LiDAR-based Pedestrian 3D Detection

COSE416 (2024 Fall) HW1

2022320017 Youjin Kim

https://github.com/zinzinyou/20242R0136COSE41600

Method

- Goal : Detecting pedestrians from 3D point cloud data
- Non-deep learning approach : based on traditional point cloud processing and clustering algorithms

1. Data Loading & Preprocessing
   - -Voxel Downsampling
   - -Radius Outlier Removal
   - -RANSAC
2. Calculate Bounding Box (every 5 frames)
   I. Moving Object Detection : KDTree Nearest-Neighbor Search
   II. DBSCAN Clustering
   III. Bounding Box Generation with Filtering

- Performing the computation every 5 frames rather than every frame offers several advantages.
  - Reduces total processing time, allowing the system to maintain responsiveness near real-time
  - Scalable for longer sequences or higher frame rates
  - Allows for detecting meaningful movements without processing redundant changes, since pedestrians do not exhibit drastic positional changes in consecutive frames

Method

Here's a detailed explanation of the three stages involved in generating bounding boxes.

I. Moving Object Detection

- KDTree is used to find the nearest neighbors between frames.
- Points with a displacement greater than the threshold (a hyperparameter) are considered as moving.
- On average, only 3% of the total points is moving. By filtering out stationary points before clustering with DBSCAN, computational efficiency is improved.

II. DBSCAN Clustering

- Clustering is applied exclusively to the detected moving points.
- To avoid missing pedestrian clusters, the eps value is increased, and the min_points threshold is reduced

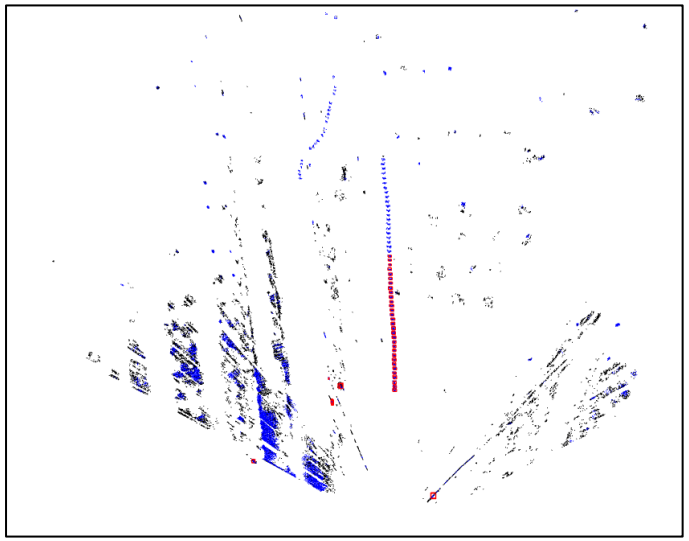
III. Bounding Box Generation with Filtering

- Relaxing the clustering thresholds can result in additional candidate bounding boxes.
  To address this, width and depth filtering are added alongside height filtering.
- These filters (min/max_width/height/depth) ensure bounding boxes are appropriately shaped for a human figure.
- The minimum height threshold is reduced to account for scenarios involving crawling or duck-walking.

