Journal of Autonomous Intelligence (2024) Volume 7 Issue 5
doi: 10.32629/jai.v7i5.1634

ORIGINAL RESEARCH ARTICLE

Detecting people in sprinting motion using HPRDenoise: Point cloud denoising with hidden point removal

Taku Itami[1,*], Yuki Takeyama[2], Sota Akamine[1], Jun Yoneyema[1], Sebastien Ibarboure[3]

[1] *Department of Electrical and Electronic Engineering, Aoyama Gakuin University, Shibuya-ku, Tokyo 252-5258, Japan*

[2] *Department of Computer Science, University of Tsukuba, Tsukuba-shi, Ibaraki 305-8577, Japan*

[3] *MEDIA ROBOTECH INC., Sagamihara-shi, Kanagawa 252-0237, Japan*

*** Corresponding author:** Taku Itami, itami@ee.aoyama.ac.jp

ABSTRACT

LiDARs are utilized in various applications, such as self-driving vehicles and robotics, to aid in sensing the environment. However, LiDARs do not provide instantaneous images and they generate noise, adding to measurement errors. This noise, often referred to as motion blur phenomenon also observed in other imaging sensors results in decreased sensing accuracy for moving objects. This study introduces HPRDenoise, a noise reduction method based on hidden point removal, specifically designed to reduce motion blur during sprinting motion. This method capitalizes on the occlusion produced by a fixed-position LiDAR. We propose a comprehensive denoising approach to filter points from a point cloud without resorting to supervised learning, unlike most existing denoising algorithms. The number of correct frames and accuracy were compared for Raw, ScoreDenoise, which is the state-of-the-art method for random point cloud denoising, and HPRDenoise (Ours). Accuracy is defined as the ratio of the number of correct frames to the total number of frames. Experimental results demonstrate that the detection accuracy of point clouds processed with HPRDenoise is 72.73%, achieving better accuracy than those using conventional methods.

Keywords: LiDAR; motion blur; noise reduction; hidden point removal

ARTICLE INFO

Received: 7 March 2024
Accepted: 22 March 2024
Available online: 3 April 2024

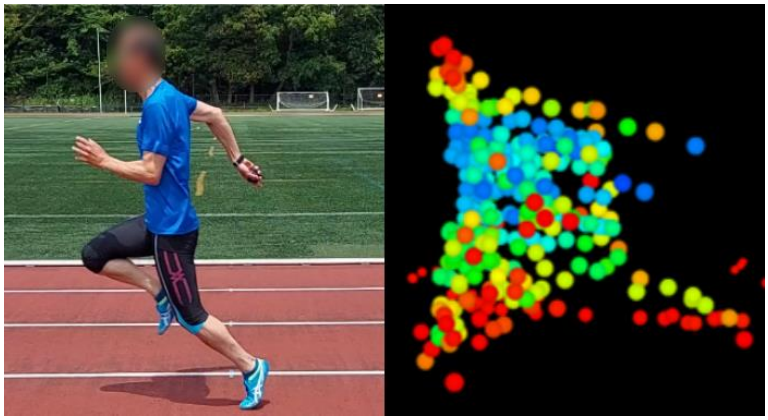
1. Introduction

The 100-meter race is one of the most renowned athletic disciplines, known for pushing the limits of the human body. Sprinting is an intense activity that engages the entire body in maintaining a high level of effort. Sprinting, as a motor learning skill, can be honed through repeated practice. Consequently, athletes must not only be fast but also consistent, and biomechanical science plays a crucial role in determining the optimal sprinting form. One area of focus is the study of the gait cycle, which describes the typical motion of sprinting, particularly the leg's kinematics. To achieve this, researchers must accurately capture the entire body's motion using either camera-based or inertial sensor-based motion capture systems[1].

However, such equipment necessitates a meticulous and costly network of cameras that must be calibrated, making it more suited for laboratory environments[2]. Alternatively, using multiple sensors mounted on different segments of a sprinter's body can be obstructive and potentially impact their performance, especially

during competitions where data collection is most valuable.

In recent years, LiDARs have become increasingly popular in a variety of applications, particularly in the fields of self-driving vehicles and robotics[3–5], where they help capture the geometry of the surrounding environment. LiDARs can rapidly generate extensive 3D point clouds at both close and long ranges. As a non-intrusive solution, these sensors can be used to capture the complete motion of a pedestrian in the form of a 3D point cloud[4]. In order to analyze a pedestrian's motion from the point cloud, it is necessary to identify the cluster of points corresponding to the target and extract their posture at various time intervals. However, as depicted in **Figure 1**, LiDARs tend to produce noisy images of dynamic objects. This issue, commonly referred to as motion blur, is present in images captured by all sensor types, including video and thermal cameras[6]. The extent of motion blur is influenced by the scanning time of the sensor and increases with the relative difference between the tracked object's speed and the scanning speed.



