Change Detection

Using Point Clouds

KIOUSIS PANAGIOTIS

University of Patras,
Visualization and Virtual Reality Group (VVR)

1. Introduction

Change detection is a key task in computer vision and remote sensing, used for monitoring structural modifications, urban growth and more. The goal of this project was to design a workflow for detecting and analyzing changes between two registered point clouds acquired at different time steps.

Given two point sets (reconstructions at time *t1* and *t2*), the task was to detect regions of change, represent them in a spatially coherent way, and apply clustering for object-level interpretation.

2. Purpose and Objectives

The main objectives of the project were:

1. **To visualize and highlight local differences** between two point cloud datasets.
2. **To design a more structured detection pipeline** that represents mismatches at cell-level (instead of individual points) using spatial subdivision.
3. **To extract and cluster change regions** for a higher-level understanding of modifications between time steps.

3. Data Preprocessing

- The two point clouds (recon1.txt and recon2.txt) were already aligned (registration was performed).
- Both datasets contained the spatial coordinates of reconstructed points.
- For analysis, distances between corresponding points were estimated using **KD-Trees** to efficiently compute nearest neighbors.

The methodology developed in this project was applied to both **2D and 3D point cloud datasets**. The following sections, present the workflow and results for the **2D case**. The extension to **3D datasets** and its results will be addressed later.

The 2d case:

The datasets correspond to reconstructions generated via SLAM from 2D laser scans collected inside a warehouse environment. Two codes where developed.

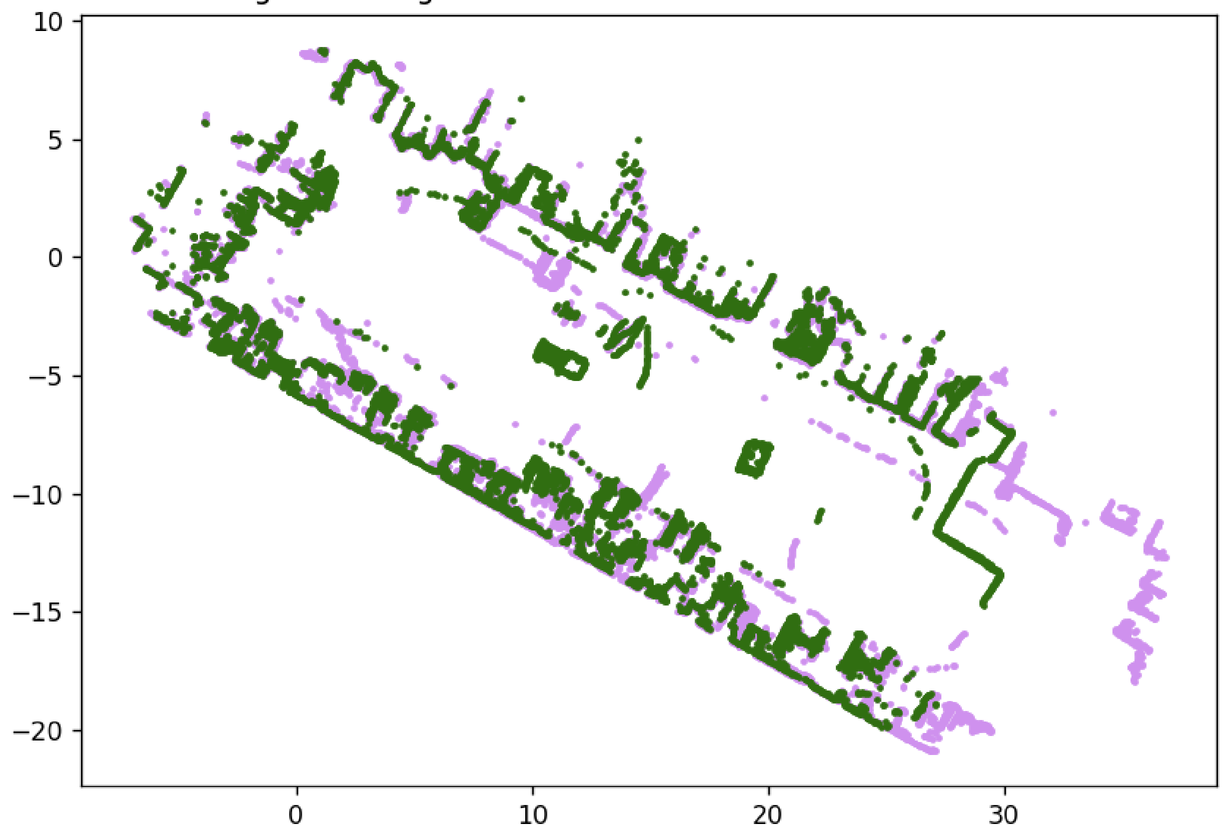
i) Visualization of Point-Level Changes (Code 1)

The first code was a simple proof-of-concept script to **showcase the change detection problem visually**.