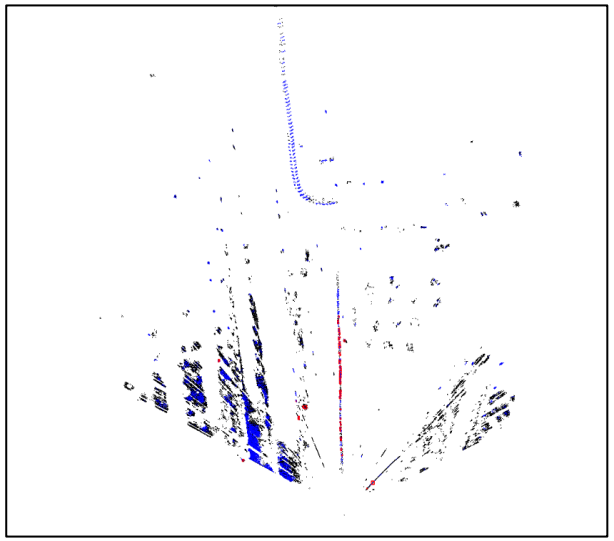
Results : Computation Time

- Total time : includes preprocessing, generating bounding boxes
  - Scenario 1 [288 files] : 8s
  - Scenario 2 [549 files] : 17s
  - Scenario 3 [1251 files] : 41s
  - Scenario 4 [364 files] : 17s
  - Scenario 5 [577 files] : 26s
  - Scenario 6 [774 files] : 35s
  - Scenario 7 [443 files] : 19s
- Total time : includes preprocessing, generating bounding boxes, and saving frames
  - Scenario 1 [288 files] : 1m 37s
  - Scenario 2 [549 files] : 3m 04s
  - Scenario 3 [1251 files] : 7m 03s
  - Scenario 4 [364 files] : 2m 08s
  - Scenario 5 [577 files] : 3m 22s
  - Scenario 6 [774 files] : 4m 40s
  - Scenario 7 [443 files] : 2m 36s

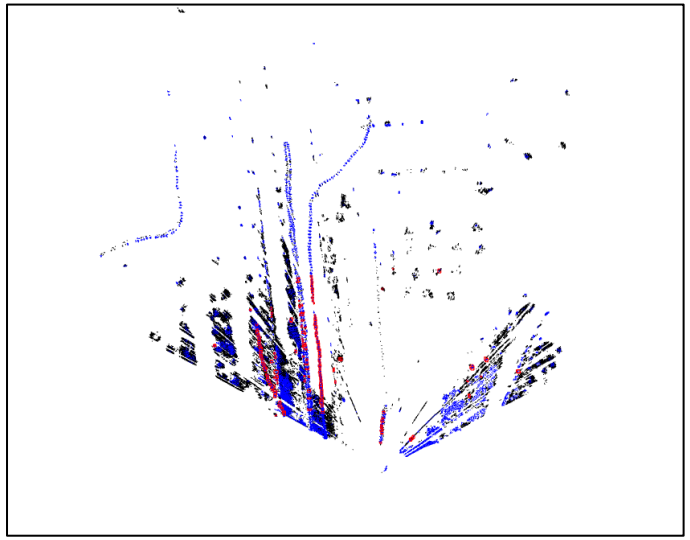
Results：Bounding Box on High-Illumination Scene