(a) **(b)**

Figure 1. An example of noise caused by scanning time, **(a)** tracked object; **(b)** 3D point cloud data with noise caused by scanning time.

Motion blur is particularly noticeable in LiDARs with high-density coverage, such as non-repeatable scanning LiDARs, which are well-suited for capturing small-profile objects like pedestrians. In the context of this study, where the tracked object is both fast and deformable, motion blur poses a significant challenge to achieving robust 3D object detection and pose estimation. In particular, the ability to detect a person in a sprinting motion, which is the target of this research, is expected to be applied to person detection by vehicle-mounted Lidar and to environmental recognition.

This paper addresses the issue of motion blur in 3D object detection, especially during sprinting motion. First, we provide an overview of state-of-the-art techniques for object tracking and denoising, allowing for comparison with our proposed method. Next, we introduce HPRDenoise, a solution based on the hidden point removal algorithm for motion blur elimination, which utilizes a combination of arbitrary algorithms that do not require supervised learning, as previously employed. Finally, we evaluate the effectiveness of our proposed method using experimental data from a sprinter acquired with a LIVOX LiDAR Tele 15.

2. Related work

2.1. Point cloud object detection

Previous research[7–10] has developed a variety of neural network architectures to enhance detection performance. PointPillars[8] proposed a novel encoder consisting of vertical columns, or "pillars," using PointNet[11] to learn point cloud representations. CenterPoint[7] introduced an anchor-free method. Traditional approaches like PointPillars perform a matching search within the detection area using a bounding box reference, known as an anchor. Objects whose Intersection over Union (IoU) exceeds a certain threshold are

then learned. Anchor-based methods can only efficiently learn objects if the search method and thresholds are correctly determined. Therefore, a heuristic parameter search must be conducted to ensure more objects exceed the IoU thresholds. In contrast, anchor-free methods learn object centers as heatmaps without using anchors, enabling them to detect objects of various shapes. This algorithm is a state-of-the-art approach on the nuScenes dataset[4], and as of 2021, it is the highest performing object detection algorithm in the LiDAR-Only category on the Waymo dataset[5]. However, noise created by fast-moving objects, due to the LiDAR sensor's laser scanning time, hampers further improvements in the accuracy and robustness of 3D object detection.

It is common practice to train supervised learning-based 3D object detection models on primary datasets[3–5] and evaluate their detection performance on these datasets. These datasets include 3D point clouds, images, and semantic labels under various circumstances. Waymo Open Dataset[5] classifies objects into four labels: vehicles, pedestrians, cyclists, and signs, while the KITTI Vision Benchmark Suite[3] divides them into six labels: car, van, truck, pedestrian, person_sitting, and cyclist. Unlike the two previous datasets, nuScenes adds additional descriptors to simple object detection. Specifically, these descriptors aid in discriminating between different types of pedestrians and further determine if they are moving or stationary. Fast-moving and deformable objects are more likely to generate motion blur, making it difficult to label them. Therefore, among the significant datasets, only nuScenes assigns status such as pedestrian.moving. Both the KITTI Dataset and nuScenes utilize Velodyne LiDAR to generate their respective point clouds, which are characterized by a parallel, repeating scanning pattern with a low density of coverage. Furthermore, the point clouds of pedestrians in these datasets exhibit little to no motion blur. These observations underscore the importance of investigating denoising techniques to improve detection rates, especially when aiming to detect fast-moving objects in a blurry point cloud. Various other object detection methods based on deep learning have also been studied[12–14].

2.2. Point cloud denoising

Point cloud data is often perturbed by noise due to inherent limitations of the acquisition devices and ambiguity in matching during reconstruction from images. Noise present in point clouds can significantly impact downstream tasks such as rendering, reconstruction, and analysis, as it can cause fundamental structures to be deformed. Therefore, noise removal from point clouds is crucial for related 3D vision applications, but it is challenging due to the irregular and unordered nature of point cloud noise[15].