- **Nearest-neighbor search:** KD-Trees were used to find closest points between the two datasets.
- **Thresholding:** If the distance between corresponding points exceeded a threshold (0.3 units), the point was flagged as a mismatch.

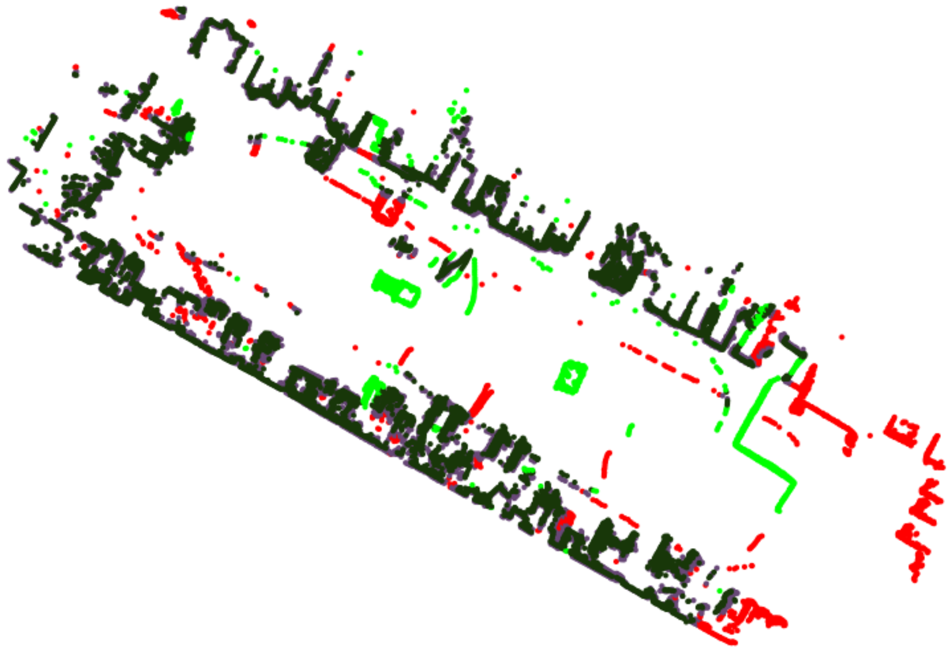
Visualization:

Figure 1. Original Reconstructions of the Two Point Sets



- Original point sets were plotted with distinct colors.

Figure 2. Point-Level Differences Between Reconstructions (Red = Missing, Green = New)



- Changes were highlighted (red = points missing from Set 2, green = points newly appearing in Set 2).

This step served as an **exploratory tool** to verify that local differences were indeed present.

ii) Quadtree-Based Change Detection (Code 2)

The second code was the main algorithmic contribution.

a) Spatial Subdivision with Quadtrees

(Why Quadtrees Were Chosen)

To achieve a structured representation of changes, a **quadtree data structure** was employed to hierarchically subdivide the 2D point space into rectangular cells. Quadtrees were chosen due to their ability to efficiently represent spatial data at multiple resolutions, enabling higher resolution in dense regions and lower resolution in sparse areas. This approach reduces computational complexity compared to point-wise analysis, while also mitigating the impact of small misalignments or outliers.

(how were the changes computed)

Within each cell, **mismatch values**—computed as average nearest-neighbor distances—were aggregated, allowing change detection at a **cell-level granularity**. This hierarchical aggregation not only smooths out noise inherent to individual points but also facilitates scalability to larger datasets, including high-resolution 2D scans and eventual 3D extensions.

b) The 4 Quadtrees of the pipeline

To systematically integrate and analyze changes between the two point cloud datasets, four quadtree structures were defined:

1. **Quadtree 1:** Represents Set 1 (time *t1*).
2. **Quadtree 2:** Represents Set 2 (time *t2*).
3. **Quadtree 3:** An updated reconstruction of Set 1, incorporating points from Set 2 where changes were detected.
4. **Quadtree 4:** Represents the intersection of unchanged regions between Set 1 and Set 2.

