Collision Prediction using LiDAR Sensor in Digital Twin Environment

Minseung Kim and Jongchae Lee and Kanghyun Jo
Dept. of Electrical, Electronic and Computer Engineering
University of Ulsan
Ulsan, South Korea
{kmsoiio, rkdsudgjs78, acejo}@ulsan.ac.kr

***Abstract*—Various research utilizing LiDAR sensors is currently underway, with collision prediction being one of the key topics. However, obtaining data for researching collision prediction algorithms in the real world incurs challenges such as time, cost, and safety concerns. Therefore, this paper solves this problem on real data and virtual target objects in digital twin [1], which brings actual sensor information to Unity. When supervised learning is performed for object tracking and collision detection, there is a problem that cannot be dealt with if an object that was not present in the training dataset appears. So, this paper uses clustering(unsupervised learning). Ground removal and pre-measured points information are utilized for dynamic object classification using clustering. For tracking dynamic objects, the clusters from the previous frame and the current frame are compared. The closest clusters in terms of distance are classified as belonging to the same cluster. Then, the direction of movement is predicted through the position change of the classified cluster. Through this, the collision with the target object is predicted by firing a ray in the corresponding direction. The experiment is conducted on the accuracy of predicting collision and measuring the location error of reality and Unity in a fixed LiDAR environment indoors. As a result of the experiment, the average accuracy of 72.5% is shown at the 10cm Threshold for position error, and the accuracy of collision prediction was 96% for predicting non-collision. The prediction of the collision achieved an accuracy of 74%.**

***Index Terms*—LiDAR, Digital Twin, Object tracking**

I. INTRODUCTION

Recent research on collision prediction with LiDAR has become extensive. Constructing environments for collision prediction requires accident data. However, collision data is rare in the real world, and creating it directly entailed significant time and costs. An approach to address this is to construct virtual environments to obtain sensor data. This method can efficiently reduce time and costs. However, obtaining data about real-world situations such as sensor noises in virtual environments is challenging. Therefore, this study predicts collisions using information obtained from real LiDAR sensor data in a digital twin environment. The experimental setup is limited to an indoor environment with fixed LiDAR. Identifying objects is crucial for anticipating collisions. There are three methods for distinguishing objects: object detection, segmentation, and clustering. Accurate object detection is possible with deep learning methods like object detection or segmentation through training datasets. However, creating 3D datasets requires a significant time, and it is practically impossible to train every object. Accordingly, this paper employs unsupervised learning, specifically clustering, to differentiate obstacles. To predict collisions, the velocity and direction of objects are essential. Thus, the mean of each cluster is utilized to calculate the central position and employed for tracking [2].

This result was supported by "Regional Innovation Strategy (RIS)" through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(MOE)(2021RIS-003)

II. RELATED WORK

A. LiDAR Point clustering

Clustering methods for data include K-means [3], K-medians, and DBSCAN [4]. For K-means and K-medians, the process involves selecting the number of clusters and initializing centroids randomly. Subsequently, the euclidean distance between data points and centroids is calculated iteratively to cluster the data. However, this approach doesn't suit the paper environment, where the number of obstacles is undetermined, as the number of clusters needs to be predefined. Density-Based Spatial Clustering with Applications (DBSCAN) is an algorithm that performs clustering based on the density of data. The algorithm operates by setting the euclidean distance and minimum points for the data, then exploring internal points within the specified distance to find clusters. It functions by designating a cluster only if the number of internal points found within it exceeds the minimum points threshold. This method has the drawback that its performance can vary depending on the hyperparameters, such as euclidean distance and minimum points. However, by appropriately setting the min points parameter, it is possible to eliminate noise data from the LiDAR and perform clustering based on density regardless of the data format. Therefore, this method is applied in this paper. Experimentally, the euclidean distance is set to 20cm, and the minimum points are set to 15.