Early point cloud denoising methods[16–19] were optimization-based and heavily relied on geometric priors, making it challenging to balance detail preservation and noise removal effectiveness. In recent years, the emergence of neural networks designed for point clouds[11,20,21] has led to the development of supervised learning-based approaches[22–25] that have achieved promising noise removal performance. Most supervised learning-based noise removal models predict the displacement from the base surface of the noisy point cloud and apply the inverse displacement to the noisy point cloud. In particular, ScoreDenoise[15] achieved the best performance for the supervised learning-based denoising models.

However, all these noise removal methods are applied to point clouds scanned for the entire object. LiDAR sensors installed in automatic driving and robots often only scan a part of an object. Moreover, when the scanning target is moving too fast, it is difficult to scan the target entirely for each frame. There has been little research on removing noise caused by scanning time from real-time point clouds obtained from a static LiDAR. Furthermore, these noise removal methods have the problem of being dependent on training data, as they are based on supervised learning.

There is scant research on motion blur removal for point clouds, with the exception of Lin and Zhang[26]. In their study, two methods were proposed to mitigate motion blur for LiDAR Odometry and Mapping (LOAM). The first approach divides the input frame into three consecutive subframes, each of

which is independently matched to the same accumulated map. The second strategy employs linear interpolation based on the LiDAR's pose. The first technique is beneficial for mapping, as there is a ground truth in the form of the accumulated map, and a sufficient number of point clouds can be obtained for 3D mapping of the environment. even when a single frame is divided. However, in object detection where the target is in motion, motion blur can't be effectively eliminated by merely dividing the point cloud, since there is not necessarily a ground truth to support the motion blur removal. Furthermore, the second method hinges on estimating two consecutive LiDAR poses to correct motion blur, and therefore, it is incapable of eliminating the blur produced by a moving target object.

Consequently, in this study, we endeavor to formulate a method to eliminate noise caused by an object moving in front of a static LiDAR, using an array of empirical algorithms as opposed to supervised learning.

3. Method

3.1. Hidden point removal

The method we propose uses point occlusion to remove motion blur. In our context, occlusion refers to areas of the point cloud where data is missing, due to the characteristic of LiDAR only returning measurements of surfaces in direct line of sight. Our proposed method involves extracting the surface portions of the point cloud as correct, and removing the parts recognized as occlusions, that is, the areas beyond the surface. More specifically, we propose a novel application of the hidden point removal algorithm. This algorithm extracts only the points visible from a specific viewpoint by projecting the point cloud onto a sphere[27].

In our case study, we aim to eliminate the motion blur generated by a sprinter running towards the LiDAR. This motion blur causes a local deformation of the point cloud, stretching the image of the sprinter along his path. We then seek to retain only the foremost points in order to reconstruct the surface of the sprinter's body. The proposed method enables the detection of persons even during sprinting motion, and is expected to detect persons whose motion exceeds the resolution of LiDAR in certain environments.

3.2. HPRDenoise

Figure 2 shows an overview of the developed noise reduction filter in one frame. First, ground removal is performed using the RANSAC algorithm[28,29] to search for the largest plane. The tolerance error for RANSAC is defined as ransac_error and is used for this process. Next, clustering is performed using the HDBSCAN algorithm[29,30]. HDBSCAN is an extension of the DBSCAN algorithm and can automatically determine the number of clusters by considering the distance between clusters during clustering.

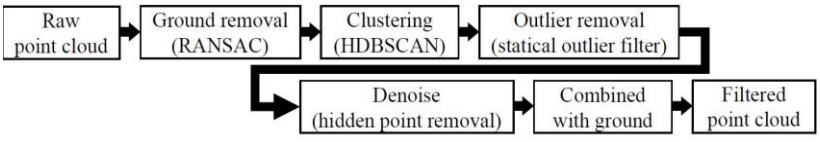


Figure 2. An overview of the proposed filter that can be applied overall: Ground removal using RANSAC, clustering using HDBSCAN, outlier removal using statistical outlier filter, noise removal using hidden point removal, and finally, merging with the ground.