The implementation in code followed a **cell-level comparison pipeline**:

First, both point sets were inserted into their respective quadtrees (Quadtree 1 and Quadtree 2), with **each cell storing aggregated mismatch values** computed from nearest-neighbor distances.

Using a **lookup dictionary keyed by cell boundaries**, cells from Quadtree 1 were compared with corresponding cells in Quadtree 2.

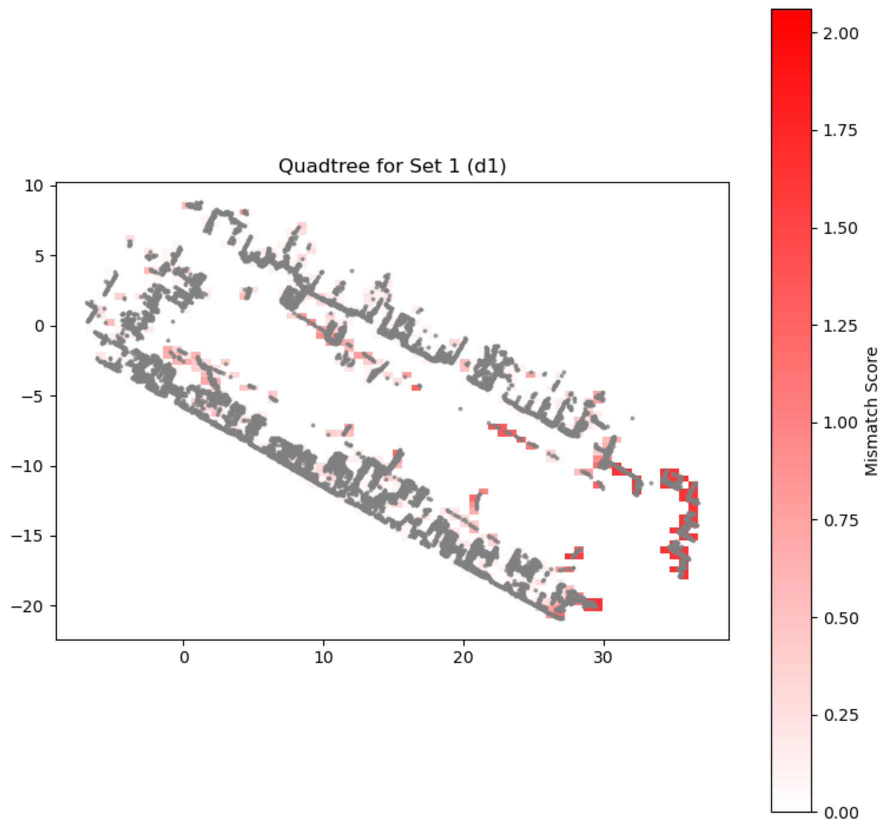
- If both cells had mismatch values below a defined threshold, points were considered unchanged and inserted into both Quadtree 3 and Quadtree 4.
- If a cell in Quadtree 2 indicated significant change relative to Quadtree 1, its points were inserted into Quadtree 3, updating the reconstruction of Set 1 to reflect detected modifications.

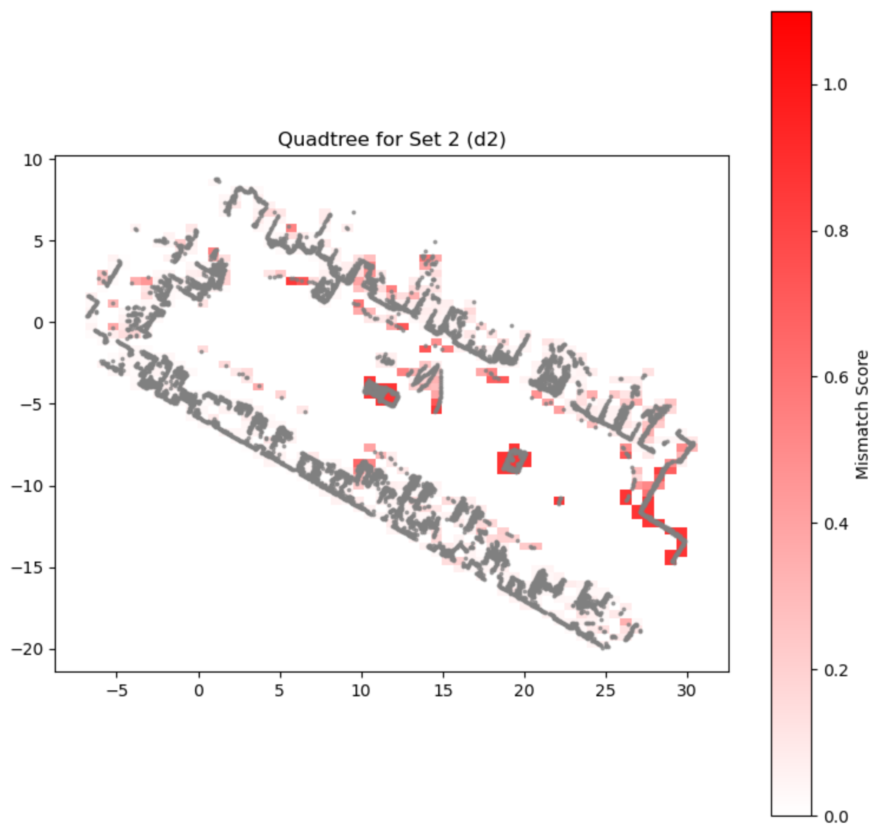
Finally, cells in Quadtree 2 without corresponding matches in Quadtree 1 were also incorporated into Quadtree 3, ensuring that all new features appearing in Set 2 were captured.

