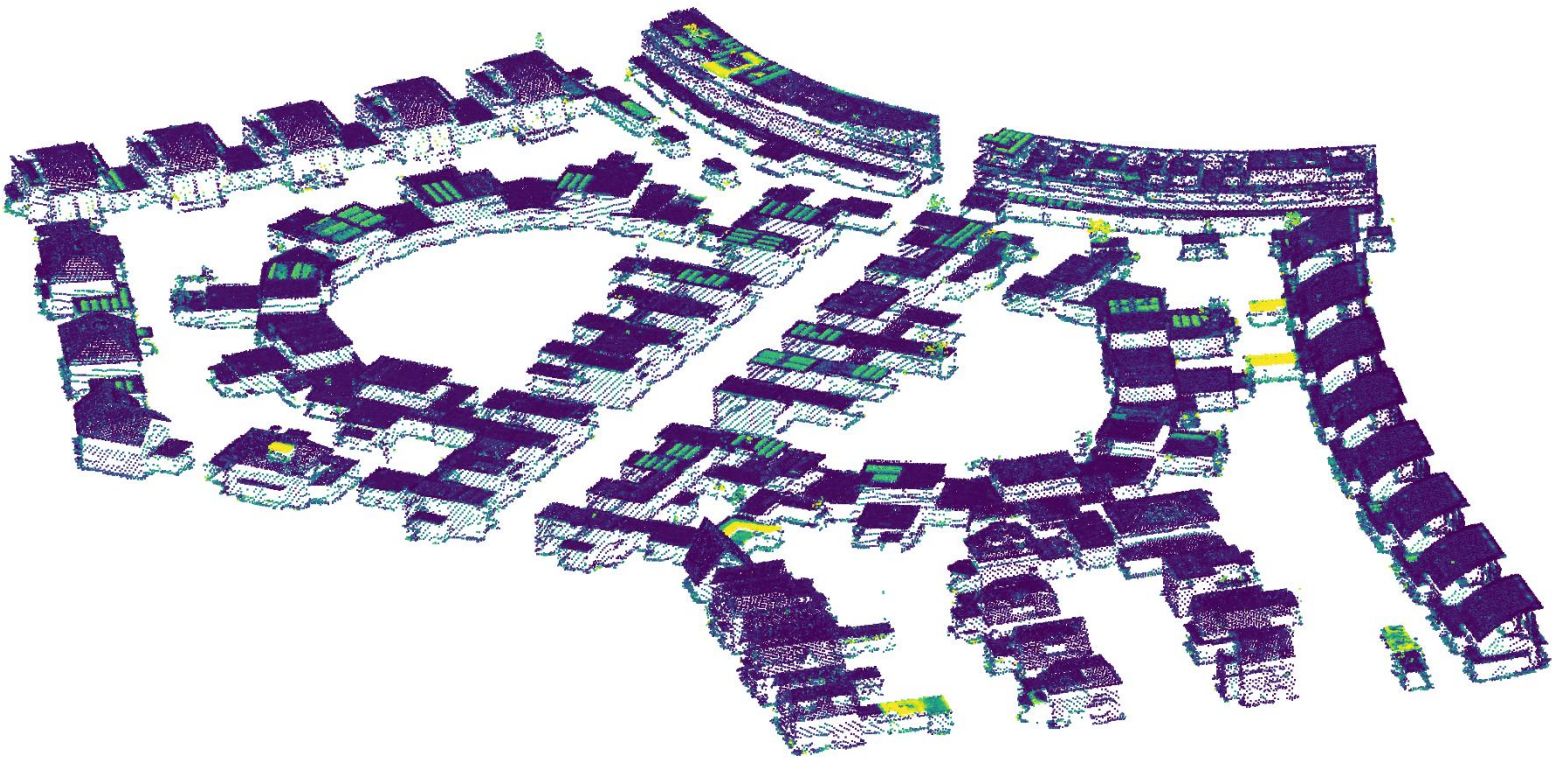
MSc thesis in Geomatics

Detecting Structural Building Changes from Bitemporal Point Cloud Datasets with a Certainty Index – Case Study: AHN and Rotterdam Datasets

Marieke van Arnhem
2025



MSc thesis in Geomatics

Detecting Structural Building Changes from Bitemporal Point Cloud Datasets with a Certainty Index – Case Study: AHN and Rotterdam Datasets

Marieke van Arnhem

June 2025

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of Master of Science in Geomatics

The work in this thesis was carried out in the:

Geo-Database Management Center
Delft University of Technology

GEODELTA Geodelta

Supervisors:	Ir. E. Verbree
	Prof.dr.ir. P.J.M. van Oosterom
Co-reader:	Ir. D.C. Hulskemper
Company supervisor:	Ir. A. Verbraeck

Disclaimer

In this thesis, datasets have been used that were made available for research purposes by the Municipality of Rotterdam (Gemeente Rotterdam). The acquisition of these datasets was carried out by MiraMap on behalf of the Municipality of Rotterdam. The Municipality of Rotterdam is the data owner.

This thesis does not imply that the datasets are owned by Geodelta, nor that the datasets can be requested from Geodelta. Any requests regarding access to the original datasets should be directed to the Municipality of Rotterdam and/or MiraMap, in accordance with the applicable agreements and conditions.

This disclaimer is added after the thesis was assessed and uploaded, to clarify the provenance and ownership of the data used.

Abstract

Up-to-date 3D data is essential for urban planning, building inspections, and monitoring changes in the built environment. While 2D aerial imagery is widely used, it lacks height information and is sensitive to shadows and seasonal effects. In contrast, 3D point clouds provide detailed spatial information and enables better interpretation.

This thesis presents a method for detecting structural building changes using bitemporal airborne laser scanning (ALS) data from the national height model of the Netherlands (AHN) and the Rotterdam municipality. These datasets are pre-aligned in the stelsel van de rijksdriehoeksmeting (RD)-normaal Amsterdams peil (NAP) coordinate system and include building classifications, which allows the focus of this research to be placed directly on detecting change.

Comparing point clouds from different time epochs is challenging due to differences in density, noise, occlusion, and scan geometry. To address this, a random forest (RF)-based classifier is trained on synthetically generated urban scenes that simulate realistic change scenarios. These synthetic scenes are made with different scanning parameters, incorporating diversity in the training dataset. A certainty index is introduced that combines the model's probability output with occlusion visibility across both epochs, providing a confidence measure for each prediction.

The method is applied to real AHN and Rotterdam datasets. Since no labelled ground truth is available, results are evaluated visually. The method successfully identifies structural changes such as dormers and extensions, and also detects moved or temporary objects such as sunshades or picnic tables. When combined with aerial imagery, the approach helps distinguish static from dynamic changes.

This work is innovative in its integration of occlusion-aware certainty scoring, visual certainty feedback, and the automated generation of synthetic training data for change detection.

Acknowledgements

This report represents the effort of a year's work to complete my Master's degree in Geomatics at TU Delft. I am grateful for the support and guidance I received throughout this research journey.

First, I want to thank my TU Delft supervisors. Edward Verbree, my primary supervisor, consistently provided valuable feedback that helped me clarify my writing and recognize the full potential of my work. Peter van Oosterom, my second supervisor, offered critical insights into my workflow and findings, which improved the quality of this research. I am also grateful to Daan Hulskemper, my co-reader, for his detailed and constructive comments on every page. All three of your time, expertise, and encouragement have been valuable.

I would also like to thank Annemieke Verbraeck, my company supervisor at Geodelta, for her expert advice and detailed feedback on the content and structure of this report. Thanks to Niels Treffers for his early support in familiarizing me with the datasets and their management. Conducting research at Geodelta has been a privilege and has provided me with insights into their high standards and expertise. I appreciate the input and help from all colleagues who attended my trial presentations and offered feedback along the way.

Finally, I want to thank Joris for his steady mental support and thoughtful advice during rehearsals and throughout the process. To my parents, thank you for your ongoing encouragement. And to my family and friends, your constant support is highly appreciated.

Contents

Acronyms

ALS airborne laser scanning . . . 109
MLS mobile laser scanning . . . 18
TLS terrestrial laser scanning . . . 6
RF random forest . . . 117
LiDAR light detection and ranging . . . 139
RMSE root mean squared error . . . 141
FOV field of view . . . 62
PRR pulse repetition rate . . . 91
DSM digital surface model . . . 109
AHN national height model of the Netherlands . . . 122
AHN4 national height model of the Netherlands number 4 . . . 135
AHN5 national height model of the Netherlands number 5 . . . 134
C2C cloud-to-cloud . . . 109
C2M cloud-to-mesh . . . 109
RD stelsel van de rijksdriehoeksmeting . . . 24
NAP normaal Amsterdams peil . . . 24
BAG Dutch Key Register of Addresses and Buildings . . . 1
3DBAG 3D building models of the Netherlands . . . 113

Glossary

dynamic occluded Is a condition where a target point is occluded due to a structural change in the scene, such as the removal of a building component. 32, 33, 60, 67–69, 72, 97, 100, 108, 112

epoch Is a specific moment in time at which point cloud data is captured, representing a snapshot of the environment. v, 2, 3, 8, 11, 15, 17–19, 28, 30–35, 41, 44, 46, 47, 49–60, 67, 69, 70, 72, 78, 83, 87, 95, 99, 103, 105, 106, 109, 112–114, 129, 130, 132, 139–141, 144

reference point cloud dataset Is the point cloud dataset used as a baseline or comparison for another (target) dataset. 3, 17, 30, 32, 34, 40, 51, 54, 58–61, 67, 70, 72, 83–86, 92, 94, 97–101, 105–108, 113, 129–131, 134–136

stability factor Is the ratio of the number of neighboring points within a sphere around a target point to the total number of points within a vertical cylinder around the same point, used to quantify local stability. 18, 20, 50, 52–55, 67–69, 87, 96, 99, 105–109, 112

static occluded Is a condition where a target point is occluded by a stable object that exists in both the reference and target scans. 32, 33, 60, 68, 69, 72, 112

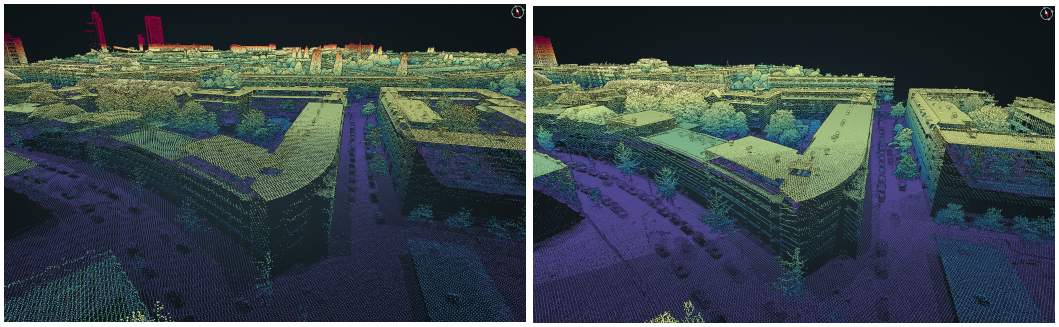
target point Is a specific point in the target point cloud currently being processed or analyzed. 18, 20, 50, 51, 54, 61–67, 106, 108, 149–152

target point cloud dataset Is the point cloud dataset being analyzed to detect changes relative to the reference dataset. 3, 10, 30, 34, 40, 51, 54, 59–61, 70, 72, 83, 86, 87, 99–101, 106, 108, 113, 129, 131, 134–136

1. Introduction

The Netherlands is a well-organized country where almost all land is carefully planned and used. However, the demand for space is increasing. Important needs such as housing, renewable energy, and climate adaptation, all compete for land (Evers et al., 2023). To support good planning and decision making, detailed 3D data is becoming more important.

One widely used 3D representation are point cloud datasets. Point clouds can be used to make 3D city models, like 3D building models of the Netherlands (3DBAG), which help with tasks such as noise mapping, energy estimation, and city management (Biljecki et al., 2015). Point clouds can also be used directly for analysis and measurements. In the Netherlands, the AHN is a national point cloud dataset collected using ALS. It is used in many applications, such as dike monitoring and noise mapping (Manders, 2024). However, many users only access the AHN data indirectly, often through rasterized 0.5-meter grids or integrated in the 3DBAG model combined with the Dutch Key Register of Addresses and Buildings (BAG), while only a small number use the raw point cloud data directly. Another source is the Rotterdam point cloud dataset, which is not publicly available but is updated every year. The municipality uses it for tasks like BAG-WOZ inspections, city planning, and maintaining public spaces and historic buildings (van Bochove, 2019). Figure 1.1 shows the same area from both datasets. The Rotterdam dataset has a higher point density, while the AHN provides more detailed classifications.



(a) Part of the urban area in national height model of the Netherlands number 5 (AHN5) in Rotterdam. Color is based on height.

(b) Part of the urban area in the 2024 Rotterdam dataset. Color is based on height.

Figure 1.1.: The same area in the AHN and Rotterdam datasets.

Point clouds play a crucial role in the development of smart cities (Lemmens, 2018) and the management of urban environments. Detecting changes in buildings is essential for maintaining accurate and up-to-date 3D data. This is particularly important for urban planning, where knowing the current state of the built environment is necessary to assess space availability and plan new developments. Furthermore, it assists in legal validation by verifying whether construction complies with approved plans and regulations, such as height limits

or building boundaries. Additionally, the detection of newly installed solar panels using point clouds contributes to environmental monitoring and helps manage subsidy programs and regulatory compliance.

Many studies on change detection still use aerial images because it is easier to obtain and updated more frequently (Qin et al., 2016, Kharroubi et al., 2022, Xiao et al., 2023). But aerial images have limitations. It does not show height, can be affected by season, and can give a distorted view (de Gélis et al., 2021b; Kharroubi et al., 2022). In contrast, light detection and ranging (LiDAR) point clouds maintain the true 3D shape of objects (Nofulla, 2023), can see through trees (Politz and Sester, 2022), and are not affected by lighting conditions (Kharroubi et al., 2022). These advantages make point clouds useful for studying city growth, damage, and forest changes (Kharroubi et al., 2022).

Detecting changes in point clouds is a difficult task. Point clouds collected at different times cannot be directly compared because the points do not line up exactly (Winiwarter et al., 2021). Many factors can affect the outcome, such as how the data was collected, the quality and density of the points, and the type of analysis used. One of the main difficulties in detecting changes is distinguishing actual changes from false alarms, often called pseudo-changes. These pseudo-changes can arise from a variety of sources depending on the situation and goals of the detection method. Common examples include occlusions, errors in aligning the point clouds (registration errors), and temporary factors such as seasonal vegetation growth or moving objects like cars. Effectively addressing these pseudo-changes is crucial for improving the reliability of change detection results. Occlusion, in particular, is a common source of error: when parts of a building or area are not visible in one of the epochs, many change detection methods may incorrectly interpret missing data as structural change.

This research focuses on detecting *structural changes* in buildings. These include additions, removals, demolitions, changes in shape, and height modifications. However, structural damage or material changes are not considered. By limiting the scope to buildings, the detection method can be more optimized. A complete method is developed to address the previously discussed challenges. The goal is to classify each point on or near a building as either "changed" or "unchanged." The criteria for this classification and how it is implemented are discussed in detail in the implementation chapter. The specific type of change is not automatically determined and is intended for manual interpretation. Figure 1.2 provides an overview of this process.

The method is trained on a synthetic dataset that includes various house types, change scenarios, and scanner settings to represent real-world conditions. A synthetic dataset is used so that a wide range of changes can be simulated and labelled automatically.