Then, the statical outlier filter is applied to each clustered point cloud to remove outliers, which ensures the size of each point cloud cluster. This is important for improving the accuracy of the hidden point removal, as the size of the point cloud cluster is considered when specifying the sphere radius for hidden point removal. The statical outlier filter is defined by Equations (1) and (2). In the Equation (1), μ_i is the average distance to the k nearest neighbors of x_i, k is the number of neighbors and $d(x_i, x_j)$ is the distance between x_i and x_j.

$$\mu_i = \frac{1}{k}\sum_{j=1}^{k} d\,(x_i, x_j) \tag{1}$$

The overall average distance $\bar{\mu}$ and standard deviation σ_μ are calculated from the individual distances μ_i for each point. Then, points that satisfy the Equation (2) are judged to be outliers and are removed.

$$\bar{\mu}_i \geq \bar{\mu} + m \times \sigma_\mu \tag{2}$$

The number of neighbors k in Equation (1) and the coefficient m for the standard deviation in Equation (2) are specified as parameters. After clustering, hidden point removal algorithm is applied to each point cloud cluster to remove noise caused by scanning time. Hidden point removal can only extract the visible points from a viewpoint by specifying the viewpoint and sphere radius. The determination of the hidden point removal sphere radius and viewpoint coordinates is described in subsections 3.3, 3.4. Finally, we merge the points with the ground to reconstruct the complete point cloud for a single frame.

In this paper, motion blur can be removed as occlusion by applying HPRDenoise as shown in **Figure 3**. Hidden point removal must be specified in terms of viewpoint coordinates and projected sphere radius. The specification methods are shown in sections 3.3 and 3.4.

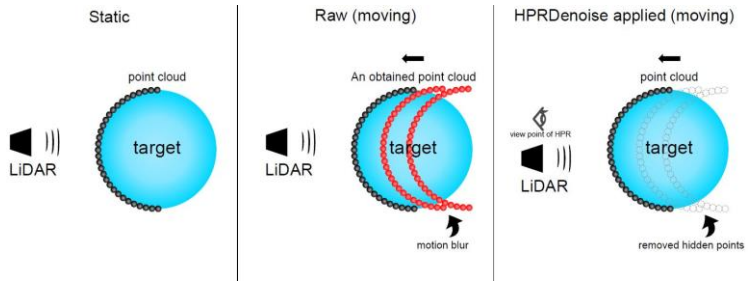


Figure 3. Planar observation of 3D point clouds in various situations from directly above.

3.3. How to determine the sphere radius r

The sphere radius r is defined by Equations (3)–(5), where x_{all}, y_{all}, z_{all} represent the *x*-coordinates, *y*-coordinates, and *z*-coordinates of the point cloud within a single frame, respectively. In this context, $\vec{v}_{\max}$ signifies the respective maximum values in x_{all}, y_{all}, z_{all}, while $\vec{v}_{\min}$ denotes the respective minimum values. The term n serves as the coefficient for the sphere radius.

$$\vec{v}_{\max} = \begin{pmatrix} \max\,(x_{\text{all}}) \\ \max\,(y_{\text{all}}) \\ \max\,(z_{\text{all}}) \end{pmatrix} \tag{3}$$

$$\vec{v}_{\min} = \begin{pmatrix} \min\,(x_{\text{all}}) \\ \min\,(y_{\text{all}}) \\ \min\,(z_{\text{all}}) \end{pmatrix} \tag{4}$$

$$r = n \times |\vec{v}_{\max} - \vec{v}_{\min}| \tag{5}$$

The magnitude of $|\vec{v}_{\max} - \vec{v}_{\min}|$ ensures that the sphere radius r varies in accordance with the size of the point cloud cluster. Furthermore, the coefficient n influences the number of points visible from the viewpoint when the hidden point removal algorithm is applied.

3.4. How to determine viewpoint coordinates $\vec{v}_v$

The coordinate of the viewpoint $\vec{v}_v$ (see **Figure 4**) is obtained by Equation (6) using the similarity of triangles, where the origin o is the origin of the LiDAR, v_o is the position of the target object and the distance between the center of the target object and the viewpoint is d_v. This enables the appropriate placement of the viewpoint. Moreover, by placing the viewpoint along the straight line connecting the

LiDAR and the target object, only the surface point cloud visible from the LiDAR can be extracted.

$$\vec{v}_v = \vec{v}_o \times (1 - \frac{d_v}{\sqrt{x_0^2 + y_0^2 + z_0^2}}) \tag{6}$$

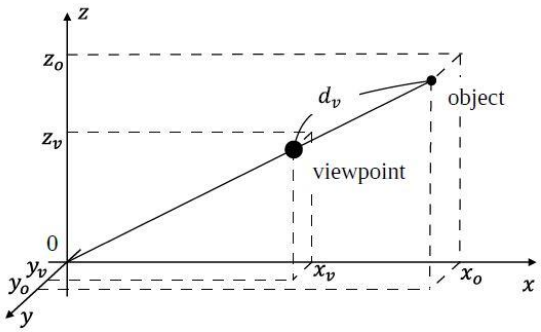


Figure 4. How the viewpoint coordinates are determined. The object is the center coordinate of the point cloud cluster from which the noise is removed, and the origin *o* is the location of the LiDAR. The viewpoint is placed at a point d_v away from the object in a straight line connecting the object and LiDAR.