Results:

Heatmap-like quadtree representations show mismatch intensity per cell.

The 1st image shows the data from day 1. The 2nd image shows the data from day 2.

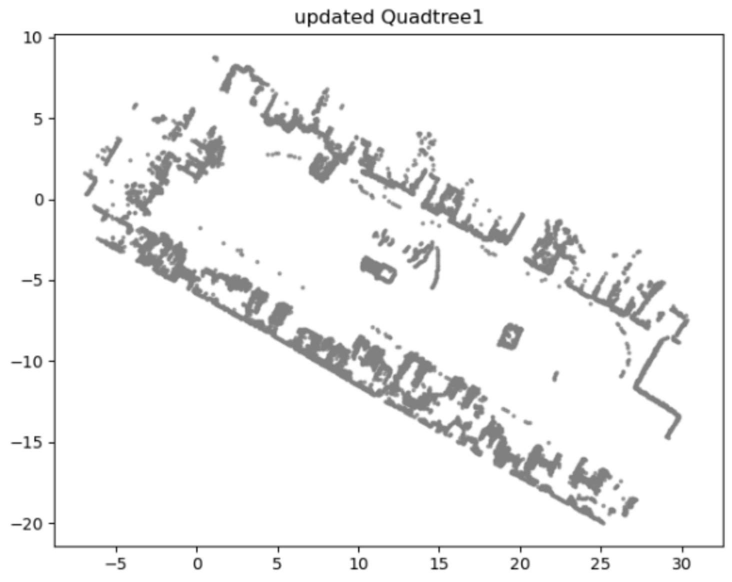




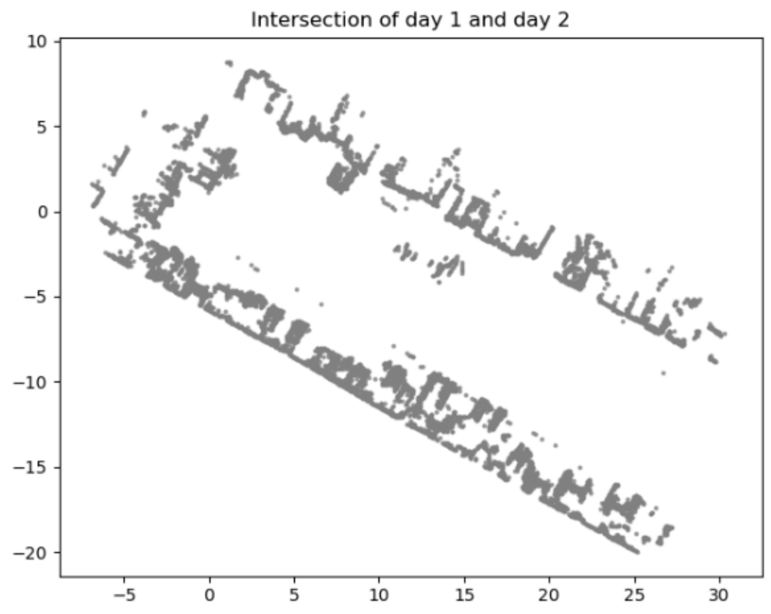
In both quadtrees the regions marked with red are regions of high change level (proportional to the mismatch score computed as explained earlier).

As seen above, isolated points are not marked since they are probably errors.

The below image in shows the quadtree from day 1 updated such that each quad is replaced with the corresponding quad from day 2, if the two differ significantly.



