data. Our method exploits the organized structure of such data as well as an assumption that most of the surfaces in the environment are smooth. Utilizing this assumption and separating the points based on the change in the angle of the line segments connecting them allowed us to improve the robustness of the estimated normals while still maintaining competitive runtime of the method. We implemented and evaluated our approach on various datasets both qualitatively and quantitatively. The experiments suggest that our method outperforms the baseline method in terms of resulting normals robustness while incurring only a constant factor runtime overhead.

Acknowledgments

We thank Olga Vysotska for fruitful discussions on the topic of normal estimation and Cyrill Stachniss for his valuable comments.

REFERENCES

[1] P. Alliez, D. Cohen-Steiner, Y. Tong, and M. Desbrun. Voronoi-based variational reconstruction of unoriented point sets. In *Proc. of the Eurographics Symp. on Geometry Processing*, pages 39–48, 2007.

[2] N. Amenta and M. Bern. Surface reconstruction by voronoi filtering. *Proc. Fourteenth Annual Symp. on Comp. Geom.*, pages 39–48, 1998.

[3] H. Avron, A. Sharf, C. Greif, and D. Cohen-Or. l_1-sparse reconstruction on sharp pointset surfaces. In *ACM Trans. on Graphics (TOG)*, volume 29, 2010.

[4] H. Badino, D. Huber, Y. Park, and T. Kanade. Fast and accurate computation of surface normals from range images. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2011.

[5] J. Behley and C. Stachniss. Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments. In *Proc. of Robotics: Science and Systems (RSS)*, 2018.

[6] Y. Ben-Shabat, M. Lindenbaum, and A. Fischer. Nesti-net: Normal estimation for unstructured 3d point clouds using convolutional neural networks. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 10104–10112, 2019.

[7] I. Bogoslavskyi and C. Stachniss. Fast range image-based segmentation of sparse 3d laser scans for online operation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

[8] I. Bogoslavskyi and C. Stachniss. Efficient Online Segmentation for Sparse 3D Laser Scans. *Photogrammetrie – Fernerkundung – Geoinformation (PFG)*, 85(1):41–52, 2017.

[9] A. Boulch and R. Marlet. Deep learning for robust normal estimation in unstructured point clouds. *Computer Graphics Forum*, 35:281–290, 2016.

[10] N. Chebrolu, T. Läbe, and C. Stachniss. Spatio-Temporal Non-Rigid Registration of 3D Point Clouds of Plants. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2020.

[11] X. Chen, I. Vizzo, T. Läbe, J. Behley, and C. Stachniss. Range Image-based LiDAR Localization for Autonomous Vehicles. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021.

[12] T. Dey, G. Li, and J. Sun. Normal estimation for point clouds: A comparison study for a voronoi based method. In *Proc. of the Eurographics Symp. on Geometry Processing*, pages 39–46, 2005.

[13] P. Guerrero, Y. Kleiman, M. Ovsjanikov, and N.J. Mitra. Pcpnet: Learning local shape properties from raw point clouds. *Computer Graphics Forum*, 37(2):75–85, 2018.

[14] M. Helmberger, K. Morin, B. Berner, N. Kumar, D. Wang, Y. Yue, G. Cioffi, and D. Scaramuzza. The hilti slam challenge dataset, 2021.

[15] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *Proc. of the Intl. Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 1992.

[16] J. Hyeon, W. Lee, J.H. Kim, and N. Doh. Normnet: Point-wise normal estimation network for three-dimensional point cloud data. *Intl. Journal of Advanced Robotic Systems*, 16, 2019.

[17] J. Lenssen, C. Osendorfer, and J. Masci. Deep iterative surface normal estimation. *Proc. of British Machine Vision Conference (BMVC)*, pages 11244–11253, 2020.

[18] B. Li, R. Schnabel, R. Klein, Z. Cheng, G. Dang, and S. Jin. Robust normal estimation for point clouds with sharp features. *Computer & Graphics*, 34(2):94–106, 2010.

[19] Q. Li, Y.S. Liu, J.S. Cheng, C. Wang, Y. Fang, and Z. Han. Hsurf-net: Normal estimation for 3d point clouds by learning hyper surfaces. In *Proc. of the Advances in Neural Information Processing Systems (NIPS)*, 2022.