4. Experiments

4.1. Setup

The experiment took place at an official training facility responding to the standard of the discipline. The test was run with a healthy semi-professional male sprinter, who was asked to run several 100 m sprinters with enough time to rest between trials. Point clouds were collected using LiDAR LIVOX Tele 15, which collects points at a rate of 240,000 points/s (strongest return) within a field of view of 14.5° × 16.2°, under an angular precision of 0.03°. Obtained point clouds are then sampled into frames every 100 ms (10 Hz), which is the time needed for full coverage of the LiDAR FOV. The LiDAR was installed facing toward the start line and placed behind the finish line (see **Figure 5**). Note that the parameters of HPRDenoise are shown in **Table 1**. The ransac_error is the RANSAC error tolerance, *k*, *m* are arbitrary constant parameters in the Equations (1) and (2).

Figure 5. The experimental environment where point clouds were acquired. The distance from the start line to the finish line was 100 m. The LiDAR was installed facing towards the start line and placed behind the finish line.

We processed a total of 154 frames containing motion blur. We conducted a comparative experiment with ScoreDenoise[15], which is the state-of-the-art method for random point cloud denoising. For each frame, we prepared point cloud data with ScoreDenoise[15] applied to only the sprinter cluster, point cloud data with HPRDenoise filter (Ours) applied to the entire point cloud, and raw point cloud data. The parameters used in the HPRDenoise filter (Ours) for the statistical outlier filter and RANSAC are listed in **Table 1**. Ransac_error is the error tolerance for RANSAC. *k*, *m* are arbitrary constant parameters in the Equations (1) and (2).

Table 1. Parameters of HPRDenoise. Ransac_error is the RANSAC error tolerance. *k*, *m* are arbitrary constant parameters in the Equations (1) and (2).

Content	Value
Ransac_error	0.1
k (Equation (1))	20
m (Equation (2))	2.0

For the object detection algorithm, we used livox detection[31]. Livox detection is a ROS-based point cloud object detection application by livox, based on CenterPoint[7]. We used CenterPoint because it is a state-of-the-art algorithm on some datasets. As a pre-trained model for livox detection, we used the one from the official livox detection repository (livox_model_2)[31]. Livox detection uses not only the 3D point cloud coordinates but also the reflectivity as input. Therefore, for the point cloud data to which the ScoreDenoise[15] filter was applied, the reflectivity of the nearest neighbor points of the raw point cloud data was used.

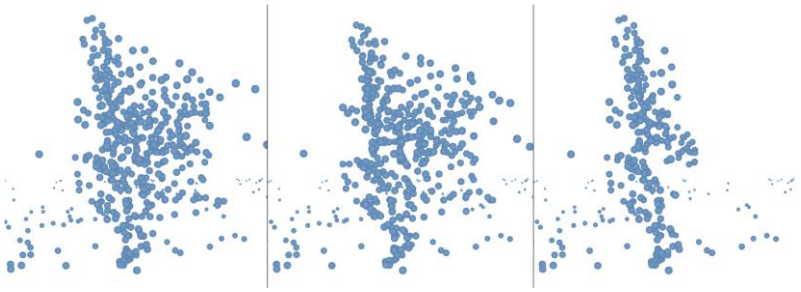
The number of correct frames and accuracy were compared for Raw, ScoreDenoise, and HPRDenoise (Ours). Accuracy is defined as the ratio of the number of correct frames to the total number of frames. Typically, the evaluation metrics for object detection models are calculated using recall, precision, and average precision (AP). However, in this study, we used accuracy at the frame level as the evaluation metric. This is because it is assumed that there is only one person to be detected in each frame. For evaluation, we adopted the method used in nuScenes[4], which calculates accuracy by assigning true/false values based on the threshold distance between object centers. However, unlike in nuScenes, we set the threshold to a distance of 2 m between centers in this study. This decision was made because we used the average coordinates of the human point cloud cluster as the center coordinates in the correct frame, which could potentially introduce errors in the center coordinates of the correct answers. For each method, an answer was considered correct if the detected object was classified as a pedestrian and the distance between the center coordinates of the detected object and the actual point cloud cluster was less than 2 m (in the $x-y$ coordinate system that is in a plane parallel to the ground). The specific filter application procedure is described below.