The below image shows the quadtree 4. It shows the intersection between day 1 and day 2 (all the things that didn't change)

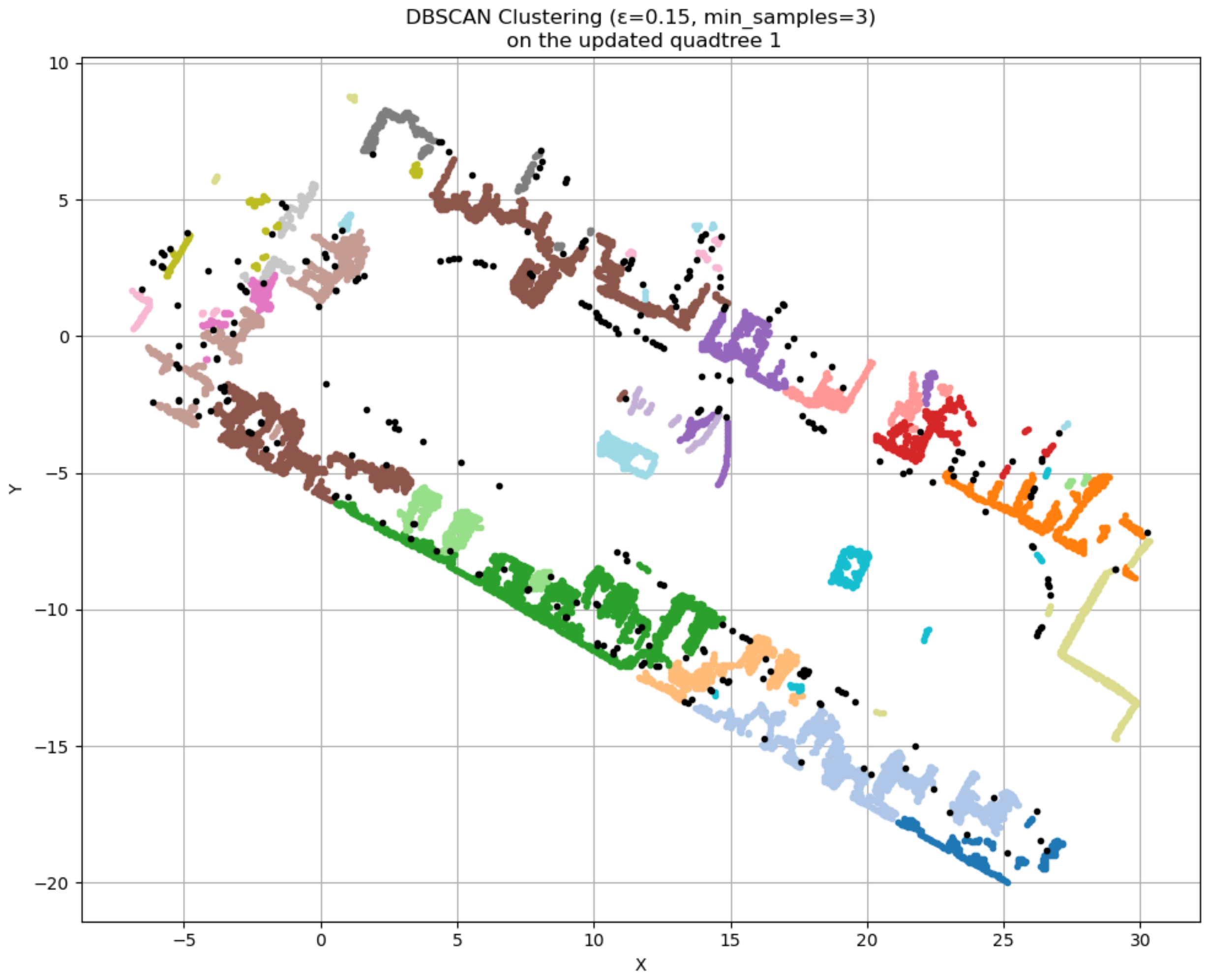


Finally, to provide a higher level of understanding...

c) Clustering with DBSCAN

- Points corresponding to detected changes (Quadtree 3) were clustered using **DBSCAN**, a density-based clustering algorithm.
- Parameters: eps = 0.15, min_samples = 3. (found experimentally)
- This step grouped neighboring changes into **objects or regions of modification**, while filtering noise.
- identify Noise and Outliers .

