INF573

LiDAR-based object detection, tracking and prediction

2022

Virgile FOUSSEREAU, Jyh-Chwen KO

Contents

Introduction

LiDAR (Light Detection and Ranging) is a remote sensing technology that uses laser pulses to measure the distance and shape of objects. It has become an essential tool for various applications, such as autonomous driving, robotics, and surveying. In particular, LiDAR-based object detection, tracking, and prediction are crucial for ensuring the safety and efficiency of autonomous systems, especially in dynamic environments with moving objects.

One of the most important objects to detect and track in such scenarios is the pedestrian. Pedestrian safety is a major concern in the field of transportation, and the number of pedestrian fatalities has been increasing in recent years. In addition to the human cost, pedestrian accidents also cause significant economic losses, such as damage to vehicles, infrastructure, and property. Therefore, developing effective LiDAR-based methods for detecting, tracking, and predicting pedestrian behavior is crucial for improving the safety and efficiency of transportation systems.

However, detecting and tracking pedestrians using LiDAR data is challenging for several reasons. First, the range and resolution of LiDAR sensors are limited, which can make it difficult to detect small and distant objects, such as pedestrians. Second, the motion and shape of pedestrians can vary greatly, which can make it difficult to distinguish them from other objects, such as vehicles, bicycles, and trees. Third, the occlusion and clutter in the environment can also affect the performance of pedestrian detection algorithms, especially in dense and complex scenes.

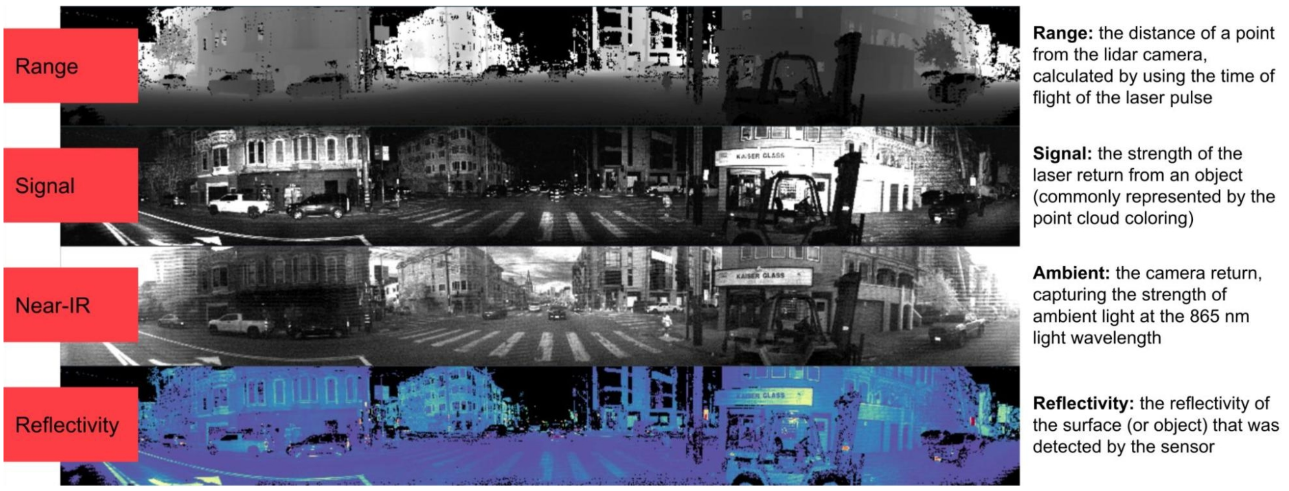
In this paper, we will present two methods to detect, track and predict pedestrian using only LiDAR data. We will work with unmoving LiDARs only, as in a smart infrastructure use case. We will discuss the challenges encountered to detect and cluster objects and the solutions we deployed.

Acknowledgements

We would like to thanks the Professor of INF573, Mathieu Brédif, for his lectures and his support. We also want to thanks teaching assistants Johannes Lutzeyer and Robin Magnet for their help.