The algorithm consists of two main steps. First, a classification model predicts the probability that a point has changed between the two epochs. The model uses different types of input features: features describing the point itself, features based on its local neighborhood, and features comparing the point across two time periods. This outcome probability score ranges from 0 to 1, where 1 indicates high confidence that the point has changed, and 0 indicates high confidence that it has not. Second, this probability score is combined with occlusion information to derive a certainty index. The certainty index reflects the overall confidence in the prediction by also considering whether the point was visible in both scans. If a point is occluded in either epochs, the prediction is treated with greater caution. A certainty index close to 0.5 reflects high uncertainty, as it lies between the extremes of full certainty in change (1) and full certainty in no change (0). The certainty index thus integrates

both model output and visibility analysis to support more reliable interpretation of change detection results. In this research the reference point cloud dataset is the point cloud that is compared to and the target point cloud dataset is the point cloud dataset of interest.

The final method is evaluated using real-world data from the national height model of the Netherlands number 4 (AHN4), AHN5, and the ALS Rotterdam 2023 and 2024 datasets. The AHN datasets are particularly well-suited for this research because they include classified point clouds, allowing the focus to remain on change detection rather than semantic classification. In addition, all datasets were acquired using high-quality ALS and are aligned within the same RD-NAP coordinate reference system, ensuring spatial consistency and precise positioning of the points. The performance of the method is assessed primarily through visual inspection of the results.

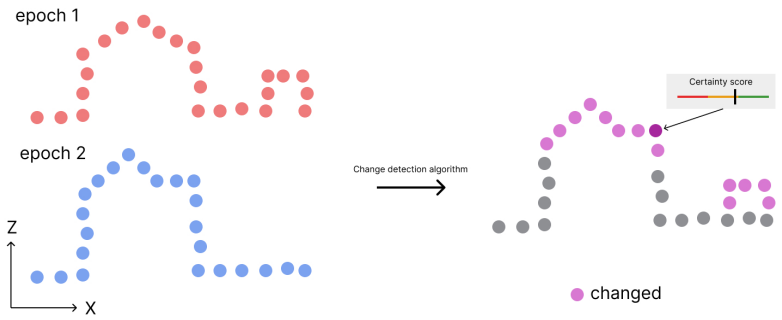


Figure 1.2.: The goal of this thesis.

This research aligns well with the Master Geomatics programme, as it focuses on 3D spatial data, in particular, point clouds. It covers the full pipeline: from generating custom data to processing and transforming it into valuable information. The work involves key aspects of geomatics, such as working with geometries, visualizing 3D data, and applying these techniques within the built environment.

This research is done in collaboration with Geodelta, a company that specializes in geo-information. Geodelta is known for its emphasis on accuracy and quality. They support this thesis by providing data, sharing knowledge, and offering expert advice.

1.1. Research objectives

The main research question of this thesis is:
To what extent can a building change detection method between two epochs of aligned point cloud datasets be developed to detect structural changes, maximizing reliability by using a certainty index?

The goal is to develop a reliable algorithm that detects building changes in the national height model of the Netherlands number 4 (AHN4), national height model of the Netherlands number 5 (AHN5), and Rotterdam 2023 and 2024 datasets. This can be broken down into the following sub-questions:

1. How can realistic urban airborne laser scanning (ALS) point cloud datasets be simulated to support the training and testing of change detection methods, including different data quality and occluded areas?
2. How can a change detection method be created that correctly finds building changes while dealing with uncertainty in point cloud data?
3. How can occlusion be included in the detection process to improve the certainty of the prediction?
4. How well does the developed method perform on real-world datasets like national height model of the Netherlands (AHN) and Rotterdam?

1.2. Thesis Outline

This thesis consists of eight chapters. After this introduction, Chapter 2 provides background information on point clouds, change detection and the random forest (RF) algorithm. The datasets used in this research, national height model of the Netherlands (AHN) and Rotterdam datasets, are described in Chapter 3. Chapter 4 explains the research method, divided into four steps: understanding the changes and datasets, creating a simulated dataset, building a detection method, and testing it with real data. Chapter 5 describes the implementation of these steps in detail. The results of the method are presented in Chapter 6, followed by a discussion of the findings and limitations in Chapter 7. The main conclusions are summarized in Chapter 8, and suggestions for future research are given in Chapter 9.

2. Theoretical background and related work

This section reviews the relevant literature for this research. It starts with a short overview of point cloud data and storing point cloud data in Section 2.1 and Section 2.2. Then, methods for detecting changes in point clouds are discussed in Section 2.3. The machine learning algorithm used in this research, RF, is explained in Section 2.4. After that, the uncertainties and errors related to point clouds are described in Section 2.5. Finally, strategies to deal with the limitations of point cloud and change detection are presented in Section 2.6. The chapter ends with some conclusion in Section 2.7.

2.1. Introduction to Point Cloud Data

This research focuses on point cloud datasets acquired through airborne laser scanning (ALS). In this section, the process of capturing point clouds will be explained, followed by an overview of common applications. Additionally, the different scanning patterns used by ALS systems will be described.

Point clouds are collections of points that show the shape of visible objects and surfaces in 3D space. Sometimes, they can also include surface texture information (Vosselman, 2010). These datasets can be created using photogrammetry (for example, with dense image matching) or using light detection and ranging (LiDAR). This research focuses on the second method, LiDAR. A LiDAR system sends out laser pulses toward the ground and measures how long it takes for the pulses to return. Since light travels at a constant and known speed, this time can be used to calculate the distance to the surface (Vosselman, 2010). With the distance, the angle of the pulse, and the position of the scanner, the 3D coordinates of the point can be determined.

Due to the nature of surfaces, a single laser pulse can produce multiple echoes. ALS systems are usually able to record at least the first and last return, and often also intermediate echoes (Vosselman, 2010). Besides recording the position of each point, LiDAR systems can also measure the intensity of the reflected signal. This value shows how much of the laser pulse was reflected by the object or surface that caused the return (Vosselman, 2010). It is important to note that this intensity is usually not corrected or normalized. In addition to information as position and intensity, each point in a point cloud can also store other characteristics, such as color, the number of returns, or classification labels.

Point clouds have many applications, including monitoring of structural and environmental changes, viewshed analysis, solar energy potential estimation, deformation studies, volume calculations, and vegetation or hydrology analysis (van Oosterom, 2016). A key benefit of point clouds is that they can represent environments in a realistic way without needing to be converted into a specific model. However, their high level of detail also means they create

very large datasets, which makes storage and processing difficult (van Oosterom, 2016). These challenges are discussed further in Section 2.5.

Point clouds can be collected using different types of laser scanners: ALS, terrestrial laser scanning (TLS), or mobile laser scanning (MLS). Each of these methods has its features, but in this research, the focus is on ALS. There are various scanning patterns in ALS, which are shown in Figure 2.1. The advantages of each pattern depend on the scanner used.

In Figure 2.1a, the rotating mirror scanner produces a point density that is evenly distributed. In Figure 2.1b, the oscillating mirror scanner has a higher point density at the edges. Figure 2.1c shows the fibre-optic scanner, which does not require mechanical movement of its components. Finally, in Figure 2.1d, the slanted rotating mirror scanner has the advantage of being able to capture data from facades at the flightline, so below the scanner. (Winiwarter et al., 2022)

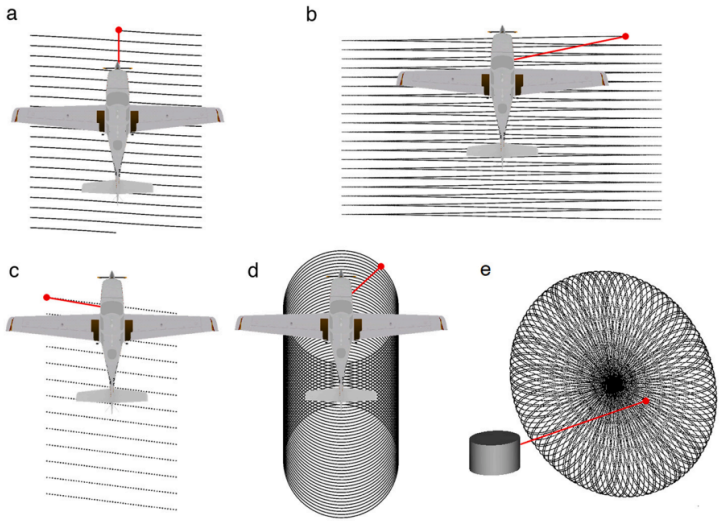


Figure 2.1.: Different scanning patterns. (a) The rotating mirror. (b) The oscillating mirror. (c) The fibre-optic line scanner. (d) The slanted rotating mirror. (e) The Risley prisms. Source figure: Winiwarter et al. [2022].

2.2. Storage and Indexing of Point Clouds

There are several ways to store and query point clouds efficiently. This section explains common methods such as Octrees, KD-trees, and Space-Filling Curves.

In a 2D KD-tree, see Figure 2.2, the space is split in two every time a point is inserted. The first split is done by placing a vertical line through the point, dividing the space into left and right parts (along the x-axis). The next point splits one of these parts with a horizontal line (along the y-axis), dividing it into upper and lower regions. This alternating pattern between vertical and horizontal splitting continues until a leaf node is formed, which defines a rectangular region (van Oosterom, 1999).

To perform a range query, the algorithm starts at the root and checks whether the query range overlaps with the two child regions. If it does, those regions are also checked, continuing until the leaf level is reached (van Oosterom, 1999). A known drawback of the KD-tree is that the structure depends on the order in which the points are inserted. This can lead to an unbalanced tree, which in the worst case may have one level per point, resulting in poor performance (van Oosterom, 1999).

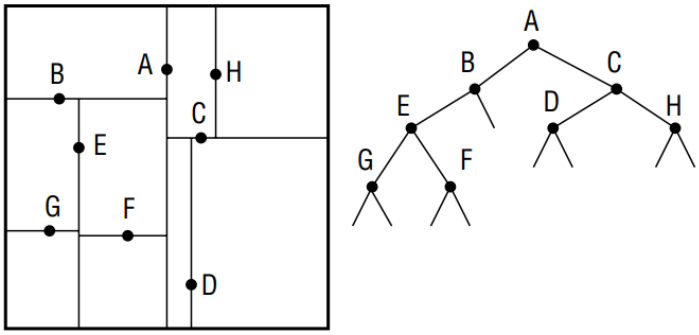


Figure 2.2.: A 2D example of a KD-tree structure, which recursively partitions space using axis-aligned splits. This data structure supports efficient spatial queries. Source figure: van Oosterom [1999].

Space-filling curves, also called tile indexing methods, convert a 2D (or 3D) space into a 1D sequence for storage and retrieval (van Oosterom, 1999). The Morton curve, see Figure 2.3, also known as the Z-order curve, does this by interleaving the binary representations of each coordinate to assign a unique index to each point (Psomadaki, 2016).

A disadvantage of the Morton curve is that nearby points in the index may not be close in space. This makes it less suitable for tasks like nearest-neighbour search. To find the nearest neighbour using a space-filling curve, the N closest entries in the one-dimensional array must be selected. Then, the actual distances between the query point and these candidates must be calculated to determine which one is truly the nearest.

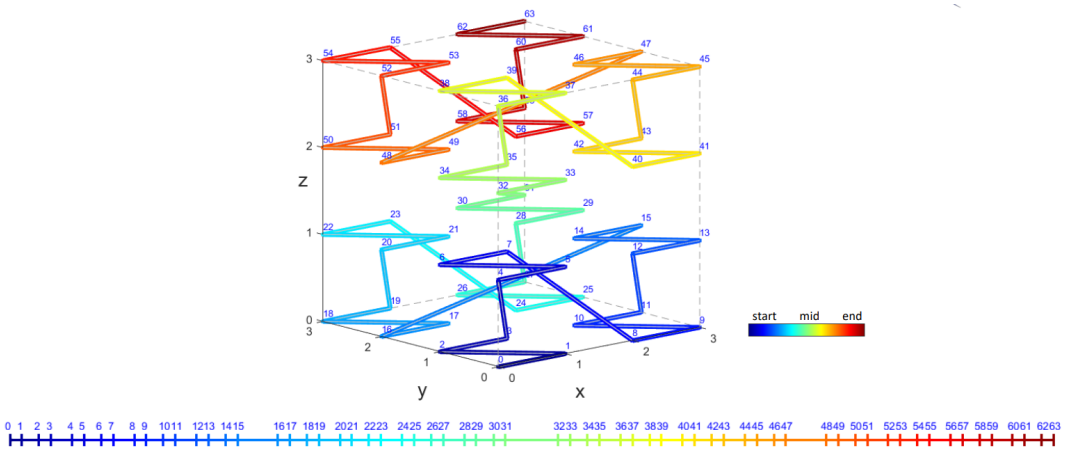


Figure 2.3.: Spatial subdivision using a 3D Morton (Z-order) curve. The space is traversed in a way that preserves locality in the linear index. Source figure: Diaz et al. [2024b].

A third method for storing point clouds is the *octree*, see Figure 2.4, a hierarchical data structure that divides the space into cubes (Psomadaki, 2016). The 3D space, typically organized in a $(2^n, 2^n, 2^n)$ grid, is recursively split into eight smaller cubes, called *octants* (Psomadaki, 2016). Octrees can be used to find nearest neighbours, but they require a large amount of storage.

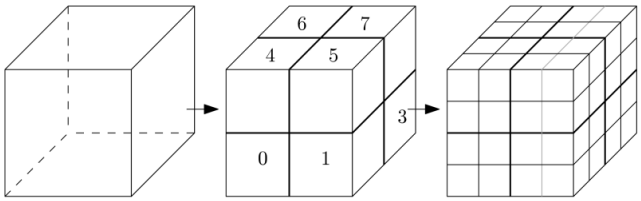


Figure 2.4.: Hierarchical subdivision of 3D space using an octree. Each node recursively divides into eight child nodes, enabling efficient spatial indexing and search. Source figure: Psomadaki [2016])

2.3. Change Detection in Point Clouds: Methods Overview

One of the main challenges in detecting changes in point clouds is differentiating real changes in the environment apart from false changes, often called pseudo-changes. Pseudo-changes vary depending on the situation and goal of the change detection method. Examples include occlusions, errors in aligning the point clouds (registration), and temporary factors such as seasonal vegetation or moving objects like cars. Many methods have been developed to detect changes in urban environments using point clouds. For detailed reviews, see Qin et al. [2016], Kharroubi et al. [2022], Stilla and Xu [2023], and Xiao et al. [2023]. This section gives an overview of the methods most relevant to this research.

Method Classifications in Literature

Before discussing specific techniques, it is helpful to understand the different types of change detection methods used for point clouds. Kharroubi et al. [2022] divided these approaches into three categories: traditional methods, handcrafted learning methods, and deep learning methods. An overview of this classification is shown in Figure 2.5.

In contrast, Stilla and Xu [2023] categorized change detection methods based on how they process point clouds. These include point-based, voxel-based, and segment-based approaches. Point-based methods compare individual points and detect changes when the distance between corresponding points exceeds a threshold. Voxel-based methods divide the space into a grid and compare the occupancy of each voxel between epochs. Segment-based methods group points into objects or regions and compare these groups to detect geometric differences. This classification is illustrated in Figure 2.6.