B. Aligning coordinate systems between Ouster and Unity

If the data output from Ouster is directly applied to Unity, the actual position of the point cloud can't be represented. This is because the Ouster data represents the height along the z-axis, while Unity uses a coordinate system where the y-axis represents height. So, dimensional transformation is necessary to map this information to match reality. In this paper, the point cloud from Ouster was stored in Unity with the Vector3(x, z,

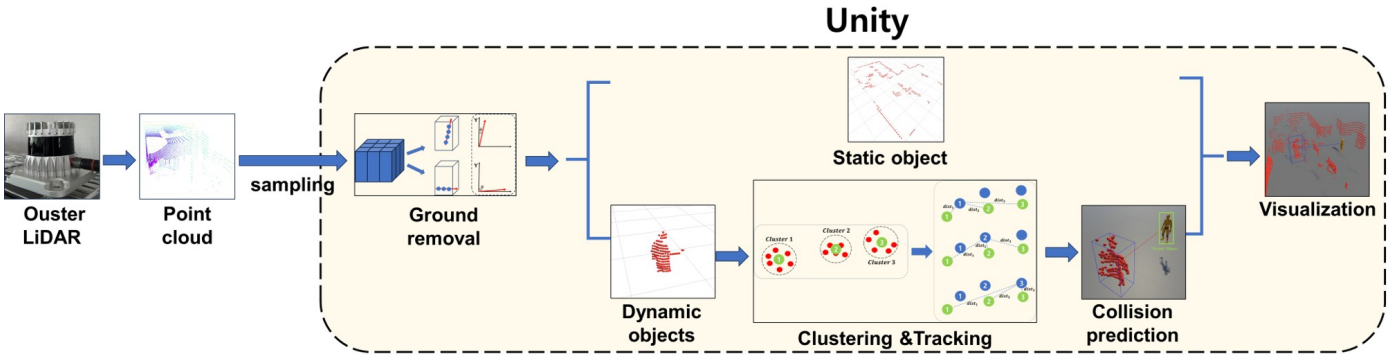

Fig. 1. This figure illustrates the overall algorithm of the paper, detailing the process of sampling the point cloud acquired through Ouster LiDAR and importing it into Unity. Additionally, **Ground removal** is conducted to enhance the accuracy of object clustering. After that, the **Static object** and the **Dynamic object** are distinguished. **Clustering** & **Tracking** and **Collision prediction** are performed only on the dynamic object. Finally, all results are visualized on the screen.

y) order to align the coordinate systems, and the dimensions in this paper are described based on Unity dimensions.

III. PROPOSED METHOD

A. Ground removal

When DBSCAN uses LiDAR point cloud data, a problem arises, as shown in Fig.2, where obstacles are recognized as a single cluster with the ground due to their attachment to the ground.

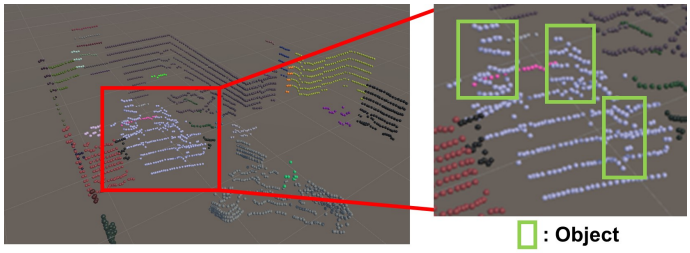

Fig. 2. The image shows improper clustering of objects due to the ground point cloud. Because the object within the green box is attached to the ground, without removing the ground, all objects in that area are classified into the same cluster.

To address this issue, Principal Component Analysis(PCA) was used to distinguish between the ground [5], [6] and obstacles. The LiDAR data was structured in three dimensions as shown in Eq.(1), and eigenvectors could be obtained through PCA. The equation followed Eq.(2,3,4).

$$X = \{x^{(1)}, x^{(2)}, \cdots, x^{(n)}\}, \quad x^{(n)} \in \mathbb{R}^3 \tag{1}$$