1 Overview

To develop our methods, we use data recorded by an Ouster OS0-128 lidar sensor. It has a 90°field of view, a 10% range of 35 meters and 128 channels of resolution. It outputs four data layers for each pixel: Range, Signal, Near-IR, and Reflectivity. They are presented in figure 1.



Range: the distance of a point from the lidar camera, calculated by using the time of flight of the laser pulse

Signal: the strength of the laser return from an object (commonly represented by the point cloud coloring)

Ambient: the camera return, capturing the strength of ambient light at the 865 nm light wavelength

Reflectivity: the reflectivity of the surface (or object) that was detected by the sensor

Figure 1: LiDAR data layers

Based on these layers, and the corresponding 3D point cloud, we are aiming at detecting moving objects, especially pedestrians. We have developed to independent methods, one using Deep Learning and the other using only geometric tools. While the first one will detect only pedestrians, the second one will detect and track any moving object. The overall pipeline of each method is presented in figure 2.

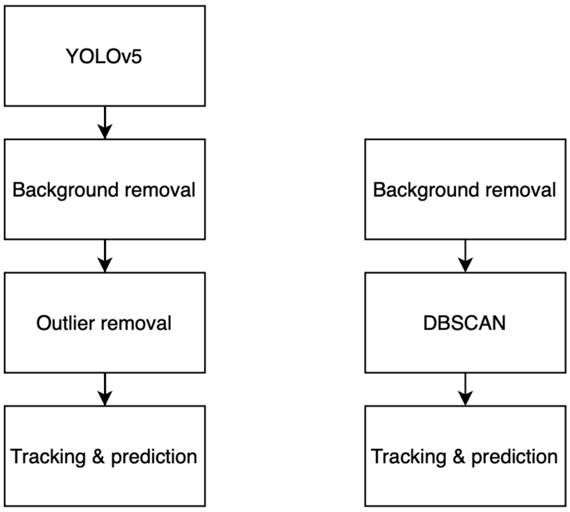


Figure 2: Pipeline of our two methods for detection, tracking and prediction

2 Deep Learning Approach

2.1 Object Detection with YOLOv5

Our first approach to the problem uses deep learning as a pre-processing step on the data. More specifically, we use a Convolutional Neural Network (CNN) based object detection algorithm that has been pre-trained on image data. We cannot use it directly on 3D data but it can be fined-tuned to work on one of the 2D layers presented in 1. For this project, we utilized the reflectivity layer to identify objects. We chose the reflectivity layer for object detection because it contains information about the inherent reflective property of objects. Where signal varies with range (objects further away return less light) and near-IR data varies with sunlight levels (not visible at night or indoors), reflectivity data is consistent across lighting conditions and range.

This fine-tuning has been done with for a social distance application by Ouster, to demonstrate a use case of their LiDARs [1]. The deep learning model used for this object detection is YOLOv5. YOLOv5 (You Only Look Once version 5) is a state-of-the-art object detection algorithm based on convolutional neural networks (CNNs) [2]. It is designed to be fast, accurate, and easy to use, making it suitable for a wide range of applications, such as real-time object detection in video streams and images.

The YOLOv5 algorithm follows the general approach of the YOLO (You Only Look Once) family of object detection algorithms, which uses a single CNN to predict the bounding boxes and class probabilities of objects in an image. The main advantage of YOLOv5 over its predecessors is its improved performance, thanks to several innovations and optimizations in the network architecture and training procedure.

Some of the key features of YOLOv5 include the following:

- **Multi-scale training and testing**: YOLOv5 uses multiple scales for training and testing, which allows it to handle objects of different sizes and shapes more effectively.
- **Anchor boxes**: YOLOv5 uses anchor boxes, which are predefined bounding box shapes, to help the network learn to predict the shape of objects more accurately.
- **Feature pyramid networks**: YOLOv5 uses feature pyramid networks (FPNs) to combine features from different scales and resolutions, which improves the performance of the network on small and distant objects.
- **Cross-scale training**: YOLOv5 uses cross-scale training, which allows the network to learn from features at different scales and resolutions, further improving its ability to detect objects of various sizes and shapes.