Stilla and Xu [2023] also discussed the variety in the output of change detection methods. Some methods only produce a binary result, showing whether a change occurred or not. Others provide more detailed information, such as the type of change (e.g., a new tree or

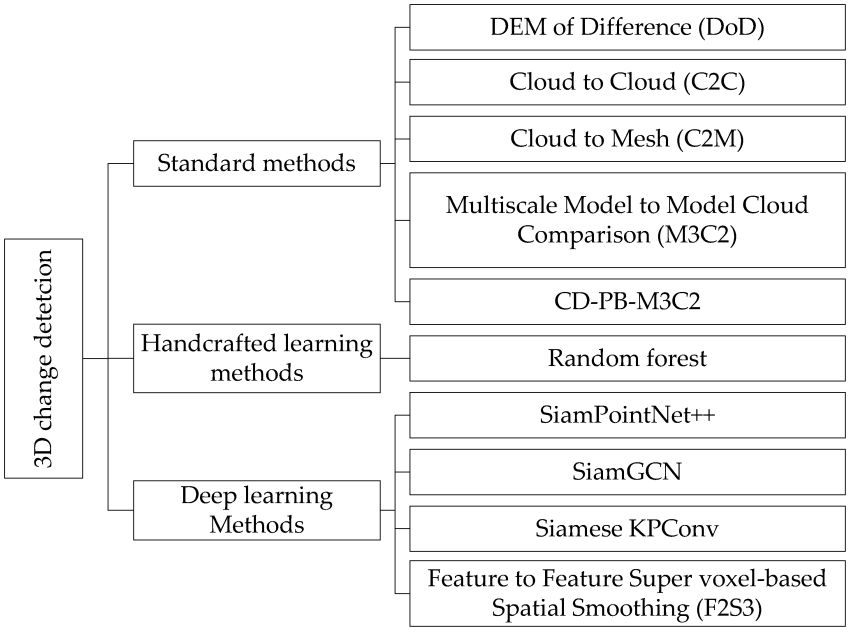


Figure 2.5.: The 3D change detection methods described by Kharroubi et al. [2022], divided into three categories. Source figure: Kharroubi et al. [2022].

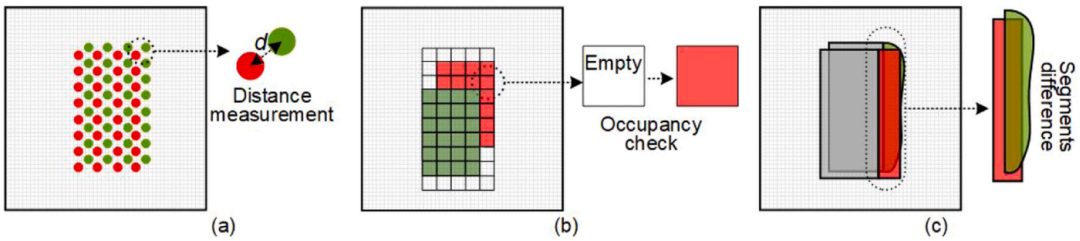


Figure 2.6.: Different strategies for point cloud change detection, as discussed by Stilla and Xu [2023]. Source figure: Stilla and Xu [2023].

a demolished building). Some methods also include quantitative information, such as the amount of change in height between two time points.

Finally, Stilla and Xu [2023] emphasized that the goals of change detection vary depending on the application. Many studies focus on changes in the built environment, particularly land use. Other common applications include monitoring urban building changes and tracking construction progress.

Since this research focuses on building change detection using the national height model of the Netherlands (AHN) and Rotterdam datasets, comparing existing change detection methods is relevant but also challenging for two main reasons. First, the nature of the objects under investigation strongly influences the choice of an appropriate method. For instance, techniques designed to detect changes in soil deformation, forest environments, or urban structures differ significantly in their assumptions, input requirements, and processing strategies. Second, the performance of change detection methods varies considerably across datasets. Differences in data quality, acquisition methods, and scene characteristics lead to a wide range of outcomes. As a result, direct comparison between methods is difficult, particularly when each study relies on different real or synthetic datasets. A major limitation in this field is the absence of a standardized, well-labelled dataset for benchmarking. This gap has been recognized as a key issue by de Gélis et al. [2021a] and Kharroubi et al. [2022].

Point-Based Methods: Cloud-to-Cloud and Cloud-to-Mesh

Commonly used techniques include cloud-to-cloud (C2C) and cloud-to-mesh (C2M) comparisons. Figure 2.7 shows an illustration of these methods. In C2C, for each point, the closest neighbor in the other epoch is found, and the distance between them is measured. Although C2C is simple, it is sensitive to outliers and small differences. Also, the best threshold for change detection depends on the dataset (de Gélis et al., 2021b). In C2M, the change is measured as the distance between a point and the surface (mesh) created from the other dataset. This method requires converting the target point cloud dataset into a mesh, but this can create triangular surfaces with gaps or artifacts (Kharroubi et al., 2022). Both methods have difficulties when point densities vary within or between datasets, and when parts of the data are hidden (occlusion) (Kharroubi et al., 2022).

While C2C andC2M are straightforward techniques, their reliance on direct spatial proximity makes them less effective in urban datasets with variable point density and frequent occlusion. In contrast, this research moves beyond direct distance thresholds by incorporating semantic and geometric features at the point and neighborhood level, trained through an random forest (RF) classifier.

Raster-Based Approaches: DSM Differencing

A popular alternative is to convert point clouds into an digital surface model (DSM) for change detection (Xiao et al., 2023). Figure 2.8 shows an example of the differences in DSM. The area is divided into grid cells, and in each grid, at most one point is selected (e.g., the lowest point). These points are then connected using triangulation, assigning one height value per grid. To calculate changes, the grids are subtracted from each other. Kharroubi et al. [2022], Stilla and Xu [2023], de Gélis et al. [2021b] state that this method is efficient,

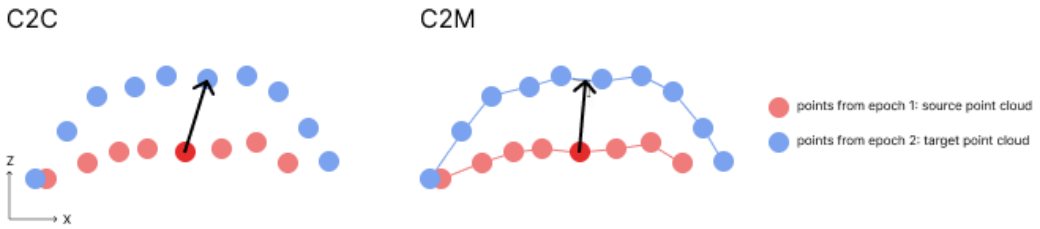


Figure 2.7.: Illustration of how Cloud-to-Cloud (left) and Cloud-to-Mesh (right) methods work.

but introduces information loss due to interpolation and only accounts for differences in a predefined direction. It also struggles to accurately capture precise building boundaries.

Due to these limitations, DSM differencing is not adopted in this research. However, raster-based processing is still used in a limited scope, to associate points with specific buildings via footprint overlays, while preserving full 3D detail for the core classification process.

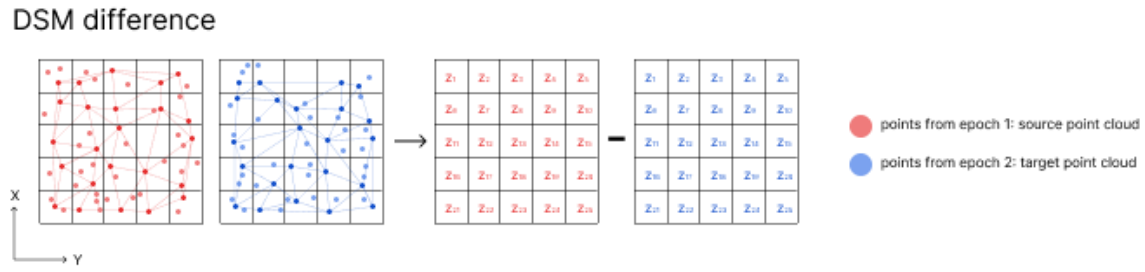


Figure 2.8.: Illustration of how the DSM difference is calculated.

Learning-Based Approaches: Random Forests and Deep Learning

The use of machine learning and deep learning in change detection is increasing. Figure 2.9 shows the general workflow of a learning-based algorithm. For each point or patch of points, features are calculated. These features are based on information from one or both epochs. Then, a classifier (either machine learning or deep learning) is trained using a labelled dataset with these features. Based on the labels, the algorithm identifies the changes and the type of change (e.g., added or removed building). The RF algorithm, for example, has shown promising results (De Gélis et al., 2021, de Gélis et al., 2021b, Kharroubi et al., 2022, Nofulla, 2023). However, its performance is sensitive to the quality of the features (Nofulla, 2023) and decreases as the data becomes more heterogeneous (de Gélis et al., 2021b). Another interesting approach involves combining Siamese architectures with deep learning, which has shown strong performance (Kharroubi et al., 2022, De Gélis et al., 2021). Kharroubi et al., 2022 state the good performance of learning-based methods, but also states their limitation of a class-imbalanced problem and the probability of failing in minority

classes. de Gélis et al. [2021a] claim to be the first to use deep learning directly on raw point clouds for change detection and characterization.

While deep learning methods such as Siamese networks have demonstrated high performance, especially in structured environments, their effectiveness relies heavily on large labelled datasets and extensive training. Given the limited availability of labelled airborne laser scanning (ALS) data in AHN and Rotterdam, this study adopts an RF model, which performs well with smaller datasets and provides interpretable outputs. Furthermore, this research expands the standard RF framework by incorporating occlusion-based uncertainty modelling.

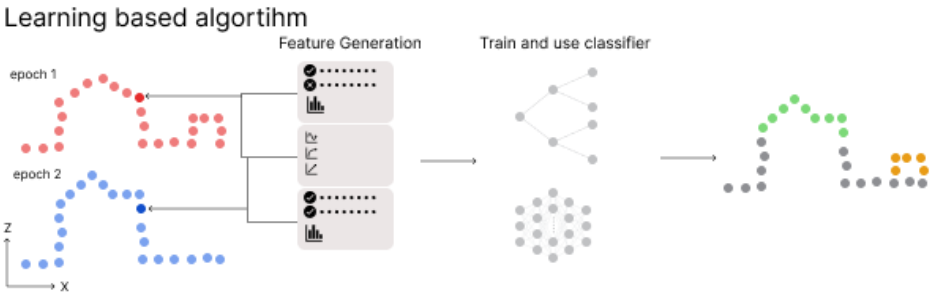


Figure 2.9.: Illustration of how a learning based algorithm works, in general.

De Gélis et al. [2021] and Kharroubi et al. [2022] suggest the development of more spatially-aware deep neural networks for urban change detection, as these models can better understand the structured nature of objects on a global scale. Kharroubi et al. [2022] suggest doing more research on the use of graph neural networks for change detection, integrating the progress made in 3D segmentation, and exploring finer levels of structural detail to ensure scalability with large datasets. Stilla and Xu [2023] also highlight the importance of integrating semantic information and addressing measurement uncertainties in future methods.

Summary and Relevance to This Research

Table 2.1 summarizes the main point cloud change detection methods discussed in this section. The classification is based on the categories introduced by Kharroubi et al. [2022] and the processing strategies outlined by Stilla and Xu [2023]. The method developed in this research positions itself between traditional geometric differencing and modern deep learning by combining a handcrafted feature-based RF model with occlusion-aware uncertainty modelling.

2.4. Random Forest in Point Cloud Analysis

The random forest (RF) classifier is used in this research, and the reasons for selecting it will be discussed at the end of this section. First, an overview of the RF classifier is provided. Then, its application in point cloud change detection methods is explained.

Table 2.1.: Overview of point cloud change detection methods and their characteristics, related to this research. This table is developed by the author to position the proposed method within existing techniques.

Category	Technique	Input Data	Strengths	Weaknesses	Relation to This Research
Traditional, Point-based	C2C	Raw point clouds	Simple, intuitive	Sensitive to outliers, dataset-specific thresholds	Serves as baseline for evaluating the proposed RF-based change detection method using point-level comparisons.
Traditional, Point-to-surface	C2M	Point clouds and meshes	Captures surface changes better	Mesh generation may introduce artifacts	Inform feature design and understanding of geometric changes but not directly applied in this research.
Traditional, Raster-based	DSM differencing	Rasterized DSMs	Efficient	Information loss, directional bias, boundary issues	Provides complementary insights; not primary focus due to geometric detail loss inherent in rasterization.
Handcrafted learning, Point-Based	RF	Extracted features from point clouds	Robust, interpretable	Sensitive to feature quality, performance degrades with heterogeneous data	Core method developed here: uses synthetic labelled datasets, combines point features, occlusion modeling, and house-level preprocessing.
Deep learning, Segment-Based	Siamese networks	Raw point clouds or features	High accuracy, learns complex patterns	Needs large labelled data, class imbalance issues	Potential future direction.

Explanation RF Model.

The RF model is visualized in Figure 2.10. As described by Biau and Scornet [2016], an RF classifier consists of an ensemble of M regression trees that are trained independently on different subsets of the dataset. Each tree makes a prediction, and the final decision is determined by a majority vote from all trees.

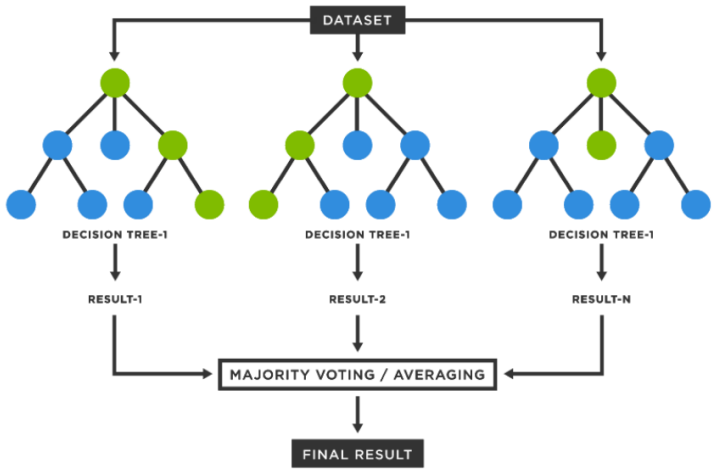


Figure 2.10.: Visualization of the RF classifier model. Source figure: Nofulla [2023].

A decision tree is a supervised learning model that partitions data into subsets by applying decision rules based on feature values. These splits are chosen to maximize the homogeneity, or purity, of the resulting subsets. A node is considered pure when all samples within it belong to the same class. To guide these splits, the model uses impurity metrics, such as the

Gini impurity, which is defined as:

$$G = 1 - \sum_{i=1}^{C} p_i^2 \tag{2.1}$$

where p_i is the proportion of samples belonging to class i within a node, and C is the total number of classes. A lower Gini impurity indicates a purer node, with a value of 0 representing a completely pure node (i.e., all samples are of the same class). The tree construction process selects feature splits that most effectively reduce impurity. However, it is important to control the depth of the tree, as overly deep trees may capture noise in the training data and result in overfitting, reducing generalization performance [Yıldırım, 2020].

In addition to predicting the final class, an RF model can also estimate the probability for each class. These probabilities are based on how many training samples of each class are found in the leaf nodes. For each decision tree, the probability is calculated as the fraction of samples in the leaf that belong to a given class. Then, the average is taken over all trees in the forest (Biau and Scornet, 2016). The formula is:

$$\hat{P}(y = 1 \mid \mathbf{x}) = \frac{1}{T} \sum_{t=1}^{T} \hat{P}_t(y = 1 \mid \mathbf{x}) \tag{2.2}$$

Where:

- T is the number of trees in the forest,
- $\hat{P}_t(y = 1 \mid \mathbf{x})$ is the predicted probability from tree t, which is the fraction of training samples of class 1 in the leaf where $\mathbf{x}$ ends up.

Niculescu-Mizil and Caruana [2005] point out that RF classifiers often do not produce probability values that are very close to 0 or 1. This happens because each decision tree is trained on a random subset of the data and features, a technique known as bagging. As a result, not every tree sees all the important information. This can make the final prediction more cautious, meaning the probability values tend to stay away from the extremes of 0 and 1.

