1/5-Second Point Cloud Pedestrian Detection on CPU

2024KU0301 이영찬

Introduction

1.과제의 목적

3D 포인트 클라우드 데이터(PCD)를 이용해 보행자를 효과적으로 감지하는 알고리즘을 설계하고 구현.

2.기존 PCD 기반 보행자 감지 알고리즘의 문제점

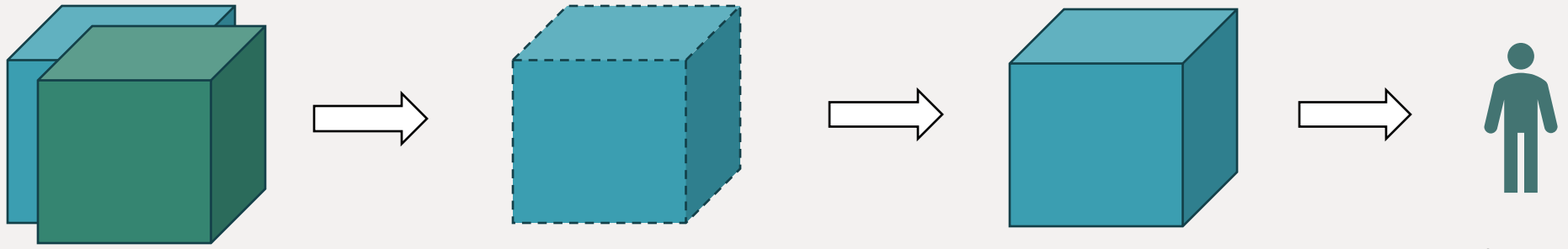
1. 처리 속도 문제:
   PCD 데이터는 대량의 3차원 좌표 정보를 포함하며, 이를 실시간으로 처리하는 데 많은 시간이 소요됨.
2. 고성능 하드웨어 요구:
   복잡한 3D 연산 및 딥러닝 기반 모델들은 강력한 GPU 및 연산 자원을 필요로 함.
3. 예시:
    1. VoxelNet: 3D 포인트 클라우드를 격자 형태로 변환하여 처리하는 방식은 높은 정확도를 제공하지만, 연산 비용이 매우 큼.
    2. PointNet: 포인트 클라우드 데이터의 특징을 추출하는 데 효과적이지만, 대량의 데이터 처리 시 성능 저하가 발생.

Methodology

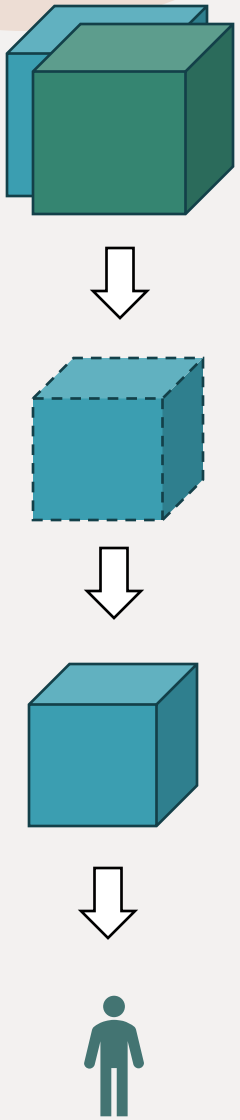
해결 방안

•배경제거:
PCD 데이터에서 불필요한 포인트(예: 고정된 구조물이나 차량 등)를 제거하여 관심 영역(보행자)에 집중.

배경을 제거 하는 방법은 이전 프레임과 비교를 통해 제거함



Methodology



1. 3D PCD Down Sampling

입력된 PCD와 첫번째 프레임의 PCD(혹은 100프레임 뒤의 PCD)의 Voxel크기를 0.2로 다운 샘플링 함.

2. Background remove

두개의 PCD를 겹쳐 가까운 점을 제거해 객체일 가능성이 있는 점만 남김. (cKDTree사용)

남긴 점에 대해서 노이즈 제거를 실행함.