Overall, YOLOv5 is a powerful and efficient object detection algorithm that has demonstrated impressive results on various benchmarks and challenges. It is widely used in a variety of applications, including autonomous driving, surveillance, and robotics. With very little retraining for LiDAR data (620 images used), it has shown good detection capabilities on our sample data. However, it has difficulties for pedestrian too far away from the sensor. An example of detection is provided in figure 3.

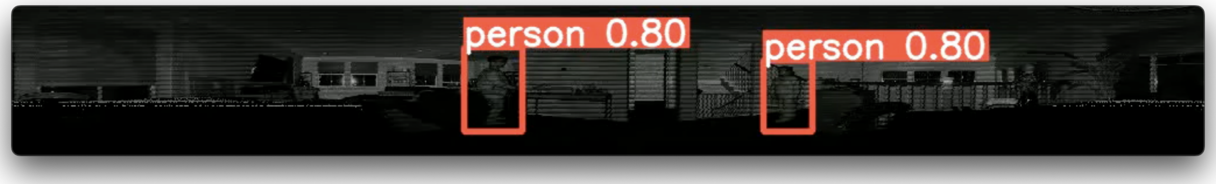
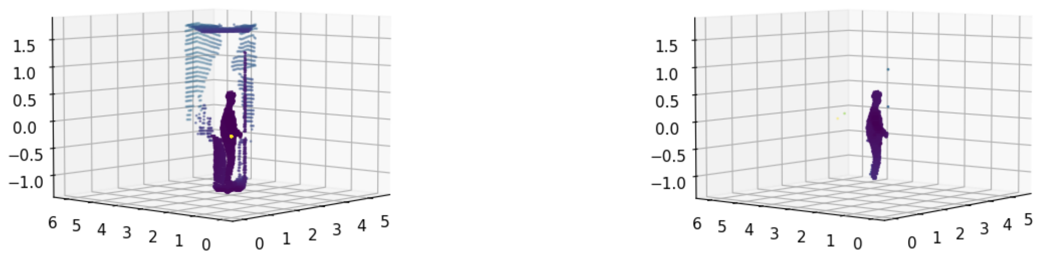


Figure 3: YOLOv5 detection

YOLOv5 provide bounding boxes as well as a confidence level. However, it does not provide a mask of the person. Therefore we cannot distinguish the background from the actual object detected. To tackle this issue, we developed a background removal method.

2.2 Background Removal

Given a bounding box provided by YOLOv5 on the reflectivity layer, we can project the corresponding points in the 3D point cloud. The result is shown in 4a.



(a) Before Background Removal

(b) After Background Removal

Figure 4: 3D point cloud corresponding to a detected bounding box, before and after background removal

To remove the points corresponding to the background, we use a maximum range filter. We compute the maximum range for each point over all the duration of the recording. As the pedestrian are moving, the result is in fact a representation of the background. However, some pitfalls need to be avoided. Indeed, a specific frame of the data can contains aberrant values for a very few points, that we do not want to keep as the value of the background. To avoid this, we can use the 99% percentile of ranges over all frames for a given point instead of the maximum range. Also, as there is a few centimeters of noise between frames, we consider that all points close enough to the background range are part of the background. For our data, we used 50cm as a distance threshold. The isolated background is shown in the reflectivity layer in figure 5, with a given frame for comparison.