Application in Point Cloud Change Detection.

The application of an RF model for 3D change detection has demonstrated significant effectiveness, as evidenced by the study of Tran et al. [2018]. This approach provides several notable advantages: (1) resistance to overfitting, (2) fast and efficient processing of large datasets, and (3) the ability to perform well even with relatively small training datasets, unlike convolutional neural networks. More specifically, Tran et al. [2018] reported a precision of 90.93% and 92.04% when using an RF classifier. Other benefits of RF include their robustness to noisy data and the interpretability of their results (Belgiu and Drăguţ, 2016). However, there are also limitations. For example, de Gélis et al. [2021a] note that RF can struggle when the training data and test data is different, while Nofulla [2023] highlights that RF is sensitive to feature quality.

In point cloud change detection, features are often grouped into three types:

- **Single-point features** are computed for each point individually and reflect its geometric or radiometric properties.
- **Neighborhood-based features** are derived from the local context around a point, helping to understand surface structure or noise characteristics.
- **Multi-temporal features** compare values across different time epochs to capture changes over time, which is essential in change detection tasks.

Tran et al. [2018] employed a combination of the above feature types. Among the single-point features, the height above the digital terrain model (DTM) was utilized. Neighborhood-based features included EchoRatio, ZRange, and ZRank, which are derived from the local neighborhood surrounding each point. Specifically, EchoRatio characterizes the vertical distribution of points within the neighborhood; ZRange quantifies the height difference between the lowest and highest points in a local 3D patch; and ZRank indicates the relative vertical position of the point of interest with respect to its neighbors. Additionally, geometric descriptors such as linearity, planarity, and omnivariance were used to describe the local surface shape around each point. Multi-temporal features were also incorporated by comparing information across different time steps. These included the number of points from a previous epoch within a specified search radius, height differences between epochs, and a stability metric (further explained in Section 2.6). Kharroubi et al. [2025] additionally introduced a distance uncertainty feature, which represents the confidence interval of M3C2 displacement, and a surface variation feature, calculated from the dispersion of normal vectors in the point's neighborhood.

Designing Features.

In an RF, different methods can be used to measure the importance of each feature. Understanding the importance of characteristics helps interpret the model and predictions. One common method is called *permutation feature importance*. In this method, the values of a feature are randomly shuffled across the rows. Then, the model's accuracy is checked again. If the accuracy drops, it means that the feature was important for making predictions (Breiman, 2001). Another method is *feature importance based on mean decrease in impurity*. This model calculates how much each feature reduces impurity (e.g., Gini) across all trees. If a feature helps reduce impurity across many trees, it is considered more important. However, this method can be biased toward features with more categories or continuous values.

Argumentation Use RF in This Research.

The advantages and disadvantages of using RF have been discussed above. It is selected for this research for several reasons:

1. It has shown promising results in previous studies.
2. It can be improved by adding or removing features, making it flexible for different datasets.
3. It works well with large datasets and handles high-dimensional data efficiently.
4. It is able to provide probability estimates for its predictions, which is useful for understanding the confidence of the model.

5. Due to bagging and the combination of decision trees, an RF reduces the risk of overfitting and increases generalization performance.

2.5. Sources of Uncertainty in ALS Data

Some challenges related to change detection methods were discussed in the previous section (Section 2.3). However, the characteristics of the dataset itself also have a significant impact on the effectiveness of these methods. As noted by Stilla and Xu [2023], the way data is acquired plays a key role in selecting an appropriate change detection approach. This section outlines important dataset characteristics and common sources of uncertainty that influence change detection results. Understanding these factors helps in developing a model that can better handle such limitations.

- **Point density and distribution**. The density and spatial distribution of points in a point cloud directly affect how well different scans can be aligned and compared (Stilla and Xu, 2023). When point clouds have inconsistent densities, it becomes more difficult to detect real changes. In contrast, when two point clouds have similar and evenly spread point distributions, it is easier to identify true environmental changes rather than artifacts from how the points were collected. de Gélis et al. [2021b] compared datasets with different densities and added noise. Their results showed that changes in point density had a larger effect on detection performance than noise. Specifically, depending on the method, the accuracy decreases around 6% to 14% due to lower density, while added noise decreases around 6% to 7%.
- **Information per point**. Semantic labels help in understanding changes at the object level (Stilla and Xu, 2023). Object-level analysis focuses on entire objects, such as cars, trees, or buildings, rather than looking at single points. Raw point clouds only contain geometric data (like X, Y, Z coordinates), but adding semantic labels allows each point to be linked to a specific type of object. This makes it possible to detect both geometric changes (e.g., movement) and attribute changes (e.g., an object changing its function or class).
- **Scan properties**. The properties of the scan itself can also affect change detection results. Kharroubi et al. [2022] highlight that factors such as the scan time, sensor position, weather conditions, sensor settings, sensing range, and background differences can all lead to real or false detections of change. Additionally, each light detection and ranging (LiDAR) sensor type has its own technical characteristics, such as scanning pattern, frequency, and beam divergence, which affect the quality and accuracy of the data, as well as the processing methods required.

Errors in point clouds create additional challenges for change detection. According to de Gélis et al. [2021b], noise makes it harder to clearly define the boundaries of detected changes. For example, if an extension is partially added to a roof, noise can make it unclear where exactly the change begins and ends.

Winiwarter et al. [2021] describe several types of measurement uncertainty. Each point in a point cloud is defined using polar coordinates (distance, azimuth, and polar angle), and each of these values has its own uncertainty. Laser beams do not hit just one point, they cover a small surface area. The reflected signal can come from anywhere within that area. When scanning tilted surfaces, like house walls, the footprint becomes larger, which increases the

error. Also, angular values are recorded in fixed steps, so there can be small errors when the actual value falls between these steps. The accuracy of distance measurements also depends on the scanning distance, the material of the object, and atmospheric conditions. Longer distances produce larger beam footprints, which leads to greater uncertainty.

In addition to measurement errors, object properties also affect the quality of the scan. For example, surfaces that reflect light poorly or in unpredictable ways, like water or glass, can cause gaps or noise in the point cloud. These surfaces may return weak or no laser signals, which can result in one scan detecting points while another does not. This can lead to false positive change detections (Kharroubi et al., 2022).

Many change detection methods struggle to detect small urban structural changes. Kharroubi et al. [2022] and Xu et al. [2015] highlight this by setting a filter for registration errors or false changes, like parked cars, which can also unintentionally exclude real changes, such as the addition of a dormer on a roof. Another reason they are mentioning is the low density of points representing small features (or other limitations in the dataset).

Large-scale point cloud datasets such as national height model of the Netherlands (AHN) require efficient data storage and access methods. van Oosterom et al. [2022] stress the importance of proper spatio-temporal data organization. Many valuable datasets are not fully used because of poor management, limited accessibility, and insufficient support from current software tools. Improving how these datasets are stored and accessed is essential for scalable and practical change detection.

2.6. Strategies to Mitigate Dataset Limitations

This section outlines potential strategies to address the challenges described in the previous sections.

Solving Occlusion.

Xu et al. [2015] address the problem of occlusion as follows. Related points in two epochs with a greater distance of 1 meter that lack any nearby points in one of the epochs within a horizontal plane are labelled as unknown. For walls, they label points as unknown if the neighboring roof has no change. They state this will also exclude lack of data due to pulse absorption by the surface material, e.g., water. But this method has some limitations, balconies and sun shades are far away from the roof and could be detected as changed and not as unknown.

Hebel et al. [2013] propose a method that uses a voxel structure to have a fast search operation on the fly while processing the airborne laser scanning (ALS) data. The reference point cloud dataset is organized in that voxel structure with wide cells (five times the average point-to-point distance) to reduce memory space, while maintaining efficient spatial indexing for queering. Two types of grid cells are then created, one representing the positions of the point cloud and another tracking the paths of laser beams traversing through each cell. For the last type, they applied Bresenham's 3D algorithm to calculate a raster line of that beam, identifying all the cells it traverses. Each voxel can have none, one, or multiple indices. With this information and the understanding of a point being a spatial extent rather than being a precise point (due to the laser's footprint, measurement uncertainty, and point

density), they modeled a gradual transition between empty, occupied, and unknown cells. Each cell is assigned belief masses of these states, following the Dempster–Shafer theory of evidence. Xiao et al. [2013] applied this to mobile laser scanning (MLS) and states it is an accurate method that distinguishes occlusion from real changes.

Liu et al. [2022] addressed occlusion detection in the context of detecting and reconstructing vehicle-related ground occlusions. In his approach, the ground surface is divided into a 2D grid, where each cell is marked as either occupied, empty, or boundary. The boundary is defined based on a concave shape, approximated using the α-shape algorithm. Within this α-shape boundary, there may be potential non-ground areas. In a separate step, cars are segmented, and based on the known 2D bounding boxes of these cars, the corresponding occluded raster cells are marked.

Stability Factor.

Tran et al. [2018] introduce the concept of a stability factor for each point in a light detection and ranging (LiDAR) dataset. This feature is calculated as the ratio of the number of neighboring points within a sphere in the other epoch to the total number of points within a vertical cylinder also in the other epoch, see Figure 2.11. They recommended using a search radius that is twice the average point spacing. Their findings showed that changed buildings and ground typically have a very low stability factor value, close to 0%, while unchanged buildings have a value of 100%. However, because vegetation is partially transparent to LiDAR, its stability factor values tend to vary and are less likely to be near 0% or 100%. de Gélis et al. [2021b] also use this stability factor as a feature for the random forest (RF) classifier, which gave more precise results.

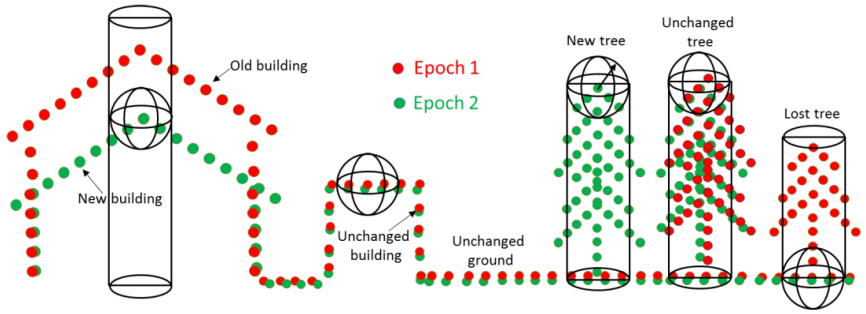


Figure 2.11.: Illustration of the search areas used for the stability factor feature. For several target points, both the 2D and 3D search areas are shown, along with the corresponding potential stability factor values. Adapted from Tran et al. [2018].

Highlighting Errors Accuracies.

Many factors can affect the quality of a LiDAR dataset. These include positional uncertainties, density, the precise alignment of the laser beam, errors from the laser device itself, the curvature of the local surface, and the geometry of the scanning process (Mayr et al., 2020). Lague et al. [2013] introduce the concept of the Level of Detection, which is the smallest detectable change in the data. Level of Detection is a spatially varying value that considers,

for every region in both epochs, the variances, point counts, and registration uncertainty (Winiwarter et al., 2021). Winiwarter et al. [2021] propose an improved version of Level of Detection. Unlike the original method, which assumes a constant registration error, their approach incorporates measurement and alignment uncertainties by using error propagation. Measurement uncertainties are derived from the laser scanner's angular and range measurements, while alignment uncertainties are calculated using an Iterative Closest Point algorithm in OPALS, which also generates a covariance matrix to represent these uncertainties.

According to Scaioni et al. [2018], LiDAR intensity can be used as an indicator of the quality of point cloud data. Their experiments show that there is a relationship between intensity and range noise. The intensity values are affected by several factors. One important factor is the material of the surface, especially its reflectivity and roughness (Kashani et al., 2015). Another group of factors relates to the scanning conditions. These include the power of the laser, the distance to the object (range), the angle at which the laser hits the surface (angle of incidence), the effect of the atmosphere (atmospheric transmittance), the spreading of the laser beam (beam divergence), and how sensitive the sensor is (detector responsivity) (Kashani et al., 2015).

Since dataset characteristics influence the results of change detection methods (see Section 2.5), metrics to measure these characteristics can be useful. Manders [2024] investigated the characteristics of two versions of national height model of the Netherlands (AHN), focusing on point density and point spacing, both of which affect computer algorithms. By `computer algorithms`, the author refers to the computer's ability to interpret and understand the structure of 3D objects. To assess point spacing, the author uses a Delaunay triangulation for all buildings in a specific area. Triangles larger than a certain threshold were considered gaps.

Size Datasets.

To be able to handle the large size of point clouds, Diaz et al. [2024b] developed a fast, space-filling curve-based method to calculate distances in epochs. They utilize a Morton curve to efficiently query neighboring points in the other epoch.

Another interesting approach for managing large point cloud datasets is the use of different levels of detail, as explained by Kharroubi et al. [2022]. This concept involves adjusting the level of detail depending on the specific needs of the task. When more detailed information is required, a higher level of detail is used by keeping more points, which works best for analyzing smaller areas. For larger areas, a lower level of detail with fewer points is used to reduce the amount of data that needs to be processed. However, the effect of using different levels of detail on the accuracy of change detection results has not yet been fully studied. This method has already been used in visualization tasks. van Oosterom et al. [2022] expand on this idea by introducing a continuous scale of detail. In their method, whether a point is displayed in a view depends on how far it is from the camera and its continuous detail value. This value is also used to organize and index the point cloud more efficiently. Liu [2022] interpret the continuous detail value in another way. In their work, this value represents how important a point is compared to other points in the dataset. This importance measure helps decide which points should be prioritized for processing or storage.

2.7. Conclusion Related Work

This section summarizes the main findings from the literature and clarifies the focus of this research.

There are many different methods for point cloud change detection. Choosing the right approach depends heavily on the goal of the research, as it influences the architecture of the method, the type of output, and the level of processing, whether at point-level, patch-level, or by converting the data to raster or models.

Many change detection methods struggle when working with real-world datasets due to their limitations. Common issues include misalignment between datasets, differences in point density, scanner-specific artifacts, and missing or incomplete data. These factors can significantly affect the performance and reliability of change detection.

The most widely used methods are those based on raster differencing (such as digital surface model (DSM) subtraction) and nearest-neighbour distance comparisons. These approaches are relatively simple to implement and are useful for initial exploration. However, they perform poorly in areas with occlusion or large variations in point density. They also require predefined thresholds to identify change, which limits flexibility.

Learning-based methods, especially using the random forest (RF) algorithm, have shown promising results. RF models are relatively easy to interpret and can be adapted by changing the input features. They also handle large datasets well, which is essential for processing point clouds. Another key benefit is their ability to output probability estimates for each point, which can be used to construct a certainty index. In addition, RF can model non-linear patterns in the data.

Some practical suggestions are available for designing a learning-based change detection method. One example is the stability factor, which compares the number of neighbouring points in a sphere to those in a cylinder around the target point. It is also useful to include both local and global features to improve classification accuracy.

Finally, the large size of point cloud datasets plays a role in performance. Efficient storage and querying techniques, such as using octrees or space-filling curves, can make processing tasks like change detection faster and more scalable.

3. Datasets

This chapter describes the datasets used in this research. Several key components are explained. The type of scanner used for each dataset is discussed, as this information is important for modelling occlusion in this research. In addition, the dataset requirements and results of the quality control are outlined, as they provide essential input for generating the synthetic scene. Further relevant details are also included to give more understanding and enable a proper comparison between the datasets in the end.

3.1. National Height Model of the Netherlands (AHN4 and AHN5)

This research focuses on the national height model of the Netherlands (AHN) datasets. This section explains the dataset and its applications, describes how the data is calibrated, and provides additional relevant information. It concludes with an overview of the scanners used to collect the dataset.