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x^{(i)} \tag{2}$$

$$\Sigma = \frac{1}{n}\sum_{i=1}^{n}(x^{(i)} - \bar{x})(x^{(i)} - \bar{x})^T \tag{3}$$

$$\Sigma V = V\Lambda \tag{4}$$

$$v_{\text{pc1}} = \text{argmax}_v(\Lambda) \tag{5}$$

In this paper, the eigenvector v_{pc1} with the largest eigenvalue, as shown in Eq.(5), is utilized. Because the ground has v_{pc1} aligned parallel to the xz-axis, while obstacles are perpendicular to the xz-axis. Therefore, it is possible to distinguish between the ground and objects based on the angle of v_{pc1}. PCA is applied to the data within each grid, which is set according to the LiDAR range. As for the grid structure, it is created as a tall cuboid along the y-axis, as depicted in Fig.3.(A). Through the tall structure along the y-axis, when obstacles are present, as shown in Fig. 3.(B), vectors closer to the vertical direction can be obtained. Conversely, in the case of the ground, as depicted in Fig. 3.(C), vectors closer to the horizontal direction can be obtained. The vectors obtained in this way, as depicted in Fig. 3.(D), are used to measure the angle concerning the Y-axis. If θ is greater than the threshold, the point is classified as ground. Otherwise, the points within that grid are deleted. To prevent the deletion of flat objects other than the ground, an algorithm is added to only delete points when the mean position of the points after applying PCA is lower than the mean position of all points. In this paper, the threshold for θ is set to 60 degrees, considering surfaces with slopes. Additionally, the height of each grid is set to the maximum height of the points, and the width is set to 10cm. The reason for 10cm is that the edge length of the Voxel grid sampling grid is set to 5cm, allowing for a minimum of 2 data points to be included. The applied results are depicted in Fig. 4. Compared to Fig. 4.(A) before ground removal, Fig. 4.(B) after using v_{pc1} for ground removal effectively demonstrates the removal of ground clutter.

B. Distinguish static and dynamic objects

LiDAR points do not measure the same location, but there is a slight change every frame. This variation causes fluctuations in the average values between the previous and current frames, leading to the problem of recognizing static objects as dynamic ones. Therefore, information about static objects measured in advance for one frame is stored, enabling the identification of

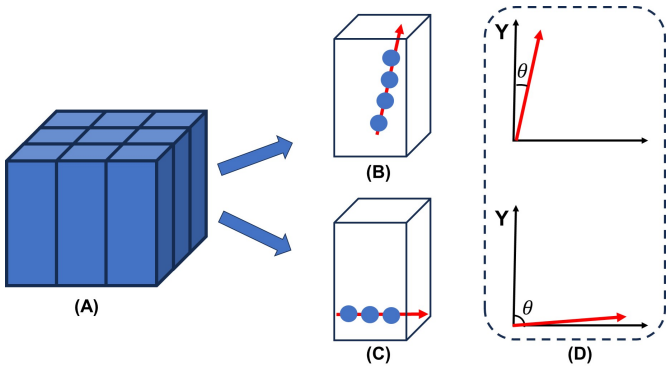

Fig. 3. The shape of the grid for calculating domain-specific PCA is represented in (A). By setting the height of the grid to the maximum value for the respective domain, the computation is reduced by distinguishing areas only along specific x and z axes. (B) and (C) refer to the eigenvectors obtained through PCA of the data within the grid. When there are obstacles in the area, pc1 is a vector perpendicular to the x and z axes, as shown in (B), whereas in the ground area, as shown in (C), a horizontal vector is formed. (D) represents a graph of pc1 with respect to the Y-axis, indicating the angle θ. In this paper, considering sloped ground, the surface is defined when θ is 60° or more.

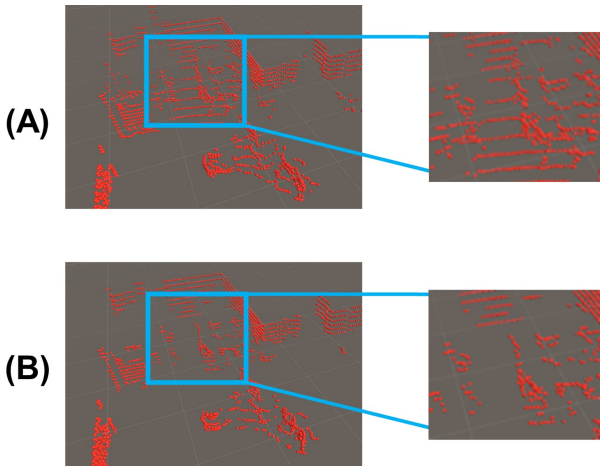

Fig. 4. (A) represents the ground point not removed. Clustering is not being properly performed due to the point cloud from the ground, as shown in Fig.(2). (B) shows that the point cloud from the ground is erased by the proposed algorithm. Compared to (A), the point cloud of the ground has been effectively removed in (B).