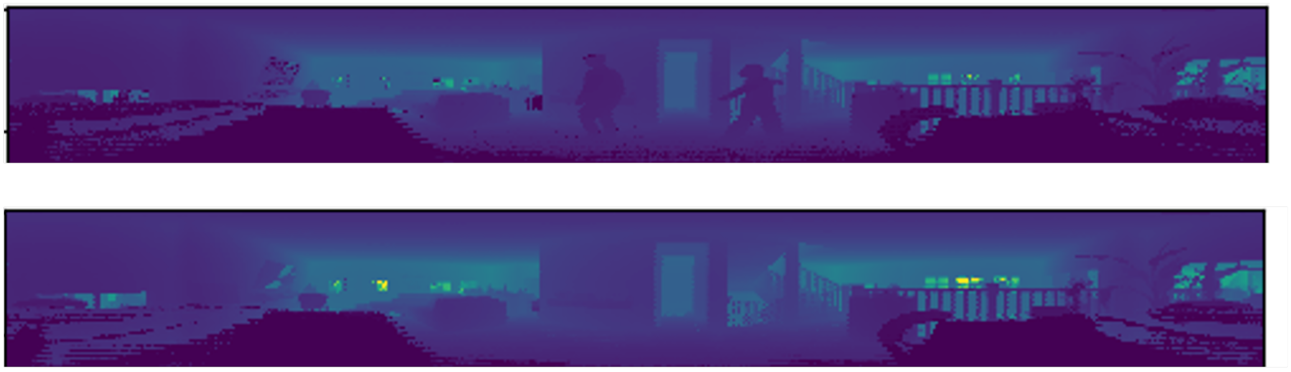


Figure 5: Background isolation

After sucessfuly removing the background, we have all the points supposedly corresponding to the person. However, as seen in 4b, a few outliers are still present. Therefore, we proceed with an outlier removal step.

2.3 Outlier Removal

Given the result obtained after the background removal, we only need to filter out a few outliers. Several methods can be used for this purpose. We developed a Kth-Distance algorithm to do this task. Given a point cloud, we compute the mean distance to the Kth nearest neighbor of each point. Then, we filter out every point for which the distance to its Kth nearest neighbor is *too big* compared to this mean. In our method, we used K=2 and implemented *too big* as superior than 110% of the mean. The results are shown in 6. Removed outliers are shown in red. Finally, we obtained a final point cloud corresponding to the detected person, that we will use in section 4.

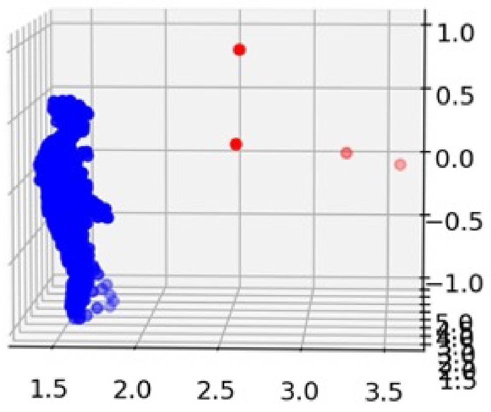


Figure 6: Outliers removal

3 Geometry-based Approach

In our previous approach, YOLOv5 fine-tuning has been made on a low amount on data and may not generalized well. To have an alternative approach, we decided to develop a geometry-based method that could provide similar results. Instead of having bounding boxes for detected pedestrian, we start directly with the entire scene. To start with, we reuse entirely our background removal step described in section 2.2. The result, shown in figure 7, contains a lot noise. Our outlier removal technique developed for the previous approach would not work in this case. Also, as the scene is taken as whole, we do not distinguish different objects at this step. We need to use a clustering algorithm. To tackle both the filtering and clustering, we developed a geometrical method described in 3.1.