(1) Obtain 3D point clouds from Livox LiDAR as lvx files.
(2) Convert lvx files to rosbag files using Livox Ros Driver[30].
(3) Output a ROS PointCloud2 type topic from the rosbag file, convert it to pcd and save it frame by frame.
(4) The continuous frames with dynamic noise are filtered to remove the noise. In hidden point removal, the viewpoint d_v and the spherical radius n are set to the following values. d_v = 5, 10, n = 800, 3000.
(5) Dynamic object detection evaluation is performed through Livox detection of the respective point clouds of Raw, ScoreDenoise, and HPRDenoise (Ours). Here, if the center point of the detected object is less than the threshold distance (2 m) from the center point of the correct object, it is counted as the correct answer.
(6) The accuracy, which is the ratio of the number of correct frames to the total number of frames, is compared for each result.

4.2. Results

Table 2 shows the number of correct frames and accuracy results when n in the Equation (5) is changed to 800, 3000 and d_v in the Equation (6) is changed to 5, 10. **Table 2** shows that the highest accuracy is obtained when n = 800, d_v = 5. In our experiments, we determined the parameters n, d_v empirically based on our experience. **Table 3** shows the results of the correct frames, accuracy and average processing time between each frame of HPRDenoise against a conventional method and raw data. n, d_v values are those with the highest accuracy in **Table 2**. As shown in **Table 3**, the accuracy when HPRDenoise is applied is higher than the accuracy for raw point cloud data. Furthermore, it was found that our method achieved higher accuracy compared to ScoreDenoise, which was a significant denoising filter in conventional methods. The raw point cloud data, the resulting point cloud after applying ScoreDenoise and the resulting point cloud after applying HPRDenoise for the set of parameter n = 800, d_v = 5 for a randomly picked frame are shown in **Figure 6**. Furthermore, a comparison of the average processing time between frames showed that it was 91.83 for the raw data, 93.01 for the ScoreDenoise, and 91.88 for the HPRDenoise. In other words, the processing time of the proposed method is almost the same as that of the raw data.

From **Figure 6**, it can be observed that the raw point cloud data displays a scattered point cloud of the person in the direction of movement. However, with ScoreDenoise, the scattered point clouds appear to cluster a bit more closely. Furthermore, HPRDenoise illustrates the extraction of a point cloud from the surface.



(a) **(b)** **(c)**

Figure 6. Comparison of each method; **(a)** raw point cloud data in an arbitrary frame, viewed from the side; **(b)** point cloud data with ScoreDenoise applied in an arbitrary frame, viewed from the side; **(c)** point cloud data with HPRDenoise (Ours) applied using parameters n = 800, d_v = 5 in an arbitrary frame, viewed from the side.

Table 2. Number of correct frames for n, d_v settings, and accuracy results. n, d_v are arbitrary constant parameters in the Equations (5) and (6). It can be seen that the accuracy varies depending on the parameters.

Method		**Correct frames**	**Accuracy (%)**
n = 800	d_v = 5	113	73.38
	d_v = 10	105	68.18
n = 3000	d_v = 5	73	47.40
	d_v = 10	112	72.73

Table 3. The number of correct frames and accuracy results for each method. Raw refers to point cloud data without any processing, ScoreDenoise refers to point cloud data processed by a conventional noise reduction method[15], and HPRDenoise refers to point cloud data processed by our proposed method.

Method	**Correct frames**	**Accuracy (%)**	**Ave. process time (ms)**
Raw	20	12.99	91.83
ScoreDenoise	44	28.57	93.01
HPRDenoise	113	73.38	91.88

5. Discussion

As shown in **Table 2**, the accuracy varies depending on the value of n, d_v. **Table 3** shows that our proposed method achieves higher accuracy than the conventional method.

When the viewpoint of the hidden point removal method and the viewpoint of the LiDAR are aligned in the same direction, this method is able to remove motion blur accurately. In other words, if hidden point removal is used when a dynamic object is moving toward or away from the LiDAR, accurate dynamic noise reduction is possible, and object detection accuracy is expected to be improved. In this experiment, the effectiveness of the method was verified when the scanning target was moving toward the LiDAR, but the method is expected to be effective even when the target is moving away from the LiDAR. When moving toward the diagonal or lateral direction, accurate noise reduction is complex, and no improvement in accuracy may be seen.