static object positions. Subsequently, clustering is conducted for the remaining areas excluding those positions. Due to the mentioned LiDAR points, an issue arises where static object division for points does not work properly when using 3D information for 1 frame. Therefore, when classifying static objects, the LiDAR points were projected in the x-z plane to reduce the dimension to two dimensions. Nevertheless, errors that occur are rounded off to two decimal places, and matching points are classified as dynamic objects.

C. Tracking method

As shown in Fig.5 DBSCAN is employed to generate clusters for each LiDAR point, and tracking is performed using the center points of the clusters. When tracking, the average position of the previous frame is compared with the average position of the current frame clusters, and the cluster with the smallest ***dist*** is set to the same cluster. At this time, the euclidean distance is used for the distance between clusters.

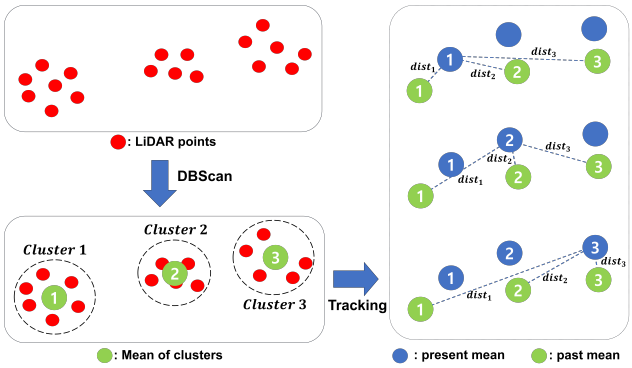

Fig. 5. This is a figure of the clustering and tracking method for LiDAR points. Clustering of the observed LiDAR points is performed using DBSCAN, and tracking is performed by matching the cluster with the closest ***dist*** to the current cluster using the average value of the clusters of the previous frame and the current frame.

D. Collision prediction

Because the moving direction and distance of an object can be obtained through tracking, collisions in the moving direction are predicted using the Unity raycast function. In the function of raycast, a ray can be fired in a specified direction and a collision with an object can be predicted by detecting the collision of the ray. When using raycast alone, collision detection is limited to a specific point and does not account for the area of a cluster. Therefore, a box-shaped ray is fired using the Unity function **Physics.BoxCast** to determine the target object in the direction of the cluster. The size of the box uses the min and max values for the x, y, and z axes of the cluster to create a box containing all points of the cluster, as shown in Fig.6. If the created box collides with the target object, a collision is predictable.

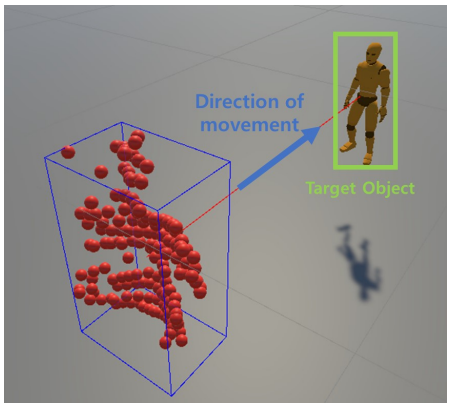

Fig. 6. This figure illustrates the process of creating a cuboid that matches the size of the dynamic object, firing a cuboid-shaped ray in the direction of movement, and predicting the possibility of collision based on whether the ray intersects with the target object.