[20] X. Lu, S. Schaefer, J. Luo, L. Ma, and Y. He. Low rank matrix approximation for 3d geometry filtering. *IEEE Trans. on Visualization and Computer Graphics*, 28(4):1835–1847, 2022.

[21] R. Marcuzzi, L. Nunes, L. Wiesmann, J. Behley, and C. Stachniss. Mask-Based Panoptic LiDAR Segmentation for Autonomous Driving. *IEEE Robotics and Automation Letters (RA-L)*, 8(2):1141–1148, 2023.

[22] N.J. Mitra and A. Nguyen. Estimating surface normals in noisy point cloud data. In *Proc. of the Nineteeth Annual Symp. on Computation Geometry*, pages 322–328, 2003.

[23] L. Nunes, X. Chen, R. Marcuzzi, A. Osep, L. Leal-Taixé, C. Stachniss, and J. Behley. Unsupervised Class-Agnostic Instance Segmentation of 3D LiDAR Data for Autonomous Vehicles. *IEEE Robotics and Automation Letters (RA-L)*, 7(4):8713–8720, 2022.

[24] L. Pedersen, M.B. Allan, H. Utz, M.C. Deans, X. Bouyssounouse, Y. Choi, L. Fluckiger, S.Y. Lee, V. To, J. Loh, et al. Tele-operated lunar rover navigation using lidar. In *Proc. of the Intl. Symp. on Artificial Intelligence, Robotics and Automation in Space (I-SAIRAS)*, 2012.

[25] C.R. Qi, H. Su, K. Mo, and L.J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[26] G. Rivera, R. Porras, R. Florencia, and J.P. Sánchez-Solís. Lidar applications in precision agriculture for cultivating crops: A review of recent advances. *Computers and Electronics in Agriculture*, 207:107737, 2023.

[27] A. Robinson and C. Cherry. Results of a prototype television bandwidth compression scheme. In *Proceedings of the IEEE*, volume 55, pages 356–364, 1967.

[28] Y. Sun, S. Schaefer, and W. Wang. Denoising point sets via l_0 minimization. *Computer Aided Geometric Design*, 35–36:2–15, 2015.

[29] I. Vizzo, X. Chen, N. Chebrolu, J. Behley, and C. Stachniss. Poisson Surface Reconstruction for LiDAR Odometry and Mapping. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021.

[30] I. Vizzo, T. Guadagnino, B. Mersch, L. Wiesmann, J. Behley, and C. Stachniss. KISS-ICP: In Defense of Point-to-Point ICP – Simple, Accurate, and Robust Registration If Done the Right Way. *IEEE Robotics and Automation Letters (RA-L)*, 8(2):1029–1036, 2023.

[31] W. Wang, X. Lu, D. de Silva Edirimuni, X. Liu, and A. Robles-Kelly. Deep point cloud normal estimation via triplet learning. In *Proc. of the IEEE Intl. Conf. on Multimedia and Expo*, 2022.

[32] Z. Wang and V. Prisacariu. Neighbourhood-insensitive point cloud normal estimation network. *Proc. of British Machine Vision Conference (BMVC)*, 2020.

[33] M. Yoon, Y. Lee, S. Lee, I. Ivrissimtzis, and H.P. Seidel. Surface and normal ensembles for surface reconstruction. *Computer-Aided Design*, 39(5):408–420, 2007.

[34] Z. Yu, T. Wang, T. Guo, H. Li, and J. Dong. Robust point cloud normal estimation via neighborhood reconstruction. *Advances in Mechnical Engineering*, 11(4), 2019.

[35] J. Zhang and S. Singh. LOAM: Lidar Odometry and Mapping in Real-time. In *Proc. of Robotics: Science and Systems (RSS)*, 2014.

[36] J. Zhang, J. Cao, X. Liu, J. Wang, J. Liu, and X. Shi. Point cloud normal estimation via low-rank subspace clustering. *Computer & Graphics*, 37(6):697–706, 2013.

[37] X.L.J. Zhang, J. Cao, B. Li, and L. Liu. Quality point cloud normal estimation by guided least squares representation. *Computer & Graphics*, 51:106–116, 2005.