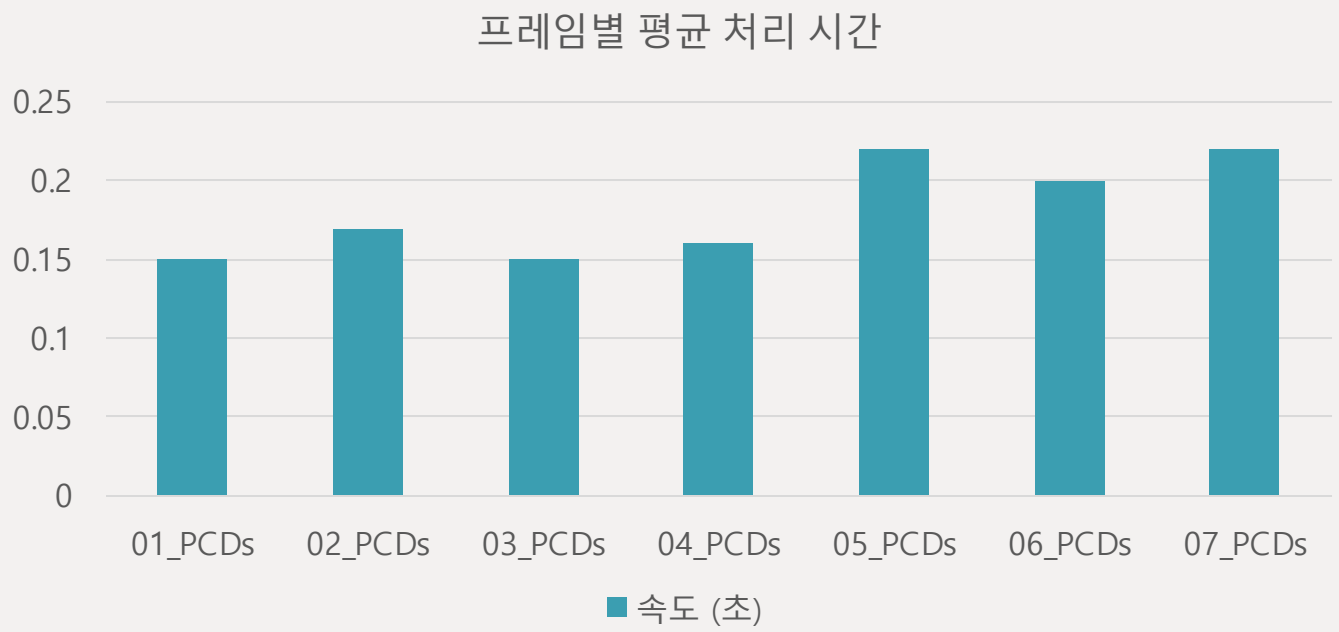
3. 3D clustering

남겨진 점에 대해서 3D 클러스터링을 진행해, 보행자를 검출함. (DBSCAN 사용)

Result

1.속도

위 표는 주어진 7가지 상황 별 각 PCD프레임 처리 시간을 나타낸 것임. 평균적으로 0.18초 걸린 것을 알 수 있음.

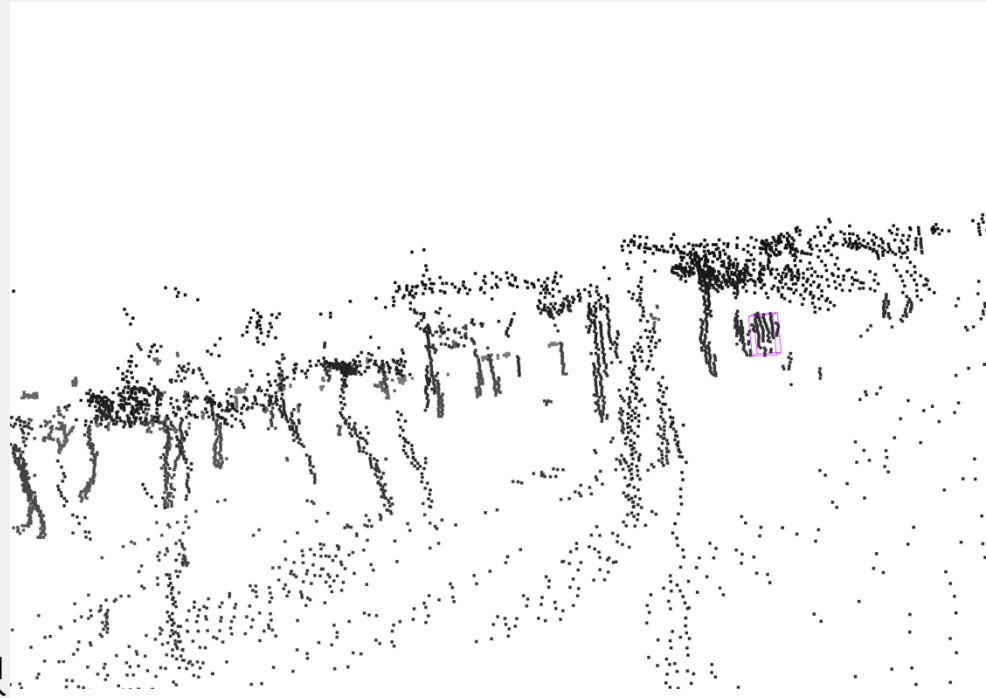


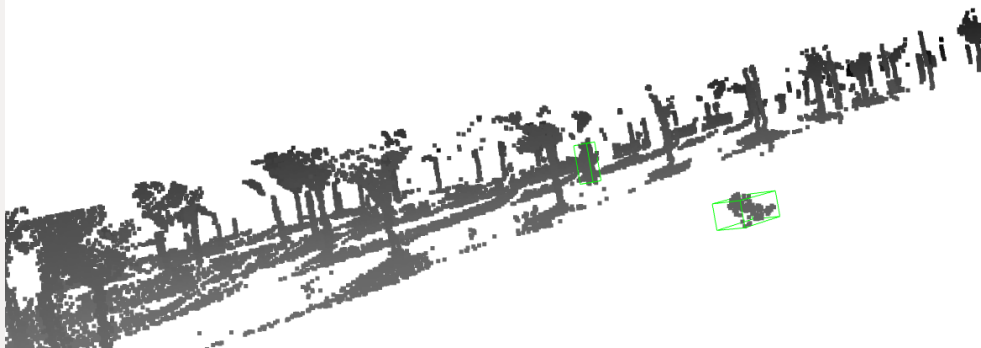
Result

2. 정확도

error rate = 0.02%로 낮게 나타났음(2)

또한, 엎드려 있는 사람이나, 나무 밑에 가려져 있는 사람도 판별할 수 있





2) error rate = (선택된 객체가 있는 PCD의 수) / (전체 PCD 수)
로 계산된 것으로 실제 값과는 조금 다를 수 있음.

Reference

1. 참고 문헌

1) Open3D. (2024, November 26). How to delete duplicate or corresponding point from 2 point cloud files? [Discussion post]. GitHub. https://github.com/isl-org/Open3D/discussions/6014

2. test 컴퓨터 사양

MacBook Pro 14 (2021) / m1pro(18C , 14G) / 16GB