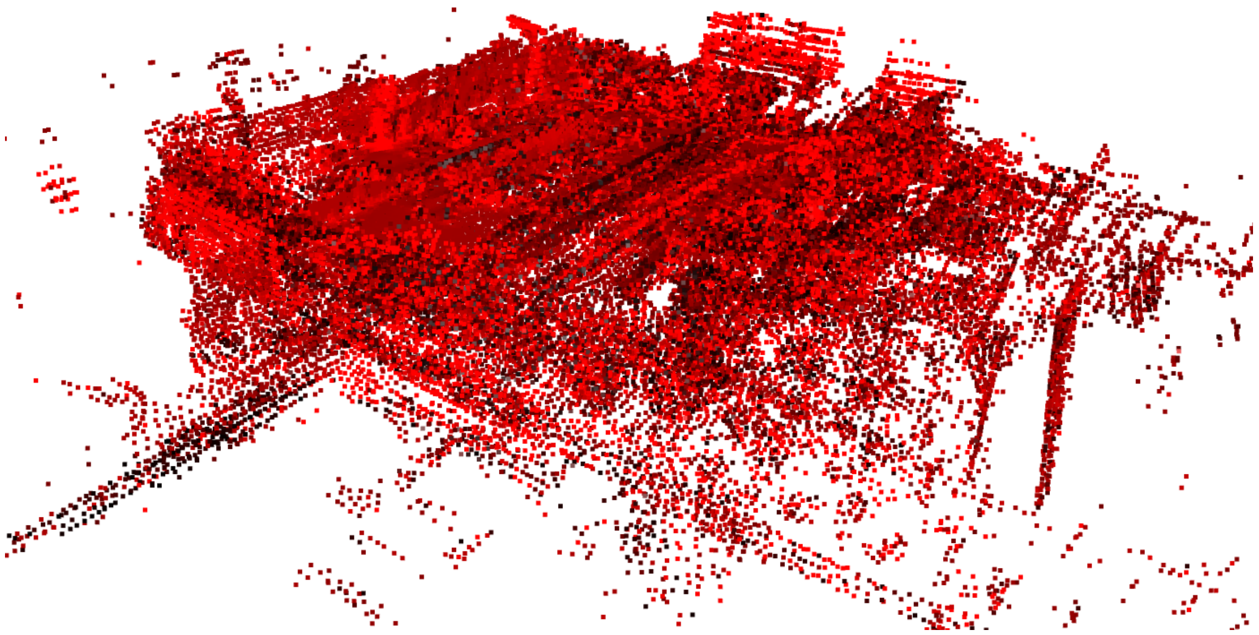
-- adjusting the parameters gives us different levels of detail/resolution in object recognition.



This pipeline allowed for **systematic generation of an updated reconstruction**, while simultaneously isolating the stable regions of the environment.

The 3d case:

The 3D point clouds were obtained from the Ficosa dataset, representing an urban city scene. Below, the 2 pointclouds from 2 timestamps are plotted together.



Pipeline Implementation:

1. Nearest-Neighbor Distance Computation:

   KD-Trees were built for each point cloud to efficiently compute the nearest-neighbor distance between points across timestamps.

   For each point, the distance to its closest point in the other dataset was calculated to quantify local changes.

2. Change Filtering:

   Points with distances exceeding a predefined threshold (THRESHOLD = 1.5 units) were considered significantly changed.

   Changed points from both timestamps were combined to form a dataset representing all modifications.

3. DBSCAN Clustering of Changed Points:

Density-based clustering (DBSCAN) was applied to the changed points to group spatially coherent changes into clusters, with parameters selected to balance sensitivity and noise suppression (eps = 1.5, min_samples = 4).

Clusters corresponding to points from time1 were interpreted as disappeared features, while clusters from time2 represented appeared features.

Noise points (not assigned to any cluster) were filtered out to reduce spurious detections.