The AHN provides height data for the entire Netherlands. It contains classified point cloud data collected using airborne laser scanning (ALS). The AHN is made in cooperation with water authorities, provinces, and Rijkswaterstaat. It offers a detailed dataset with a high point density. There are five different datasets, collected in different years. In this research, only national height model of the Netherlands number 4 (AHN4) and national height model of the Netherlands number 5 (AHN5) are used. Both AHN4 and AHN5 have a systematic error of 5 cm and a random error (stochastic) of 5 cm. AHN4 and AHN5 have a minimum point density of 10–14 points per square meter, with higher densities (20–24 points/m^2) around Schiphol Airport. The data uses the stelsel van de rijksdriehoeksmeting (RD) coordinate system and the normaal Amsterdams peil (NAP) for height. Along with point data, information about the flightlines is also available. In addition, Geotiles provides RGB color values for each point in the AHN datasets. However, as noted by Nofulla [2023], these RGB values can be inaccurate because the images and point clouds were taken at different times. According to an AHN update, the goal is now to capture images and point data at the same time.

The light detection and ranging (LiDAR) data calibration begins by correcting rotational misalignments in the overlapping flight strips using three parameters: roll, pitch, and heading (or yaw). These represent rotations around the aircraft's front-to-back axis, side-to-side axis, and vertical axis, respectively. Adjusting these parameters helps ensure that features such as gable roofs align correctly between overlapping strips by correcting angular deviations. Once rotational corrections are applied, the data undergoes horizontal alignment, where the XY positions of points are refined by ensuring that corresponding features in different strips overlap on a 2D plane. Following horizontal alignment, vertical alignment focuses on height accuracy. Using overlap areas, height differences between flight strips at ground level are computed and adjusted. Besides that, it also involves comparing measured heights against

known reference surfaces (e.g., calibrated fields with precise NAP height values) to correct any systematic vertical offsets.

This combined approach, first correcting orientation via rotation parameters, then adjusting horizontal and vertical positions, results in a geometrically consistent, well-calibrated LiDAR point cloud.

Flightline and flight strip data for AHN4 and AHN5 are available in the AHN data room. Flightlines are stored as `LineString` or `MultiLineString` geometries. In certain areas, especially near coasts or islands, flightlines are interrupted by water, resulting in `MultiLineStrings`. Each flightline comes with metadata such as acquisition date and `LineID`. Flight strips are represented as `(multi)polygons`, which correspond to the convex hulls of the point clouds along each flightline. These polygons also contain metadata similar to that of flightlines. For AHN5, more detailed metadata was provided by Geodelta, including aircraft speed, scanner type, pulse repetition rate (PRR), scan rate, field of view (FOV), overlap, and points per square meter (ppm), reflecting Geodelta's role in performing AHN5 quality control.

Most of the AHN4 data was collected using the RIEGL VQ-1560i scanner. A smaller part of the dataset was captured with the newer VQ-1560II model. Both scanners use the same scanning pattern, as described in the manufacturer's manual (Figure 3.1). The scanner operates with two laser channels. In the (x,y)-plane, channel 1 scans from the lower left to the upper right, and channel 2 scans from the lower right to the upper left. The angle between the two channels is 14° across-track. Additionally, each channel is tilted either forward or backward by 8°. However, the manual does not specify which channel is tilted forward and which one is tilted backward. This detail was interpreted from Figure 3.3, which shows an aircraft on its flight path (black line) for capturing AHN4. The green areas represent points that have already been collected, while the grey areas show the locations that are yet to be scanned. The crossing point of both channels appears to lie to the right of the flightline. This suggests that channel 1 is tilted backward and channel 2 is tilted forward. Based on this interpretation, the scanning geometry shown in Figure 3.2 was created.

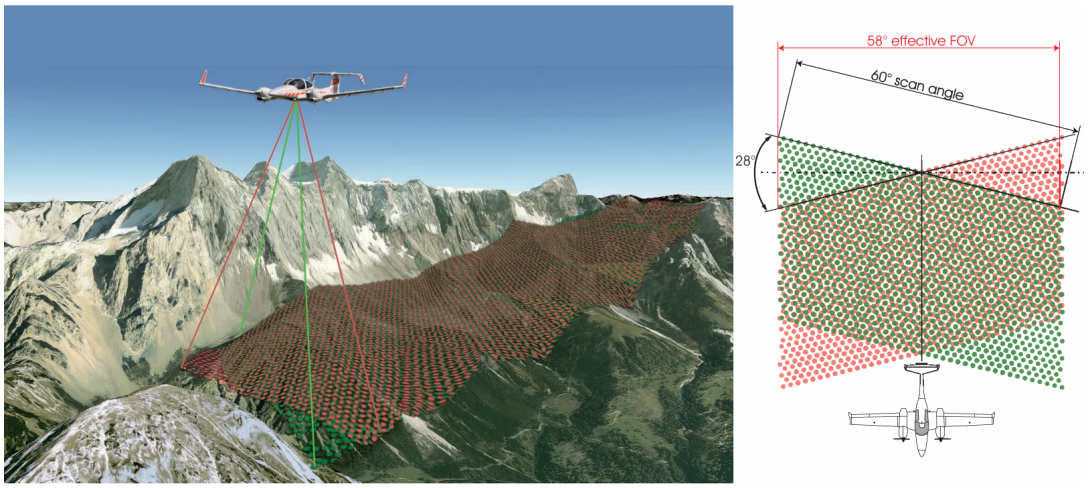
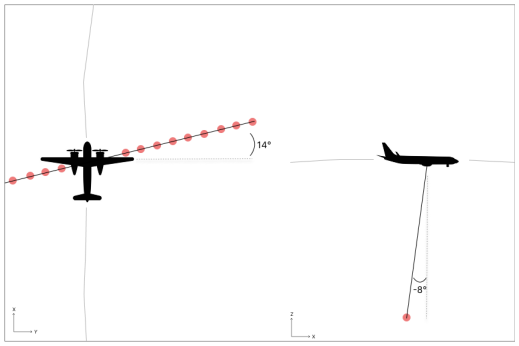
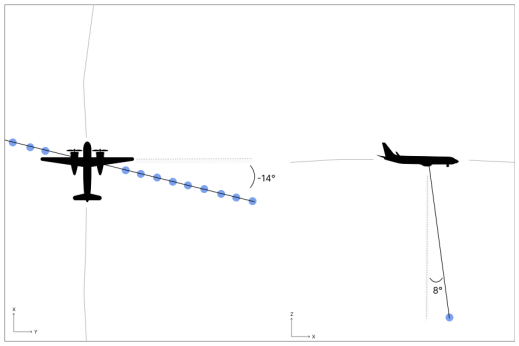


Figure 3.1.: Scanning pattern from the official RIEGL manual. Source figure: RIEGL Laser Measurement Systems GmbH [2017].



(a) Backward-tilted scan channel with a 14° across-track angle and 8° backward tilt.



(b) Forward-tilted scan channel with a 14° across-track angle and 8° forward tilt.

Figure 3.2.: Custom scan line visualizations for the RIEGL VQ-1560i, VQ-1560i-DW, and VQ-1560II scanners.

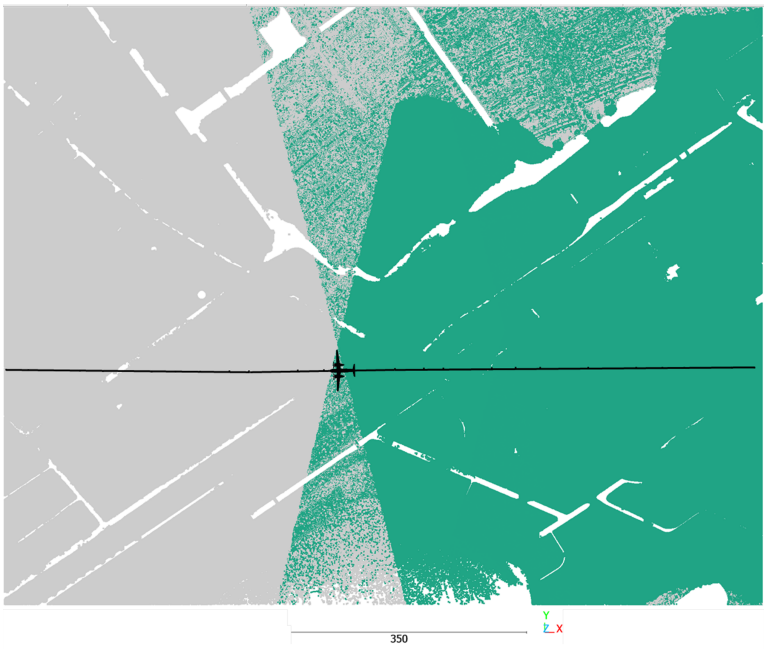


Figure 3.3.: The airplane is shown along its trajectory (black line). Green points represent the points already captured by the airplane up to its current position. Gray points are points yet to be captured, either on the current flightline or on another flightline. Note: There is a spatial offset between the aircraft's actual position and the cross-points of the scan channels.

In contrast, all AHN5 data was collected using the Leica CityMapper-2, which uses an oblique LiDAR scanning system. This system scans in a cone-shaped pattern, shown in Figure 3.4. To also visualize this in the AHN5 data, see Figure 3.5.

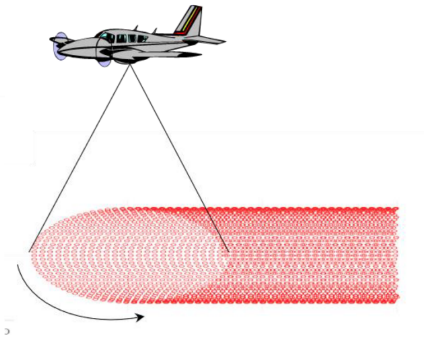


Figure 3.4.: Conical scan pattern used in oblique LiDAR systems like the Leica CityMapper-2 in AHN5. Source figure: Bacher [2022].

3.2. Rotterdam ALS Dataset

The Rotterdam airborne laser scanning (ALS) datasets cover the entire municipality of Rotterdam. These datasets are not publicly available and can be obtained through Geodelta. There are datasets available for the years 2021, 2022, 2023, and 2024. This research uses only the 2023 and 2024 datasets.

Both datasets require a minimum point density of 30 points per square meter. This density could not always be reached in the city center due to the presence of tall buildings. To improve coverage in this area, extra flightlines were added. All data is provided in the stelsel van de rijksdriehoeksmeting (RD) and normaal Amsterdams peil (NAP) coordinate systems.

A calibration flight was carried out before the data collection to set and verify the system parameters of the scanner. Quality control was performed in several ways. For absolute horizontal accuracy, ground control points were used. These are white areas on the ground that were compared to the intensity values of the point cloud. For relative horizontal accuracy, the gable roof method was applied, as described earlier in the context of national height model of the Netherlands (AHN) alignment. To check the absolute height accuracy, ground control points were compared with the elevation values from the point cloud. For relative height accuracy, height difference grids were created in overlapping flightline areas. These grids were compared to check for consistency between flightlines. The results of these checks are shown in Table 3.1.

The datasets include general point classifications such as ground, non-ground, and unclassified. They also contain metadata about flightlines and flightstrips, which are relevant for this study.

Both datasets were collected using the Leica CityMapper-2 sensor, the same system used for national height model of the Netherlands number 5 (AHN5). A visualisation of this scanner is shown in Figure 3.4.

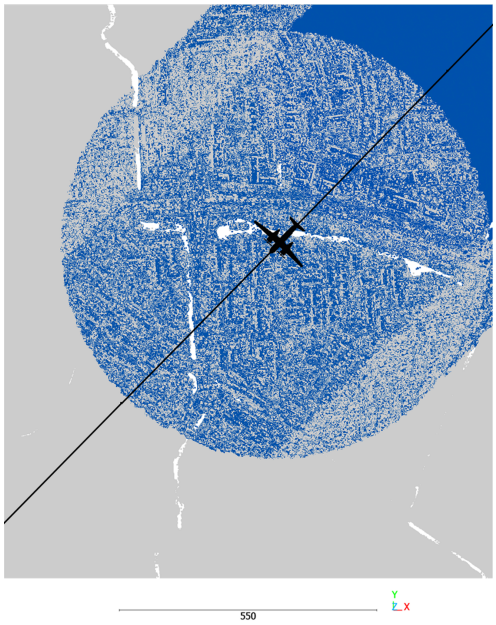


Figure 3.5.: The airplane is shown along its trajectory path (black line). Blue points indicate the points already captured by the airplane up to its current position. Gray points represent points still to be captured, either on the current flightline or on another one.

Table 3.1.: Accuracy Statistics of the Rotterdam Datasets (2023 and 2024)

Metric	**2023 Dataset**	**2024 Dataset**
Maximum individual vertical deviation (z)	0.050 m	0.053 m
Standard deviation in z (σ_z)	0.010 m	0.014 m
Maximum individual horizontal deviation (d_{xy})	0.050 m	0.093 m
Standard deviation in d_{xy} (σ_{xy})	0.019 m	0.017 m
Maximum systematic horizontal error (d_{xy})	0.061 m	0.152 m

4. Research Methodology

This chapter describes the methodology used to design and evaluate a point-based building change detection method for ALS datasets. A key part of the method is the introduction of a certainty index, which takes occlusion into account when estimating how reliable each detected change is.

The core of the method is developed in Phase 1 and Phase 2. These are framed by two other stages: an initial pre-phase and a final test phase.

- **Pre Phase** (Section 4.1): This phase focuses on understanding the types of structural changes present in the AHN datasets, as well as exploring the limitations of basic geometric comparison techniques.
- **Phase 1** (Section 4.2): In this phase, synthetic ALS point clouds are generated. These simulations include different types of changes and acquisition conditions, and allow for automatic labelling of changed and unchanged points.
- **Phase 2** (Section 4.3): This phase covers the development of the actual detection method using an random forest (RF) model. It also includes occlusion detection and how this is used to calculate a certainty score for each predicted change.
- **Test Phase** (Section 4.4): The final phase applies the developed method to real-world datasets (the national height model of the Netherlands number 4 (AHN4), the AHN5, and Rotterdam). This phase includes the practical steps needed to prepare and evaluate the method on real data.

The chapter concludes with a brief summary of the methodology process in Section 4.5.

4.1. Pre Phase: Data Understanding and Change Definition

The goal of this phase is to explore how building-related changes between national height model of the Netherlands number 4 (AHN4) and national height model of the Netherlands number 5 (AHN5) can be identified, and to understand which characteristics of the data influence change detection. This step is essential for defining what constitutes a meaningful change, identifying data limitations, and guiding the feature design of the detection method.

Visual Exploration and Area Selection

To support this, five urban areas were selected for closer inspection, each containing various building types and change scenarios. These areas were downloaded from GeoTiles [van Natijne and Optical and Laser Remote Sensing group TU Delft, 2025], which provide subdivided national height model of the Netherlands (AHN) tiles for more efficient processing. The selected areas are shown in Figure 4.1.