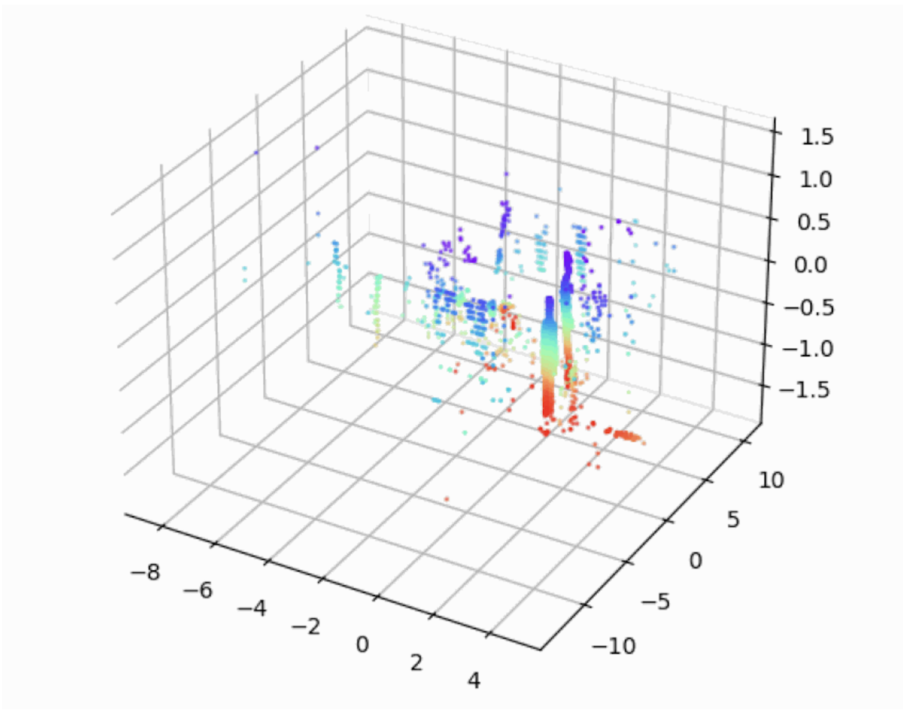


Figure 7: Scene after background removal

3.1 Filtering and Clustering

As we can see on figure 7, point clouds corresponding to pedestrians are quite dense while noise points, while numerous, are much more sparse. To both cluster and filter the points, we apply the following simple algorithm:

- Start with each point being its own cluster
- For each point, merge its cluster with the clusters of points closer than a distance d
- Delete clusters with less than N points

To implement this algorithm, we used a union-find data structure. For our application, we took d = 8 cm and N = 500 points. Results are shown in figure 8. The algorithm is able to both filter noise and cluster points to distinguish different objects. This algorithm is in fact equivalent to a DBSCAN (density-based spatial clustering of applications with noise) used with min_points = 2 and a distance d = 8 cm [3]. An optimized implementation of DBSCAN is available in Python in the scikit-learn library. For better performance, we use this implementation in our code.