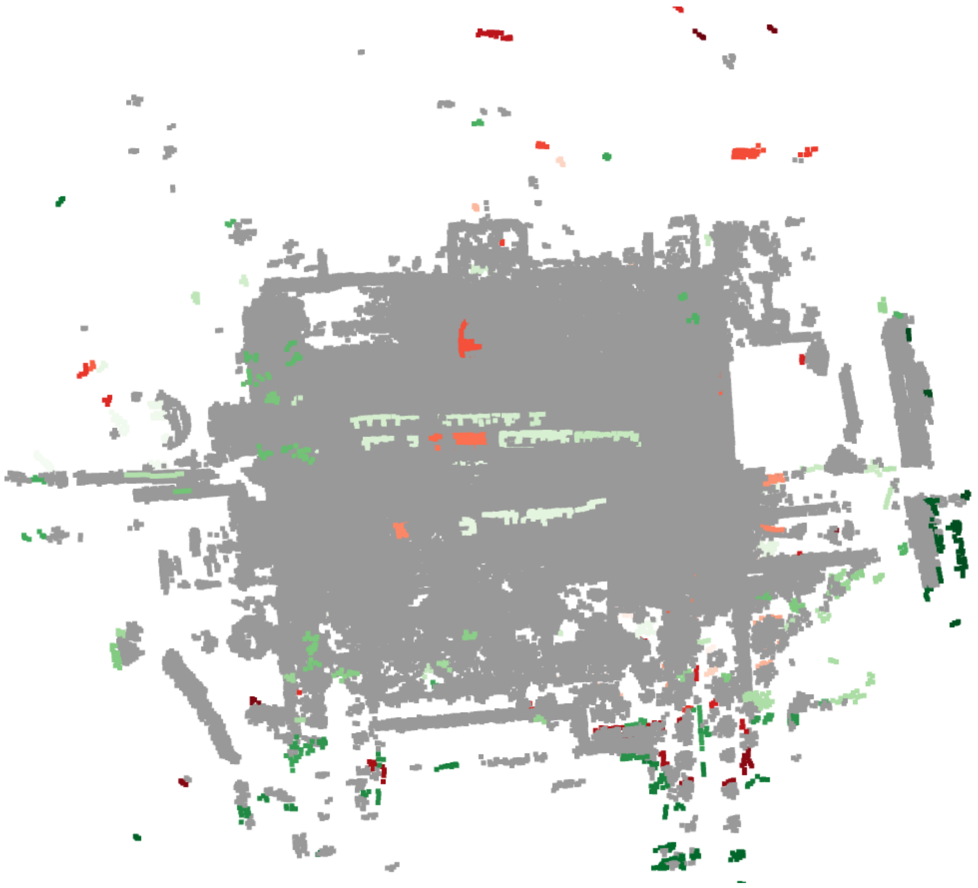
4. Visualization of Changes and Unchanged Regions:

   Unchanged points from both point clouds were displayed in light gray to provide context.

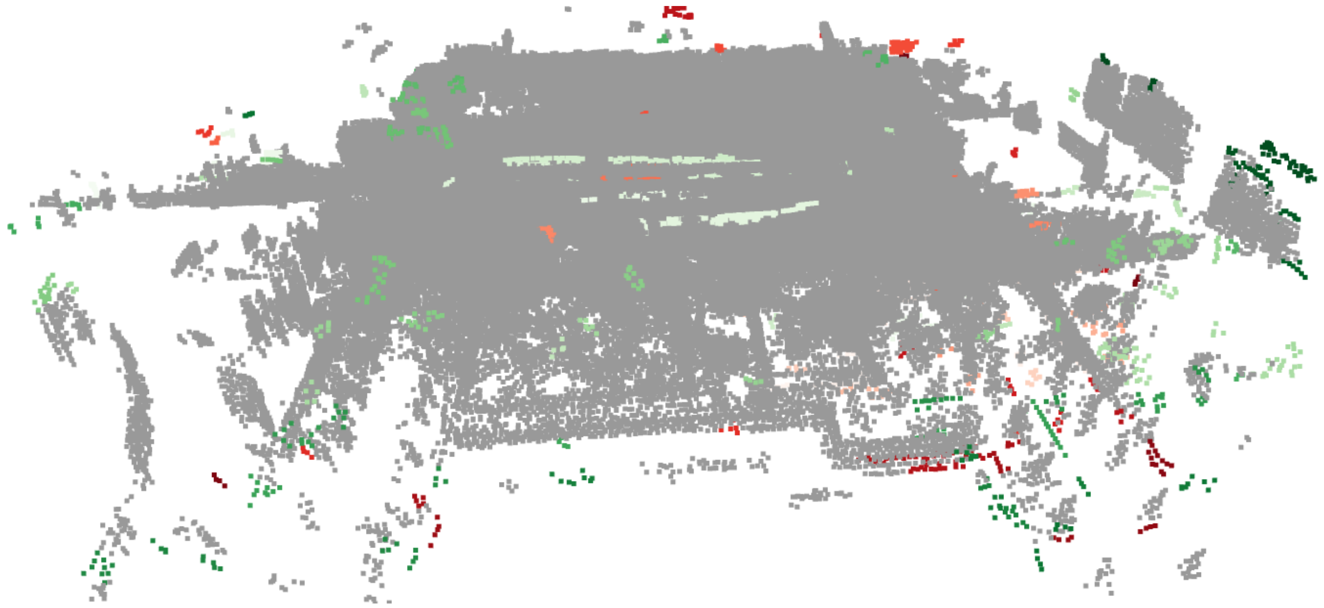
   Changed clusters were color-coded**: red hues for disappeared points (time1), green hues for appeared points (time2)**, and black for noise.