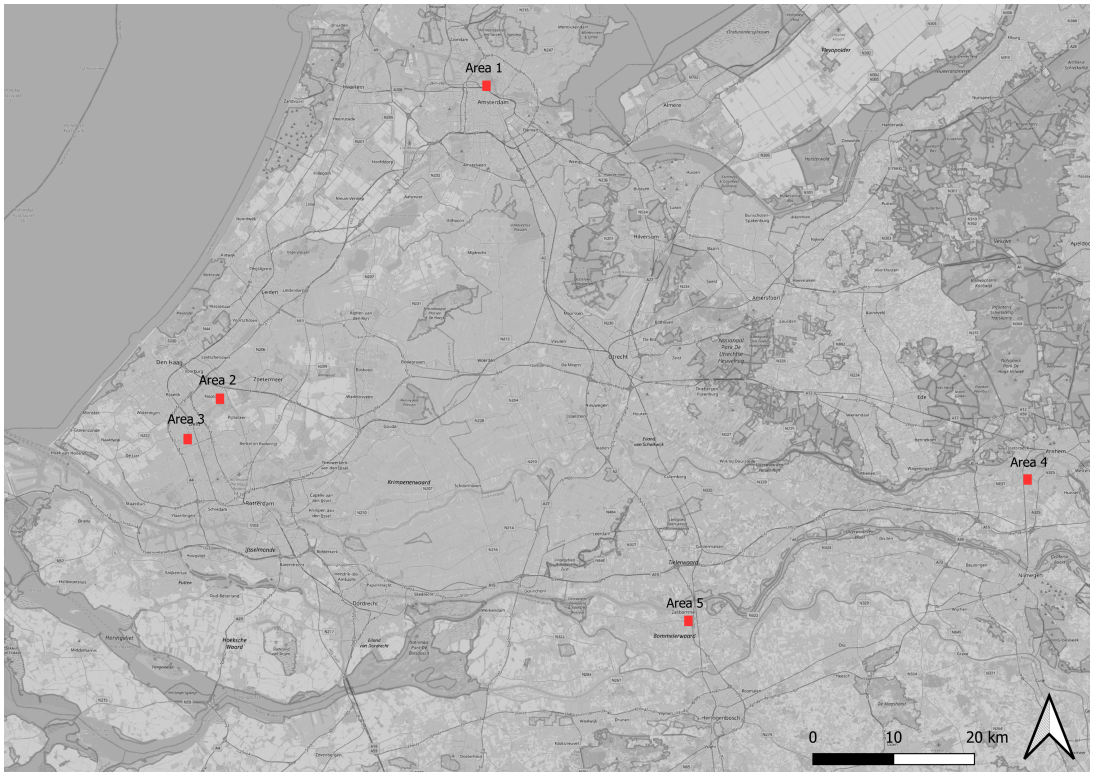


Figure 4.1.: Areas of interest in AHN4 and AHN5 for the exploration phase. Background map from © OpenStreetMap contributors, ODbL.

To visually explore the point clouds, the PotreeDesktop viewer of [Schütz et al., 2020] was used, see Figure 4.2. This enabled a detailed inspection of point attributes such as classification, height, and intensity across both epochs.

Initial Change Detection: Nearest-Neighbor Methods

To evaluate simple change detection approaches, both 2D and 3D nearest neighbor distance metrics were applied. This helped assess how small changes (e.g., dormers, solar panels) appear in point-level comparisons and where traditional methods fall short.

Two spatial indexing methods were tested: SciPy's `KDTree` for exact nearest-neighbor search, and the Morton curve algorithm from Diaz et al. [2024b] for approximate nearest-neighbor

Figure 4.2.: PotreeDesktop visualization of Area 5, with point clouds colored using aerial imagery.

search based on spatial locality. For each point in AHN5, the nearest point in AHN4 was identified, and both 3D Euclidean distance and height difference (ΔZ) were recorded.

Definition of Change Types

Through this analysis and a review of related work, a set of relevant building change types was defined to be included in the synthetic training dataset. These represent common structural modifications observed in Dutch urban areas and include:

1. Addition or removal of balconies and dormers,
2. Modifications to roof type and height,
3. Horizontal and vertical extensions of buildings,
4. Installation of solar panels with varying thicknesses (4–16 cm),
5. Addition or removal of buildings (detached and row houses),
6. Geometric variations in building shape to increase scene diversity.

These change types were later used in the simulation of synthetic urban environments and informed the feature design for the random forest (RF) classifier developed in Phase 2. It is the goal of the developed method to detect these changes.

4.2. Phase 1: Synthetic Point Cloud Generation

The goal of this phase is to create synthetic, labelled airborne laser scanning (ALS) point cloud datasets that closely resemble real-world data. These datasets will be used for training, validating, and testing the change detection model. Several datasets are generated to cover these purposes.

This step is important because in Phase 2, an random forest (RF) will be used for the change detection method, and this method needs labelled data. Some labelled data is already available, but it is limited and does not cover all types of changes that are studied in this research.

In addition, by creating synthetic data, it is possible to control the settings of the scanner and other parameters. This helps to make sure that the algorithm is trained well and can work with the national height model of the Netherlands (AHN) and Rotterdam datasets.

Scene Design and Building Changes

Synthetic urban scenes are created using Blender, where 3D building models are generated or imported from the 3D building models of the Netherlands (3DBAG) dataset. The scene includes a variety of house types and predefined structural changes. The buildings are placed on a flat ground plane, with no additional landscape elements.

Simulation ALS Point Cloud Datasets with Helios++

Helios++ is used to simulate ALS data over the synthetic scenes. The tool allows full control over scanner parameters such as scan angle, frequency, aircraft speed, and pulse repetition rate. Two types of scenes are created:

- A large urban scene designed to mimic national height model of the Netherlands number 4 (AHN4) (epoch 1, reference point cloud dataset) and national height model of the Netherlands number 5 (AHN5) (epoch 2, target point cloud dataset). The scanner configurations are adapted based on real metadata from AHN4 and AHN5, ensuring that the synthetic datasets reflect the characteristics of real ALS surveys. Parameters such as platform height, point density, and beam divergence are fine-tuned using visual comparisons with real data.
- A smaller test scene, used to generate multiple variations in acquisition settings. To evaluate the algorithm's robustness under different data conditions, five synthetic scene variants are simulated. These differ in scanner models, scanning patterns, pulse repetition rate (PRR), and flightline configurations. This variation allows investigation of how uncertainty in acquisition affects detection performance.

Automatic Labelling the Synthetic Data

Once the simulated point clouds are generated, the next step is automatic labelling of changes. This is done by matching each point to the surface of the 3D models. The labelling process includes:

- Determining whether a point belongs to a roof or wall,
- Classifying the type of change (added, removed, modified, or unchanged),
- Assigning a change category (e.g., dormer, solar panel).

House-level labelling is also performed: if at least one point within a house is changed, the entire house is labelled accordingly. Ground points are generated and classified separately to support height-based features.

Output and Applications

The final output of this phase is a set of synthetic, fully labelled point cloud datasets. These datasets are used for:

- Training and validating the change detection algorithm,
- Comparing the varied acquisition scenarios,
- Supporting error analysis and performance interpretation based on ground truth.

This controlled simulation and labelling approach enables systematic evaluation of algorithm performance under known conditions.

4.3. Phase 2: Change Detection Method Design

This phase focuses on developing a method that predicts, for each point in a point cloud, whether it has changed between two epochs. The method is trained using the synthetic, labelled datasets created in Phase 1.

Based on the literature discussed in Section 2.3, the following principles are important when developing a point cloud change detection method:

1. Avoid converting point clouds into raster or voxel formats, as this can lead to a loss of geometric detail.
2. Include both global and local spatial information.
3. Learning-based methods tend to perform well.
4. The method should work with different input datasets, making it more robust.

Although deep learning models can give high accuracy, they are not used here. These models need large labelled datasets, are harder to interpret, and require a lot of computing power. These issues make them less practical for large-scale datasets like national height model of the Netherlands (AHN). Instead, an random forest (RF) model is used, with an additional step to handle occlusion for better reliability.

The RF model works directly on point cloud data and does not convert it to raster form. It is trained using labelled points and uses input features that include both local and global spatial patterns, satisfying the first three design principles. The method also follows the fourth principle by using several different synthetic datasets and by including features that capture uncertainty (such as distance to the flightline).

The method produces two outputs for each point: a classification label (”changed” or ”unchanged”) and a certainty score. The certainty score shows how confident the method is in its prediction and whether the point was visible in both scans. Occlusion is handled by checking the visibility of each point in both epochs. If a point is not visible, the prediction may be less reliable. The certainty score combines the predicted probability of the RF model (see Equation 2.2) with occlusion information. The certainty score ranges from 0 to 1: values close to 0 indicate high certainty the point is unchanged, while values close to 1 indicate high certainty it has changed. A score around 0.5 suggests a high level of uncertainty. This concept is shown in Figure 4.3.

Figure 4.3.: Horizontal color map showing distinction the certainty score.

This method is developed in two steps, explained below. Step 1 estimates the probability that each point has changed using an RF model. Step 2 improves how uncertainty is handled by incorporating occlusion information.

Step 1: Train Random Forest to Detect Point-level Changes

In the first step, an RF classifier is used to estimate the probability that each point has changed. The model uses three types of input features:

- **Intrinsic features:** Describe properties of the point itself, such as the distance to the flightline.
- **Inter-epoch features:** Capture changes between the two epochs, such as differences in height.
- **Dataset-level features:** Include general scan properties like point density, beam divergence, or scanner accuracy.

Redundant or irrelevant features are removed to improve model performance. The classifier then gives a probability value for each point, indicating how likely the point has changed.

Step 2: Integrate Occlusion to certainty Score

The second step improves the interpretation of the results by considering occlusion. Occlusion occurs when a point is not visible in one of the scans, which can make change detection less reliable. Each point is assigned to one of three occlusion classes (see Figure 4.4):

- **Class 0 (Visible):** Point is visible in both epochs.
- **Class 1 (Static occluded):** Point is blocked by a stable object present in both scans.
- **Class 2 (Dynamic occluded):** Point is blocked because of a structural change, such as a removed building part.

The occlusion class is determined using ray tracing, based on aircraft positions and the reference point cloud dataset. Two visibility checks are done for each point, this is described in Section 5.6.

Two methods are tested to include occlusion in the model:

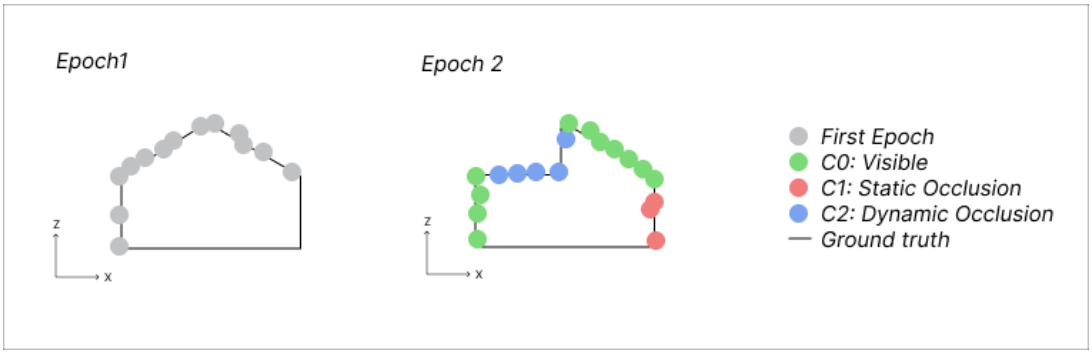


Figure 4.4.: Types of occlusion. **Red points** are static occluded (hidden in both epochs). **Blue points** are dynamic occluded (hidden due to change). **Green points** are visible in both epochs.

- Incorporating occlusion as an additional input feature for a second RF model. This secondary model receives the initial probability scores from the first RF model, both for individual points and their neighbours, and combines them with occlusion information to produce a refined certainty score that accounts for visibility constraints.
- Post-processing the probability scores to generate a certainty score based on the occlusion status of each point. Examples of this approach include:
  - Decreasing the certainty score for points that are statically occluded, reflecting higher uncertainty in those predictions.
  - Retaining the original score for dynamically occluded points.

Validation and Evaluation

The method results are evaluated using both numerical metrics and visual inspection. For each class (changed or unchanged), the following are reported:

- **Precision:** The proportion of predicted changes that are correct.
- **Recall:** The proportion of actual changes that are correctly detected.
- **Error by change type:** Accuracy is also reported for different kinds of changes, such as dormer additions, solar panels, or other changes.
- **Visual Assessment:** Precision and recall only show whether points are classified as changed or unchanged. They do not show how confident the method is. Visual inspection allows direct analysis of the certainty scores. Visual assessment is also useful because changes often affect groups of points, not just single ones. For example, if a dormer affects 20 points and 18 are correctly classified, the change is still clearly visible in the certainty map. In such cases, visual inspection provides useful information that standard metrics might miss.

The effect of different synthetic dataset settings is also evaluated. This shows which scanning conditions or point cloud properties most influence accuracy.

Finally, the impact of occlusion on change detection is also visually assessed by checking whether points were assigned the correct occlusion class.

Exploratory Step: House-Level Change Detection (Discarded)

An early version of the method tested change detection at the building (house) level. This aimed to reduce processing time by first filtering out unchanged areas. The rule-based approach checked for differences in footprint area, maximum height, and roof shape using concave hulls, height comparisons, and histogram correlation. Details of this approach are in Appendix F.

While this method worked well for large changes (e.g., new floors or large roof modifications), it had limitations. The concave hull method was sensitive to point density and often misclassified unchanged houses. Smaller changes, like dormers or solar panels, were often missed unless they had a large effect on height or shape. Histogram-based methods also lacked consistent thresholds due to differences in data quality and building complexity.

Because of these issues, this step was removed from the final method.

4.4. Test Phase: Testing on Real-World Datasets (AHN and Rotterdam)

In this phase, the developed change detection algorithm is applied to real-world datasets, specifically national height model of the Netherlands number 4 (AHN4), national height model of the Netherlands number 5 (AHN5), and a separate dataset covering the urban area of Rotterdam. These datasets were selected due to their high spatial resolution, availability, and widespread use in various applications in the Netherlands. Before applying the method, the datasets must be properly registered, as accurate alignment between epochs is crucial for reliable change detection. Since this research focuses solely on detecting changes, not on point cloud registration, this preprocessing step is necessary but not part of the method itself.

This phase is included to acknowledge that certain preparatory steps are necessary before applying the algorithm to real-world datasets.

To focus specifically on building changes, a filtering step is used to isolate the relevant points. Because the classification between datasets differs (see Chapter 3), the scene is first rasterized. For AHN4 and AHN5, any raster cell that contains at least one building-classified point is marked as a building cell. All points in the target point cloud dataset that fall within these cells are then checked to see whether they differ from the reference point cloud dataset. Since the Rotterdam datasets do not contain building classifications, the rasterized scene from the corresponding national height model of the Netherlands (AHN) dataset is overlaid onto the Rotterdam point cloud. All points within the overlapping building cells are then included in the analysis for the target point cloud dataset. This process is explained in more detail in Section 5.8.

After filtering, the change detection method is applied to the selected areas. As there are no ground truth change labels available for the AHN or Rotterdam datasets, a qualitative evaluation is performed. This involves visually inspecting the results to assess whether the algorithm correctly identifies structural building changes.

Finally, the impact of occlusion on the certainty score is also examined, similar to the process used for the synthetic data. This is done by visually checking whether points have been assigned the correct occlusion type, based on their visibility in each epoch.

4.5. Conclusion Methodology

The methodology consists of four main phases. The first phase explores the national height model of the Netherlands number 4 (AHN4) and national height model of the Netherlands number 5 (AHN5) datasets to understand how building changes appear in point clouds and to compare different nearest neighbour methods.

In the second phase, synthetic point cloud datasets are created by simulating airborne laser scanning (ALS) data using Helios++. These datasets are automatically labelled to mark changed and unchanged areas, allowing training and validation of the change detection model. Six bitemporal datasets are generated: one designed to mimic AHN4 and AHN5, and five with different scanning parameters.