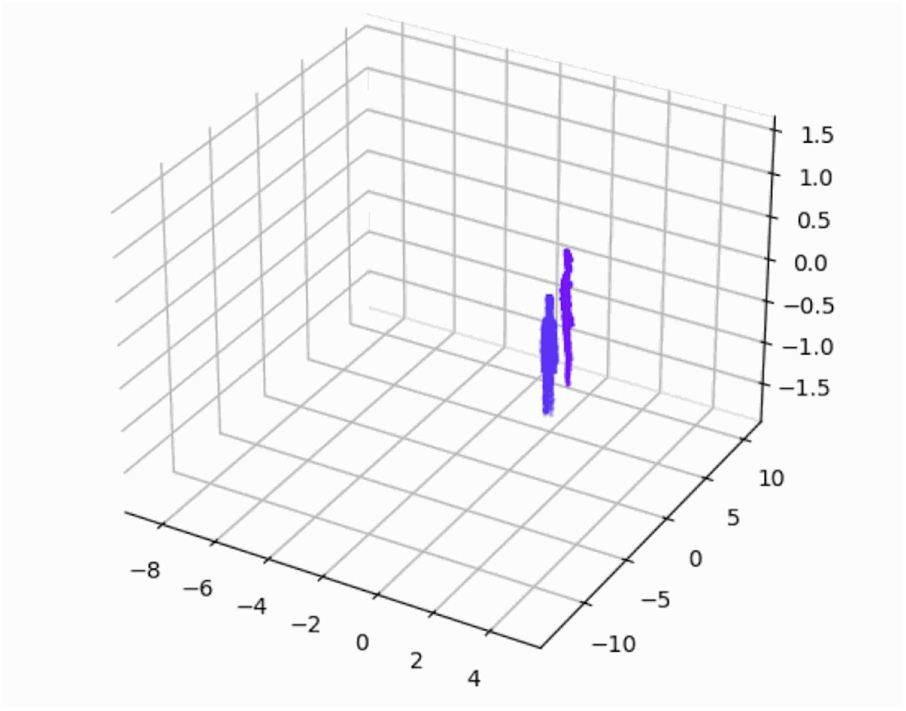


Figure 8: Scene after filtering and clustering

4 Tracking and Prediction

The two different methods, learning or geometry-based, that we have described so far give us point clouds of independent moving object. The learning-based method also ensure us that these objects are pedestrians. We can then proceed to track and make prediction for these objects. As our LiDAR sensor has a high frame rate generation of 10 fps, the position of a given object does not change a lot between two frames. Therefore we can track objects simply by picking the closest clusters between each frame. From a tracked cluster of points, we can for example compute the trajectory of the center of mass. Then, based on this trajectory, we can extrapolate the future movement of the center of mass. In figure 9, we provide a top view of the scene with red arrows to visualize the predicted movement.

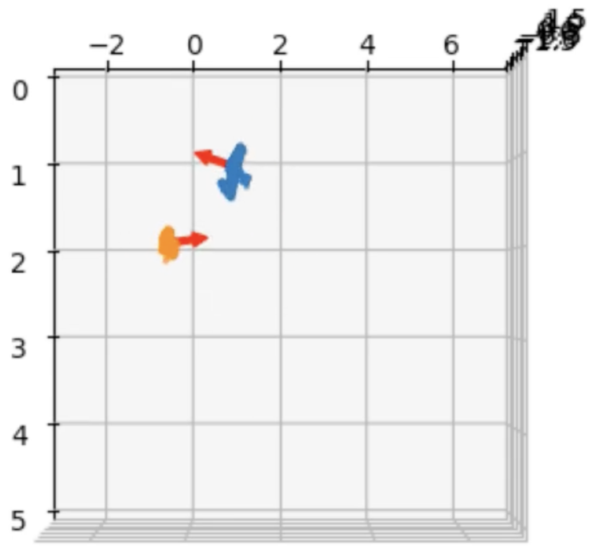


Figure 9: Top view of predicted movements

Given a prediction at the time t, we can use the real position of the object at time t+1 to evaluate the correctness of the prediction. First, we evaluate the orientation of our prediction: are we predicting a movement in the correct direction? We plot the precision of our predictions based on the tolerated angle error in figure 10. For example, in 82% of cases, our prediction is within 30°of the correct orientation. We can see that both of our methods have very similar results. To give a reference, on a suggestion of Robin Magnet, we provide the results of a *naive* approach where we do not remove the background. For an equal tolerated error of 30°, the naive approach only achieve a precision of 22%.

These results are obtained on a record where pedestrian are walking erratically in a room and the precision achieved would probably improve in a real-life situation. To further improve the predictions, a model of pedestrian behavior could be used.