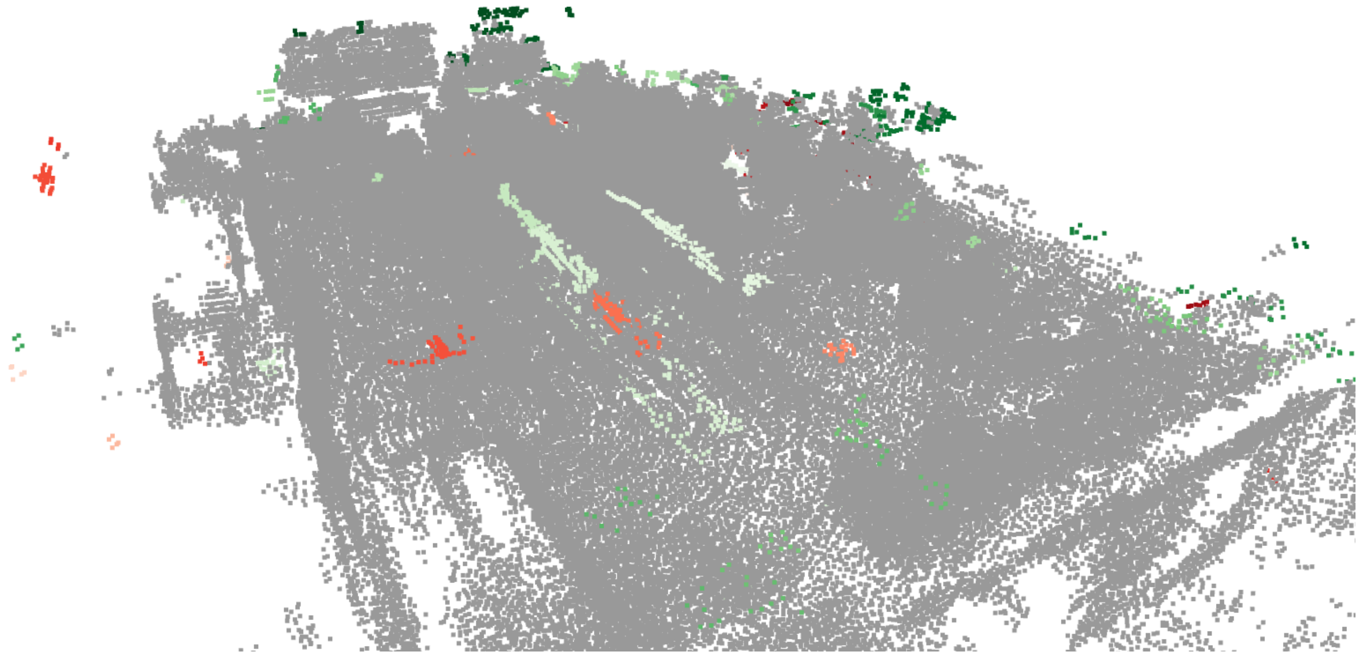
   The final visualization enabled intuitive inspection of volumetric changes in the 3D environment.

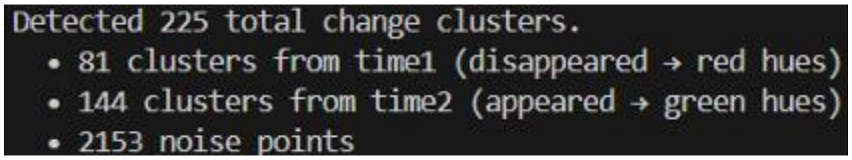
   Below is a "top view" of the result.



Below are some other perspectives of the result.







```
Detected 225 total change clusters.
  • 81 clusters from time1 (disappeared → red hues)
  • 144 clusters from time2 (appeared → green hues)
  • 2153 noise points
```

Summary:

The project successfully demonstrated a pipeline for detecting changes between two point clouds:

- **Point-level visualization** is intuitive but sensitive to noise and small misalignments.
- **Quadtree-based representation** provided a more robust overview by aggregating changes in spatially meaningful cells.
- **Clustering** added another layer of interpretation, enabling object-level analysis of detected changes.

This framework provides a solid foundation for more advanced change detection applications in remote sensing, robotics, and urban monitoring.