In some cases, the application of a filter does not result in an improvement in accuracy. There are mainly two factors that can explain this phenomenon. The first is that the person is not recognized as a person due to factors other than noise. For example, if there is not enough training data similar to the unrecognized frame, it cannot be accurately recognized. Second, hidden point removal only partially

removes dynamic noise in all frames. The proposed method must be revised to handle situations involving lateral movements, such as transitioning from an open position to a closed one. In such situations, the hidden point removal algorithm will fail to distinguish the different movement transitions since all points already face the sensors.

From the experimental results, this method is effective for LiDARs with non-repeating scan patterns. However, it is also expected to be effective for LiDARs with repeating scan patterns less prone to noise generated by scanning time. In addition, considering that HPRDenoise has the ability to denoise multiple clusters, it should also prove effective for denoising multiple individuals, even though this experiment involved denoising only a single person.

However, it was difficult to identify other methods that are readily available to specifically remove motion blur affecting a moving object inside a larger point cloud for our experiment. In addition, the resource needed to implement multiple solutions in parallel where limited, de facto, we focus our time on the method that we thought to be the most promising (ScoreDenoise) and on the proposed method. This is because ScoreDenoise assumes that all surfaces have been scanned, so it is not possible to remove motion blur itself.

6. Conclusion

This paper proposes a motion blur filtering method based on the hidden point removal algorithm when tracking a fast-moving object using static LiDAR. The proposed method is a step-by-step methodology applied on each point cloud frame to extract and denoise the target point cloud locally. Unlike the supervised learning approach mentioned in this paper, the proposed methodology uses unsupervised learning, allowing us to avoid the problem of training data dependence. Experimental results show that the proposed method is promising, with an accuracy of up to 75% and thus outperforming existing techniques. In the future, we intend to extend our approach to handle lateral motion and thus extend our methodology to any random fast-moving object into a linear trajectory. With the progress of this research, it is expected to be able to detect people even when using Lidar, which has low resolution. Therefore, it is expected to be applied not only to the detection of runners, but also to the estimation of people with uncertain movements in urban areas, which has been difficult to achieve with in-vehicle systems, for example.

Author contributions

Conceptualization, TI and YT; methodology, YT and SI; software, YT; validation, TI, YT, SA and SI; formal analysis, YT; investigation, TI and YT; data curation, TI and YT; writing—original draft preparation, YT; writing—review and editing, TI, SA and SI; supervision, TI and JY; project administration, TI; funding acquisition, TI. All authors have read and agreed to the published version of the manuscript.

Acknowledgments

This work was supported by MEDIA ROBOTECH INC.

Conflict of interest

The authors declare no conflict of interest.

References

1. Pueo B, Jimenez-Olmedo JM. Application of motion capture technology for sport performance analysis. Retos. 2017; (32): 241-247. doi: 10.47197/retos.v0i32.56072
2. Rekant J, Rothenberger S, Chambers A. Inertial measurement unit-based motion capture to replace camera-based systems for assessing gait in healthy young adults: Proceed with caution. Measurement: Sensors. 2022; 23: 100396. doi: 10.1016/j.measen.2022.100396