IV. EXPERIMENT

A. Accuracy of dynamic object actual and virtual locations

To evaluate the alignment between the actual positions of dynamic objects and the positions of Unity-based clusters, tracks are constructed as depicted in Fig. 7. The formula for accuracy measurement is presented in Eq.(6,7). While moving along the track until reaching the destination, the distance($dist$) between the track and the mean position of clusters along the x and z axes(n_i) at each frame is compared with the error threshold[10cm(T_{10}), 5cm(T_5), 3cm(T_3)]. The average(Eq.(6)) is calculated by dividing the total number of points(N_{total}) falling within the error bounds(N_i). Each track is iterated 30 times, and the results are as follows in the table below.

$$\frac{\sum_{i=1}^{N_{\text{total}}} N_i}{N_{\text{total}}} \tag{6}$$

$$N_i = \begin{cases} 1 & \text{if dist}(n_i, \text{track}) < T_k, \quad k \in 3, 5, 10 \\ 0 & \text{else} \end{cases} \tag{7}$$

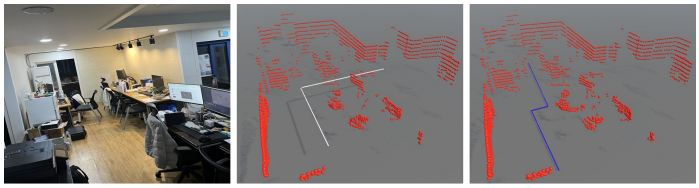

Fig. 7. This figure is constructed to measure the accuracy of the position of a dynamic object in the real environment and the measured position in Unity. Track A is marked with a white line, and Track B is marked with a blue line.

TABLE I
THE ACCURACY RESULT OF ACTUAL AND VIRTUAL LOCATION

	T_{10}	T_5	T_3
Track A	0.76	0.5	0.32
Track B	0.69	0.32	0.2

In the case of measurement results, it is confirmed that the accuracy of Track A tended to be higher than that of Track B. Because Track B is a track that is relatively farther than Track A.

B. Collision Prediction Accuracy

The accuracy measurement for collision detection is conducted with human subjects physically moving. The experiment aims to evaluate whether individuals can accurately predict the likelihood of collision when a target object is present in the direction of their movement and whether they perceive no collision when there is no target object in their path. The experiment involved varying the location of the target object, resulting in a total of 3,022 data points. The results are shown in the table below. The accuracy for **collision (O)** is measured at 73.5%, and the accuracy for **collision (X)** is measured at 96%. In the case of **collision (X)**, the measurement accuracy for not colliding because the moving direction of the observed object does not match the target object is 100%, but the measurement accuracy for not colliding when the object stops is 93.8%. The reason is that, as mentioned earlier, the measurement position of the LiDAR point changes slightly every frame and the change causes an error in which an unmoving target moves, resulting in a decrease in accuracy.

TABLE II
THE RESULT ACCURACY OF COLLISION PREDICTION

actual / predict	**collision (X)**	**collision (O)**
collision (X)	0.96	0.26
collision (O)	0.04	0.74

V. FUTURE WORK

Currently, the experiment is conducted in situations where the location of the target object is fixed, assuming an indoor environment where LiDAR is fixed. As a result of the experiment, it is confirmed that even in limited situations, accuracy is greatly reduced due to the position of the LiDAR point changing every frame. Therefore, we plan to first conduct filtering research to reduce measurement errors. Afterward, we plan to study how collision prediction for moving targets can be applied to mobile LiDAR.

REFERENCES

[1] A. Fuller, Z. Fan, C. Day, and C. Barlow, "Digital twin: Enabling technologies, challenges and open research," *IEEE Access*, vol. 8, pp. 108 952–108 971, 2020.

[2] T. Eppenberger, G. Cesari, M. Dymczyk, R. Siegwart, and R. Dubé, "Leveraging stereo-camera data for real-time dynamic obstacle detection and tracking," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 10 528–10 535.

[3] K. P. Sinaga and M.-S. Yang, "Unsupervised k-means clustering algorithm," *IEEE access*, vol. 8, pp. 80 716–80 727, 2020.

[4] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[5] H. Lim, M. Oh, and H. Myung, "Patchwork: Concentric zone-based region-wise ground segmentation with ground likelihood estimation using a 3d lidar sensor," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6458–6465, 2021.

[6] A. Nurunnabi, D. Belton, and G. West, "Diagnostics based principal component analysis for robust plane fitting in laser data," in *16th Int'l Conf. Computer and Information Technology*, 2014, pp. 484–489.