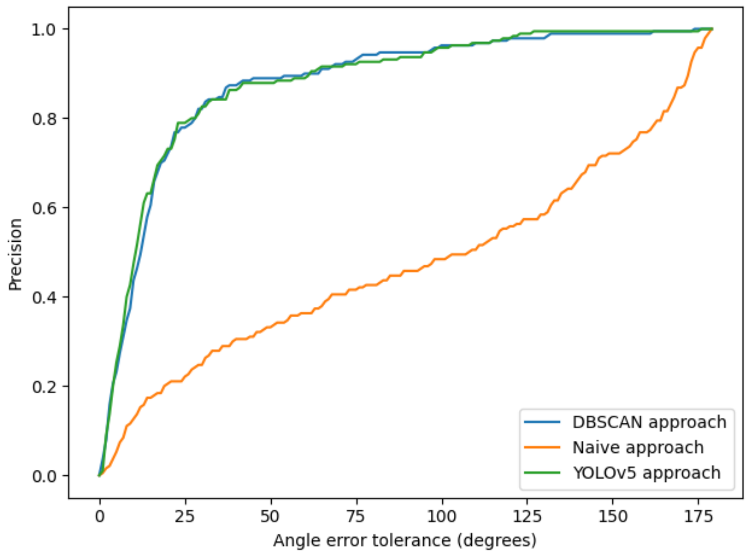


Figure 10: Precision curve on orientation prediction

We plot a similar curve for the precision over the distance covered during the movement, in figure 11. To give a distance scale reference, the mean distance covered between two frames is plotted in red. As we can see, both our methods have a precision of more or less 80% for a tolerated error of 20mm, which is a bit less than half the mean distance covered. For the same error tolerance, the naive approach achieve 37% precision. We can also see that the learning-based method has a slight advantage on this metric compared to the geometric approach.

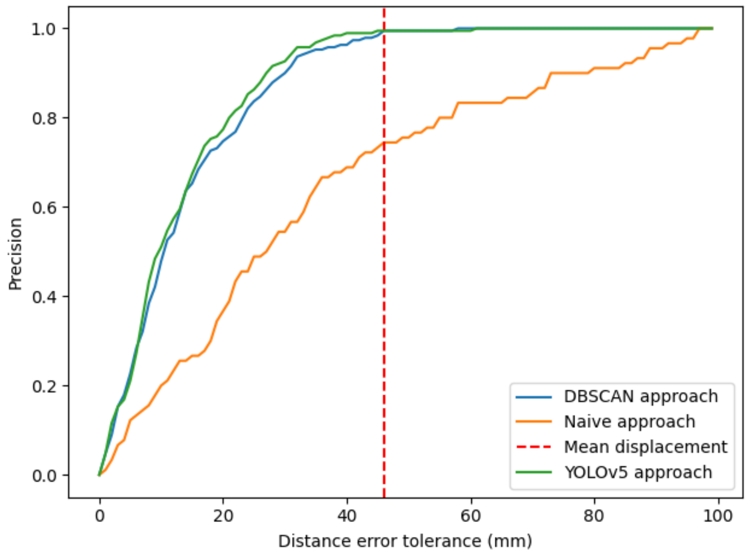


Figure 11: Precision curve on distance prediction

Conclusion and perpestives

The use of LiDAR technology in detecting and tracking pedestrians is crucial for improving the safety and efficiency of transportation systems. However, detecting and tracking pedestrians using LiDAR data can be challenging due to the limitations of LiDAR sensors and the complex and dynamic nature of pedestrian behavior. In this paper, we present two methods for detecting, tracking, and predicting pedestrian behavior using only LiDAR data. The first method relies on deep learning to detect pedestrians, while the second method uses geometric tools to detect and track any moving object. We discuss the challenges and solutions to detecting and clustering objects using these methods and evaluate their performance. Our methods demonstrate the feasibility of detecting, tracking and predicting pedestrian movements but could be improved in several ways. A specific improvement to the learning-based approach would be to train our YOLOv5 model on much more LiDAR data to improve its robustness. For the geometry-based method, a classification step could be added after clustering to determine the detected objects (pedestrians, cars...).

One possible improvement to both of the methods presented in this paper would be to incorporate additional data sources, such as camera or radar data, to improve the accuracy and robustness of pedestrian detection and tracking. This could help overcome the limitations of LiDAR sensors and provide a more comprehensive view of the environment. Another possible improvement would be to incorporate more advanced deep learning techniques, such as multi-object tracking and object-level fusion, to better handle occlusion and clutter in dense and complex scenes. Additionally, the integration of more sophisticated prediction models, such as those based on machine learning or physics-based simulations, could improve the ability to anticipate and react to pedestrian behavior in real-time.

References

[1] Fisher Shi. *Object Detection and Tracking using Deep Learning and Ouster Python SDK*. `https://github.com/fisher-jianyu-shi/yolov5_Ouster-lidar-example`. Mar. 2022.

[2] Glenn Jocher et al. *ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements*. `https://doi.org/10.5281/zenodo.4154370`. Oct. 2020.

[3] Martin Ester et al. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD'96. Portland, Oregon: AAAI Press, 1996, pp. 226–231.