The third phase focuses on developing a point-level change detection algorithm based on an random forest (RF) model. This model uses intrinsic, inter-epoch, and dataset-level features while maintaining the geometric fidelity of the input point clouds. The RF model outputs a probability score for each point, representing the likelihood of change. To enhance the reliability of these predictions, occlusion information is integrated to adjust the probability score into a certainty score. Two strategies are evaluated for incorporating occlusion: the first introduces a second RF model in which occlusion is included as an additional input feature; the second applies a post-processing step that modifies the probability score based on the occlusion status of each point.

Finally, in the fourth phase, the trained method is applied to real-world datasets, including AHN4, AHN5, and the Rotterdam ALS datasets from 2023 and 2024. Although no ground truth data is available, visual evaluation will be conducted to determine whether the method effectively detects structural changes in buildings. The use of occlusion information will also be verified by comparing the visibility of building facades between epochs with the occlusion features assigned to points.

The below figure shows an overview of the methodology.

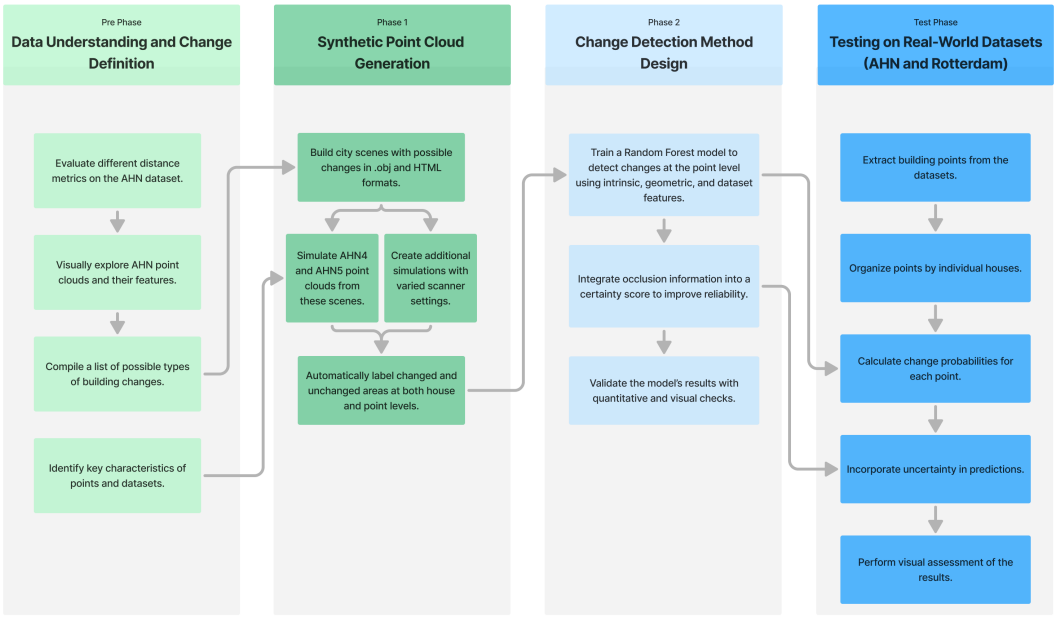


Figure 4.5.: Overview of the research workflow, including methodological design, synthetic data generation, and final testing on real-world datasets.

5. Implementation

This chapter describes how the proposed method was implemented, following the pipeline shown in Figure 5.1. The process consists of three key steps that transform raw bitemporal ALS data into point-level change detection outputs enriched with a certainty index. Each step is detailed in one or more of the following sections.

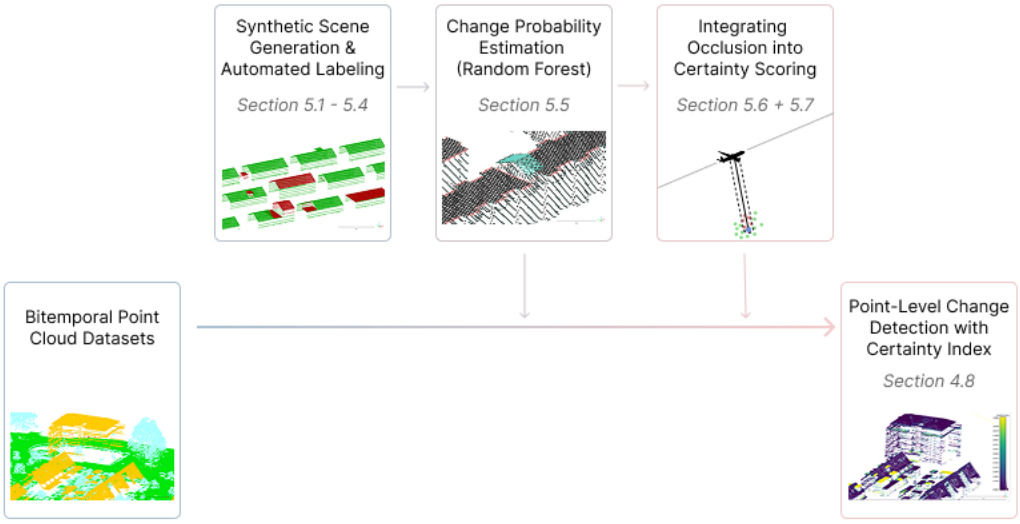


Figure 5.1.: Overview of the process.

The first step involves the creation of a synthetic labelled dataset to train and validate the change detection model. This includes four sections. (1) A visual exploration of the national height model of the Netherlands (AHN) datasets (Section 5.1). (2) The design of a 3D synthetic urban scene (Section 5.2). (3) Simulation of ALS point clouds over this scene (Section 5.3). (4) Automatic labelling of changed and unchanged regions in the point cloud (Section 5.4).

Next, in the change probability estimation step, an RF classifier is used to predict the likelihood of change at the point level, based on a combination of geometric, contextual, and temporal features (Section 5.5).

The third step focuses on occlusion-aware certainty modeling. This involves first identifying where occlusions occur in the point clouds (Section 5.6) and then incorporating this visibility information into the model's certainty score (Section 5.7).

Finally, the trained method is applied to real-world ALS datasets, the AHN4, AHN5, and the Rotterdam dataset, to assess the performance and visualize the outcomes (Section 5.8).

The chapter concludes with a summary of the implementation process in Section 5.9.

5.1. Exploratory Analysis Implementation Details

This section describes the technical implementation of the exploratory analysis steps performed on the national height model of the Netherlands (AHN) datasets. These steps were designed to support the data understanding and change definition tasks discussed in Section 4.1.

The AHN tiles were processed using GeoTiles [van Natijne and Optical and Laser Remote Sensing group TU Delft, 2025], which subdivide standard AHN tiles into 25 smaller GeoTiles. These smaller tiles allowed for more efficient loading and processing during both visual exploration and automated analysis.

Point cloud visualization was carried out using PotreeDesktop [Schütz et al., 2020]. The tool was used to visually inspect areas of interest for anomalies, such as occlusion, misclassification, or gaps in data. AHN point clouds were converted to Potree-compatible format and overlaid with orthophotos to assist with manual inspection.

The 2D distance between each point and its corresponding flightline was calculated using the NetTopologySuite library. Each point in the dataset contains a `source id`, which corresponds to the `flightline id`. Flightlines were modeled as LineString or MultiLineString geometries, and distances were computed using the library's `Distance` method.

Intra-point distance was computed by finding the average distance from each point to its k nearest neighbors, where $k = 8$, based on the recommendation of Diaz et al. [2024a]. This metric was later used to define neighborhood-based features and thresholding strategies in change detection.

To compare the spatial location of points between national height model of the Netherlands number 4 (AHN4) and national height model of the Netherlands number 5 (AHN5), two nearest neighbor search strategies were implemented:

- **KDTree:** Provided by the `NetTopologySuite.Index` library, for efficient exact nearest-neighbor lookup in 2D or 3D space.
- **Morton Curve:** An approximate indexing method based on space-filling curves, implemented following Diaz et al. [2024b]. For each point in AHN5, the $N = 100$ closest candidates were found along the Morton curve, and the point with the smallest 3D Euclidean distance was selected as the nearest neighbor.

Both 3D and height-only (ΔZ) distances were stored as attributes for further analysis and visualization. The calculated distances were stored in the `GPS`-time of the point in a structured format to allow overlay and filtering during Potree visualization and later training. This step supported the manual validation of detected changes and the design of training labels and features in subsequent phases.

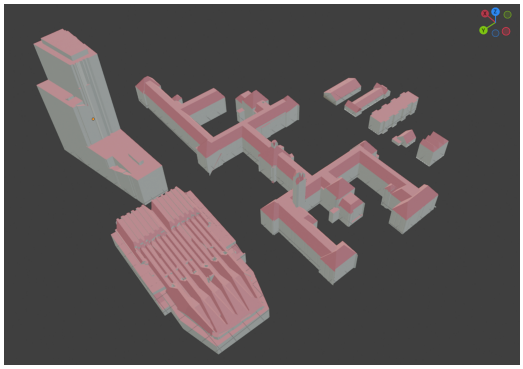
5.2. Designing the Synthetic Urban Scene

In this section, it is explained how the city scene is created. This step is important for this research because:

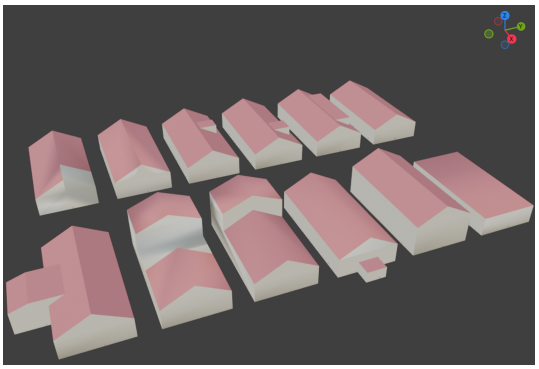
- the scene will be used as input for `Helios++`, which will simulate the data needed to train the algorithm,
- the scene can be changed in a controlled way, so the author knows exactly what has changed,
- the building surfaces (faces) are labelled so the labelling process can be done automatically,
- the scene includes challenges that most change detection methods usually struggle with,
- different types of houses and different kinds of building changes are included.

To ensure the synthetic dataset is suitable for training and testing the algorithm, it is essential to create a realistic city scene. This is accomplished using Blender. First, several houses are downloaded from 3D building models of the Netherlands (3DBAG). These include terraced houses, semi-detached houses, detached houses, apartment blocks, and more complex structures, such as the Aula and the Architecture building at TU Delft, as shown in Figure 5.2a. To ensure the input is compatible with `Helios++`, the faces of the houses must be valid. For example, no three or more points in a polygon should be collinear. Additionally, faces with five vertices were not recognized by the `Helios++` 'aircraft', so these planes were divided into multiple faces to ensure the simulation runs smoothly.

To begin, the complexity of the first four types of houses is reduced by simplifying their geometry, such as removing unnecessary vertices. This makes the next step, modifying the houses, much easier. The adjustments made to one type of house are shown in Figure 5.2b. Similar changes are then applied to the other house types.



(a) Various types of houses included in the city scene.



(b) Various modifications made to the houses in the city scene.

Figure 5.2.: Different house types and modifications made in the city scene.

As described in Section 4.2, two synthetic scenes are created. One is a larger city that is designed to simulate the structure of national height model of the Netherlands number

4 (AHN4) and national height model of the Netherlands number 5 (AHN5). The other is a smaller scene, used to create multiple versions with different scanner settings.

Larger Scene

The larger scene is built using several types of houses. This setup, shown in Figure 5.3, contains seven different groups. Each group is made to test a specific situation:

- **Group A:** Tests how detection works at different distances from the flightline. All houses in this group have the same change in one row from the reference point cloud dataset to the target point cloud dataset.
- **Groups B and F:** Represent areas with many closely placed houses.
- **Group C:** Contains houses placed at different angles.
- **Group D:** Includes more complex building shapes.
- **Group E:** Contains flats (apartment buildings) that cause occlusion.
- **Group G:** Features large, detached houses.

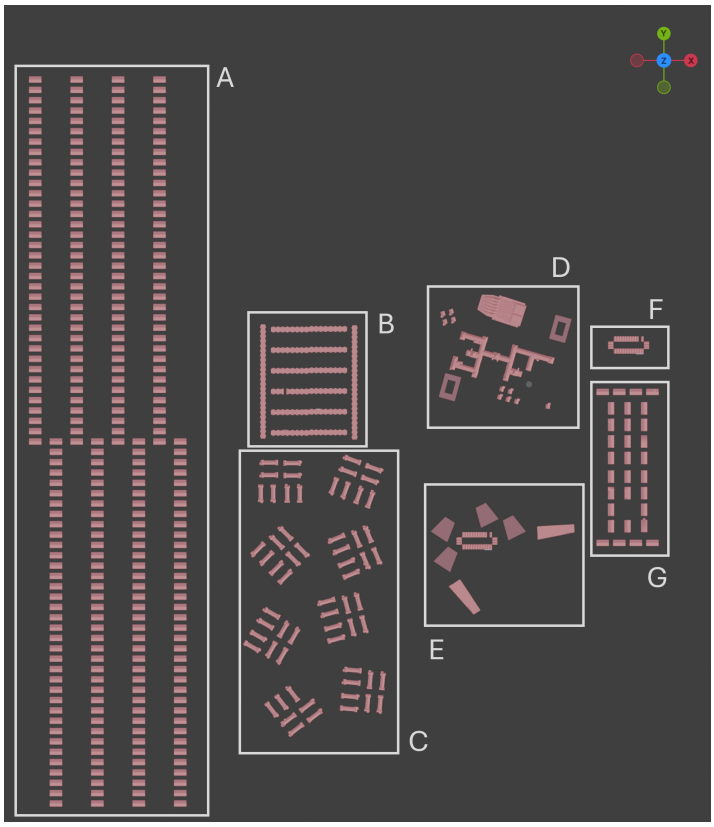


Figure 5.3.: The structure of the city scene. Each group of houses is designed to test a specific aspect of the change detection method.

Figure 5.3 shows the setup for the first epoch. For the second epoch, several types of changes are applied to the buildings, including new structures, removals, and other modifications. Most of the buildings in the scene are changed, which makes the dataset imbalanced (more changed than unchanged buildings) when doing the houselevel change detection (which is, in the end, discarded). To fix this, extra copies of unchanged buildings are added. These copies include different types of buildings and unchanged versions with added dormers and balconies. They are also rotated and placed in new locations to increase variation. This helps create a better balance between the changed and unchanged buildings in the dataset.

Smaller Scene

Figure 5.4 shows the smaller scene. This scene includes all types of buildings and changes, just like the larger one, but is made smaller to reduce processing time. In the next section, it will be explained how different scanning parameters are used to create multiple versions of this scene.

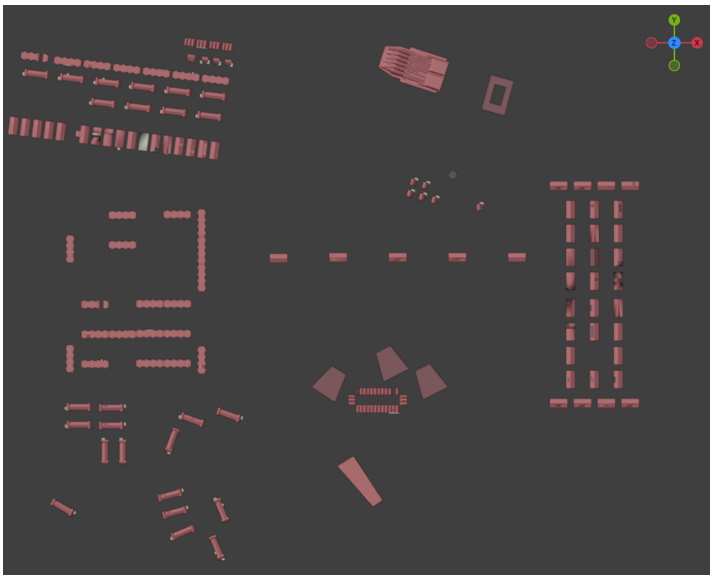


Figure 5.4.: Smaller scene used for generating variations in acquisition settings.