3. Geiger A, Lenz P, Urtasun R. Are we ready for autonomous driving? The KITTI vision benchmark suite. 2012 IEEE Conference on Computer Vision and Pattern Recognition. Published online June 2012. doi: 10.1109/cvpr.2012.6248074
4. Caesar H, Bankiti V, Lang AH, et al. nuScenes: A Multimodal Dataset for Autonomous Driving. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Published online June 2020. doi: 10.1109/cvpr42600.2020.01164
5. Sun P, Kretzschmar H, Dotiwalla X, et al. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Published online June 2020. doi: 10.1109/cvpr42600.2020.00252
6. Yang W, Gong Z, Huang B, et al. Lidar with Velocity: Correcting Moving Objects Point Cloud Distortion from Oscillating Scanning Lidars by Fusion with Camera. IEEE Robotics and Automation Letters. 2022; 7(3): 8241-8248. doi: 10.1109/lra.2022.3187506
7. Yin T, Zhou X, Krahenbuhl P. Center-based 3D Object Detection and Tracking. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Published online June 2021. doi: 10.1109/cvpr46437.2021.01161
8. Lang AH, Vora S, Caesar H, et al. PointPillars: Fast Encoders for Object Detection from Point Clouds. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Published online June 2019. doi: 10.1109/cvpr.2019.01298
9. Shi S, Guo C, Jiang L, et al. PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Published online June 2020. doi: 10.1109/cvpr42600.2020.01054
10. Shi W, Rajkumar R. Point-GNN: Graph Neural Network for 3D Object Detection in a Point Cloud. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Published online June 2020. doi: 10.1109/cvpr42600.2020.00178
11. Charles RQ, Su H, Kaichun M, et al. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Published online July 2017. doi: 10.1109/cvpr.2017.16
12. Yin S, Li H, Laghari AA, et al. An Anomaly Detection Model Based on Deep Auto-Encoder and Capsule Graph Convolution via Sparrow Search Algorithm in 6G Internet-of-Everything. IEEE Internet of Things Journal. Published online 2024: 1-1. doi: 10.1109/jiot.2024.3353337
13. Zhao Z, Zheng P, Xu S, Wu X. Object Detection Based on Deep Learning: A Brief Review. IJLAI Transactions on Science and Engineering. 2023; 1(2): 1-6.
14. Jiang M, Yin S. Facial expression recognition based on convolutional block attention module and multi-feature fusion. International Journal of Computational Vision and Robotics. 2023; 13(1): 21. doi: 10.1504/ijcvr.2023.127298
15. Luo S, Hu W. Score-Based Point Cloud Denoising. 2021 IEEE/CVF International Conference on Computer Vision (ICCV). Published online October 2021. doi: 10.1109/iccv48922.2021.00454
16. Huang H, Wu S, Gong M, et al. Edge-aware point set resampling. ACM Transactions on Graphics. 2013; 32(1): 1-12. doi: 10.1145/2421636.2421645
17. Mattei E, Castrodad A. Point Cloud Denoising via Moving RPCA. Computer Graphics Forum. 2016; 36(8): 123-137. doi: 10.1111/cgf.13068
18. Sun Y, Schaefer S, Wang W. Denoising point sets via L0 minimization. Computer Aided Geometric Design. 2015; 35-36: 2-15. doi: 10.1016/j.cagd.2015.03.011
19. Faisal Zaman and Ya Ping Wong and Boon Yian Ng: Density-based Denoising of Point Cloud, CoRR, http://arxiv.org/abs/1602.05312 (2021)
20. Qi, CR, Yi L, et al. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space, Proceedings of the 31st International Conference on Neural Information Processing Systems. 2017. pp. 5105-5114.
21. Wang Y, Sun Y, Liu Z, et al. Dynamic Graph CNN for Learning on Point Clouds. ACM Transactions on Graphics. 2019; 38(5): 1-12. doi: 10.1145/3326362
22. Duan C, Chen S, Kovacevic J. 3D Point Cloud Denoising via Deep Neural Network Based Local Surface Estimation. ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Published online May 2019. doi: 10.1109/icassp.2019.8682812
23. Rakotosaona M, La Barbera V, Guerrero P, et al. PointCleanNet: Learning to Denoise and Remove Outliers from Dense Point Clouds. Computer Graphics Forum. 2019; 39(1): 185-203. doi: 10.1111/cgf.13753
24. Hermosilla P, Ritschel T, Ropinski T. Total Denoising: Unsupervised Learning of 3D Point Cloud Cleaning, Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). 2019.
25. Pistilli F, Fracastoro G, Valsesia D, Magli E. Learning Graph-Convolutional Representationsfor Point Cloud Denoising, The European Conference on Computer Vision (ECCV). 2020. pp. 103-118.
26. Lin J, Zhang F. Loam_livox: A fast, robust, high-precision LiDAR odometry and mapping package for LiDARs of small FoV. Published online 2019. doi: 10.48550/ARXIV.1909.06700
27. Katz S, Tal A, Basri R. Direct visibility of point sets. ACM Transactions on Graphics. 2007; 26(3): 24. doi: 10.1145/1276377.1276407

28. Fischler MA, Bolles RC. Random sample consensus. Communications of the ACM. 1981; 24(6): 381-395. doi: 10.1145/358669.358692
29. Mariga L. pyRANSAC-3D. Available online: https://github.com/leomariga/pyRANSAC-3D (accessed on 2 March 2023).
30. Campello RJGB, Moulavi D, Zimek A, et al. Hierarchical Density Estimates for Data Clustering, Visualization, and Outlier Detection. ACM Transactions on Knowledge Discovery from Data. 2015; 10(1): 1-51. doi: 10.1145/2733381
31. LIVOX: livox ROS driver, 2021. Available online: https://github.com/libov-SDK/livox_ros_driver (accessed on 2 March 2023).