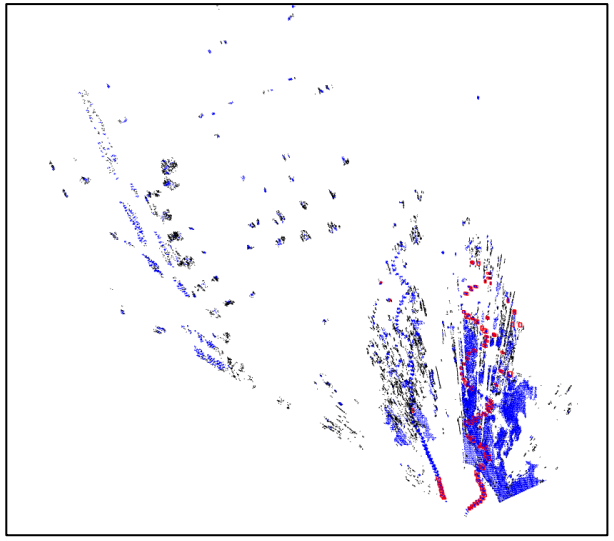
01_straight_walk



02_straight_duck_walk



03_straight_crawl



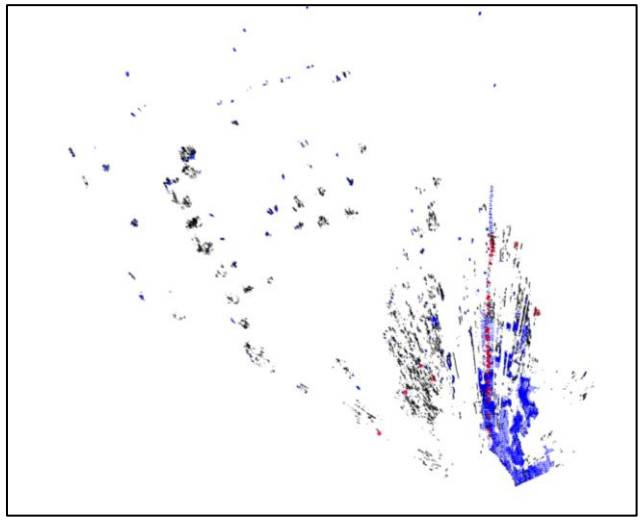
04_zigzag_walk

Since the scenario shares the coordinate system (LiDAR sensor is fixed), we can visualize the accumulated bounding boxes on the scene.

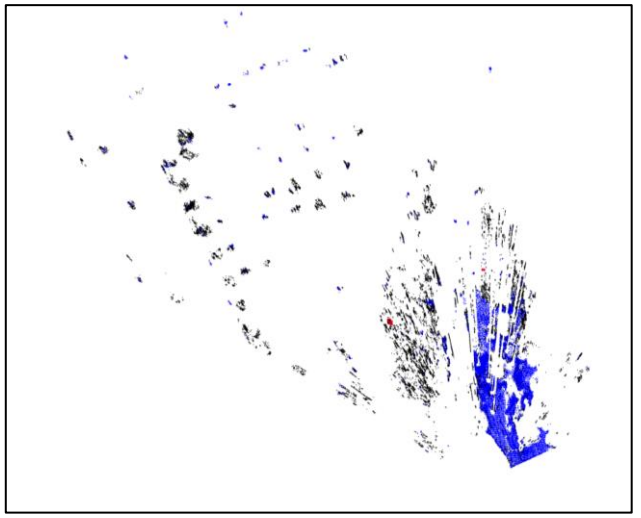
- Black points indicate noise
- Blue points indicate clustered points
- Red boxes are the bounding box.

By tracking the red boxes, we can find out the overall trajectory of the pedestrian.

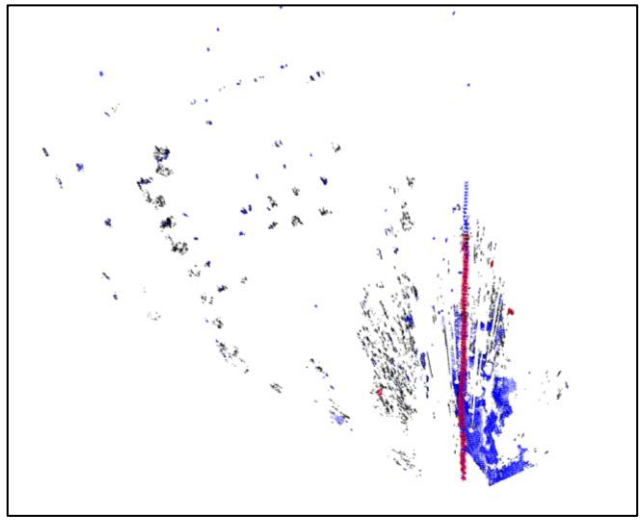
Results : Bounding Box on Low-Illumination Scene



05_straight_duck_walk



06_straight_crawl



07_straight_walk

Result Analysis

- The proposed method works better on high-illumination scene. Also, walking scenario works best, followed by zig-zag, duck_walk, and crawl.
- Lighting conditions : In low illumination, reduced contrast and increased noise leads to difficulties in detecting motion and clustering, particularly for low-height movements like crawling.
- Pedestrian height and motion patterns : Lower pedestrian heights (e.g. Duck_walk, Crawl) are challenging for the algorithm, as the bounding box generation depends on thresholds for height, width, and depth.

References

- https://scikit-learn.org/1.5/modules/generated/sklearn.cluster.DBSCAN.html
- https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.KDTree.html
- https://www.open3d.org/docs/release/tutorial/geometry/pointcloud.html