Finally, all scenes are exported in the `.obj` format, making them compatible with `Helios++`, which is used in the next phase of the workflow. The changes made to houses are listed in Section 4.2.

In addition to the houses, a flat surface is added beneath them to represent the ground. The scene does not include other elements such as trees or roads; it consists only of buildings and the ground.

5.3. Simulating ALS Scans from Synthetic Data

In this section, it is explained how a point cloud is created from the city scene, which is described in Section 5.2. This step is important for this research because:

- the data is needed to train the algorithm,
- the exact locations of the changes are known,
- the bigger scene is designed to mimic the national height model of the Netherlands number 4 (AHN4) and national height model of the Netherlands number 5 (AHN5) datasets as closely as possible,
- the smaller scene incorporates different types of scanners and dataset characteristics to make sure the algorithm works well in many different situations.

In this section, different datasets are created. The first area is a larger scene designed to match the AHN4 and AHN5 datasets. The second area is smaller and is used to test different scanner types and various settings. After explaining these areas, the implementation of `Helios++` will be described.

Mimic Synthetic Scene with AHN4 and AHN5

To make sure the synthetic scene matches AHN4 and AHN5, the following scanner settings can be adjusted: the height of the aircraft, the spacing between flightlines, the speed of the aircraft, the scan angle, the pulse repetition rate (PRR), and the scan rate.

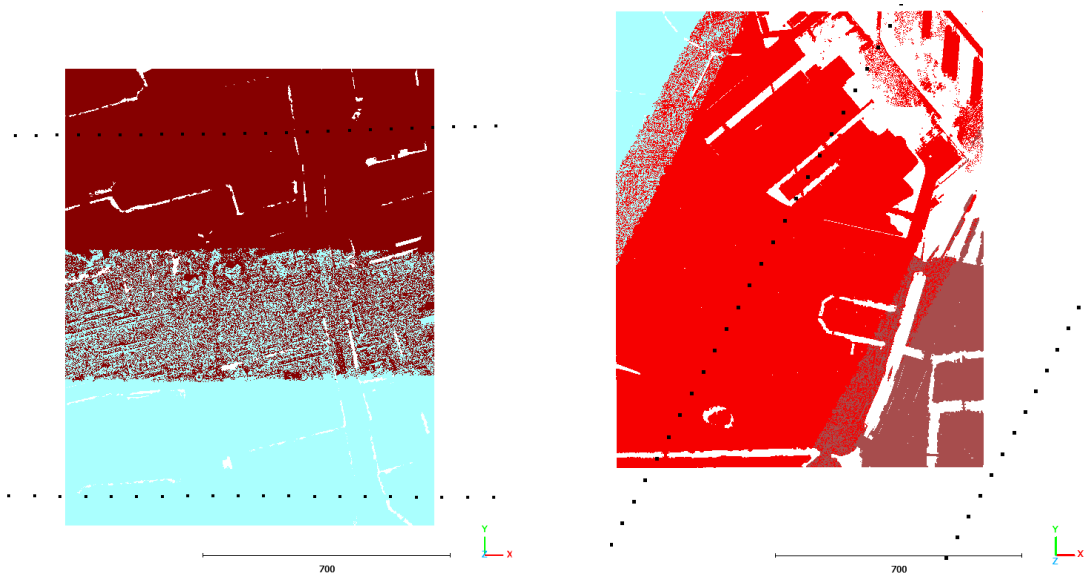
To check if the synthetic dataset matches the real data, a part of the Netherlands is downloaded from both the national height model of the Netherlands (AHN) and 3D building models of the Netherlands (3DBAG) datasets. Two areas are selected, Area 1 and Area 3, as shown in Figure 4.1. Area 1 is used for AHN5 because the edges of the flight strips are straight, making it easier to compare the scan settings. Area 3 is used for AHN4 for the same reason. This can be seen in Figure 5.5.

Using 3DBAG and flightline data, synthetic versions of AHN4 and AHN5 were created for the same geographic location. These synthetic point clouds were visually compared to the real datasets, with a focus on evaluating whether occluded (hidden) building walls align correctly. This visual inspection also allowed for manual adjustment of simulation parameters, as described in the sections below.

The ground surface is not included in the synthetic scene, and the 3DBAG buildings already match the real AHN data in both height and coordinate system. As a result, the synthetic and real point clouds are spatially aligned without the need for additional transformations.

For both AHN4 and AHN5, the **scanner type** is recorded in the flight strip metadata, as explained in Section 3.1. The Leica scanner used in AHN5 is not available in the `Helios++` database. Therefore, a custom scanner was created, as shown in Appendix E.

`Helios++` does include the `Riegl_vq-1560i` **scanner**, but it could not generate points from heights above 600 meters. Since all AHN4 flights were flown at altitudes above 1100 meters, using this scanner would produce different results and could not recreate the same scene. At first, the same approach as with the Leica CityMapper was tried, but this resulted in



(a) Area 3 from Figure 4.1, showing AHN4. Colors represent the `source id` of the points.

(b) Area 1 from Figure 4.1, showing AHN5. Colors represent the `source id` of the points.

Figure 5.5.: Scenes used to match scanning parameters with AHN4 and AHN5.

missing wall points and roof parts in all scenes. For this reason, the `Riegl_vq-880g` scanner was used instead, which gave results that were more similar and complete.

For AHN data, the **aircraft height** is found by calculating the median height for each flightline.

Next parameter is the **distance between flightlines**. This estimation of flightline spacing in the real AHN datasets is used to configure the synthetic scanner's flightline distance, ensuring the same overlap and coverage pattern in the simulated data. To calculate this, the direction (angle) of each segment in a flightline is first determined, and the average angle is then computed. Each flightline is processed in a loop, where a 2000-meter buffer is created around it. This buffer helps identify nearby flightlines using the "intersects" method. Only those flightlines with similar directions (i.e., matching angles) are retained. The distance from each filtered flightline to the main flightline is then calculated, and the closest one is selected. This process is repeated for all flightlines, and the resulting distances are stored.

Some of the distances are zero, which is expected. As shown in Figure 5.6, green flightlines overlap at the ends with orange flightlines, resulting in a zero distance. Additionally, there are flightlines, such as the red one, which have no nearby neighbors. For the final distance calculation for AHN4 and AHN5, only those flightlines with a closest distance of more than 300 meters are considered.

For the **scan angle** the Areas 1 and 3 are compared as described earlier. The scan angle is adjusted until the flight strips of the AHN and the simulated dataset match. The correct scan angle depends on the aircraft's height, the distance between flightlines, and the amount of overlap.

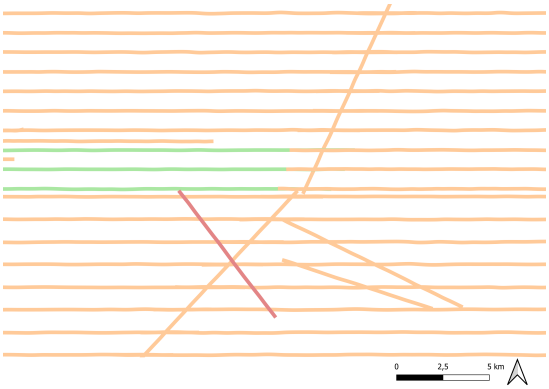


Figure 5.6.: Flightlines in South Holland from AHN4. The green lines have 0 distance to their nearest neighbor because they overlap. The red line has no close neighboring flightlines.

	AHN4	AHN5
Scannername	Riegl VQ 780i	Leica Citymapper2
Platform name	sr22	sr22
Height aircraft (m)	1354.35	1251
Distance Flightlines (m)	1032.96	801.5
Move speed (m/s)	10	135
Scan Frequency (Hz)	100	75
Scan angle (deg)	26.9	23.1
Pulse Frequency (Hz)	464900	2789000

To find the right **PRR**, a small area on top of a building is selected in the scenes shown in Figure 5.5. The point density of this area is calculated for both AHN4 and AHN5, as well as for the simulated datasets. PRR is then adjusted until the densities match. Since the flightlines are the same for both the real and simulated data, the point density should be made similar.

For AHN5, the **aircraft speed** is available in the flightstrip data. For AHN4, the speed is manually adjusted to match the expected point density.

For AHN5, the **scan frequency** is taken directly from the scan rate in the flightstrip data. For AHN4, it is manually adjusted to match the expected point density.

Generate Synthetic Scenes with Different Parameters

To ensure that the algorithm can handle different dataset characteristics, the smaller test scene (Figure 5.4) was created. Several simulated survey scenes were then generated using `Helios++` over this area.

The scanners and their settings used to create the five synthetic scenes are shown in Table 5.1. In Scene 1, all settings are identical. In Scene 2, the PRR differs between the two epochs. In Scene 3, the main difference lies in the scanning pattern and beam divergence. Scene 4 varies in the flightlines used. In Scene 5, the main difference lies in the accuracy between the two scanners.

	Scanner		Scanning Pattern		PRR (Hz)		Flightlines
Name Scene	E1	E2	E1	E2	E1	E2	E1E2
Scene 1	Riegl VQ-880g	Riegl VQ-880g	Conic	Conic	450000	450000	Same
Scene 2	Riegl VQ-880g	Riegl VQ-880g	Conic	Conic	100000	600000	Same
Scene 3	Leica ALS50-ii	Riegl LMS-q560	Oscillating	Rotating	450000	450000	Same
Scene 4	Riegl VQ-880g	Riegl VQ-880g	Conic	Conic	450000	450000	Different
Scene 5	Optech 3100	Leica ALS50	Oscillating	Oscillating	450000	450000	Same

Table 5.1.: Overview of the five scenes with different scanners and acquisition settings.

Helios++ Implementation

Below the implementation of `Helios++` is explained. For `Helios++`, two files need to be created:

1. A `scene file`, which defines which '.obj' file to use, and how to place, scale, or rotate it in the scene.
2. A `survey file`, which contains information about the scanner setup. This includes which scanner is used, its path, and the values of its parameters. Paths are defined using "legs", where each leg is a straight line between two 3D points (xyz). The same parameters are used throughout each leg. Scanners can be set as active or inactive for each leg.

Different types of scanners can be used. A list of the available scanners Helios++ uses can be found in this XML file.

The software generates a point cloud for each leg in '.xyz' format. Each point includes its x, y, z coordinates, intensity value, number of returns, return number, and GPS time. It also creates a text file that shows the scanner's trajectory for each leg.

5.4. Automatic Labelling of Changes

This section explains how the synthetic point cloud dataset and the ground truth (the `.obj` scene) are combined to create a labelled point cloud dataset. This labelling shows which points have changed and which have not. This step is important for the research because:

- the workflow includes many iterations, and this method automatically updates labels when the city scene or simulated scene is changed: no manual relabelling is needed,
- one large scene and five small scenes are created: this method allows labels to be generated automatically for each scene, which avoids manual labelling and saves time during testing and training,
- this approach is more accurate than manual labelling, since it matches each point to a specific face,
- it also gives insight into the type of change, because each point is matched to a specific face, and the type of face it belongs to.

In this section, the steps of the labelling process are explained. First, the faces in the `.obj` files are labelled. Second, the `.obj` houses are imported in C#. Third, the points are matched to each house. Fourth, the points inside each house are matched to a face. Fifth, the `.obj` houses from different epochs are matched to compare changes. After these steps, it is explained how the ground points are identified and separated.

Labelling Faces in OBJ Files

This subsection describes the first step of the process: labelling the faces in the OBJ files.

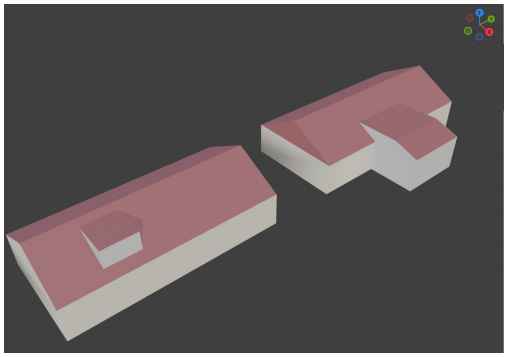
In `Blender`, each face of a house is assigned a specific material name that encodes the type of change that has occurred. This material name begins with three important numbers. The first number indicates the face type: 1 represents a roof, while 0 or 2 indicates a wall. The second number describes the nature of the change:

- 1 means the face has been added in the current epoch,
- 2 means it has been removed compared to the previous epoch, and
- 3 indicates a complete replacement, such as when a building is entirely rebuilt.
- If the second number is another value or separated by a comma, it means no change occurred.

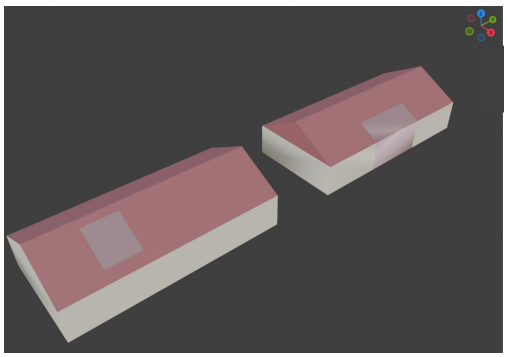
The third number represents the category of the change, with the following mapping:

- 1: House
- 2: Barn
- 3: Height or Roof change
- 4: Width expansion
- 5: Dormer
- 6: Balcony
- 7: Solar panel (4 cm thick)
- 8: Solar panel (8 cm thick)
- 9: Solar panel (12 cm thick)
- 0: Solar panel (16 cm thick)

This encoded information allows for the identification of which parts of a house have changed and in what way. It is also important that, when a feature is removed between epochs, the corresponding faces are stored separately in the epoch where they are removed. This is illustrated in Figure 5.7. Otherwise, if the roof consists of a single face, all points would be classified as either ”no change” or ”changed.” If all faces are grouped without distinction, any point matched to that surface would incorrectly be associated with either no change or all changes. An illustration of this separation is shown in Figure 5.7.



(a) Two types of buildings with additions. The left house has a new dormer, and the right house has an extension in width.



(b) Two types of buildings with removals. The left house lost a dormer, and the right house had a width reduction.

Figure 5.7.: Visual examples of added (a) and removed (b) parts in two building epochs, used to link 3D geometry with the synthetic point clouds.

Importing OBJ Houses

This subsection describes the second step of the process: importing the obj houses.

Import the 3D house models from OBJ files, and a custom parser was implemented to extract both geometric and semantic information. Each building is identified by o or g tags within the OBJ file, which denote object or group boundaries. Vertex definitions (v) are parsed to collect 3D coordinates, which are then used to construct polygonal surfaces from face definitions (f). The usemtl tags encodes metadata about each surface, including the type of face (roof or wall), the nature of change (add, remove, or modify), and the specific component affected (e.g., house, dormer, balcony). This encoded information is decoded using a mapping function and associated with each face. All labelled surfaces are grouped into a `HousePolygonsByLabel` object, which encapsulates the house name, a collection of labelled polygonal faces, and metadata indicating whether any change occurred. Each labelled face also maintains an initially empty list of associated point cloud points. This list is later populated with the closest light detection and ranging (LiDAR) points, allowing for labelling the points since the type of change for the faces is known.

Matching LiDAR Points to Each House

This subsection describes the third step of the process: matching the synthetically generated LiDAR points to individual houses.

The next step is to match the LiDAR points to each `HousePolygonsByLabel`. The most accurate method is to calculate the 2D distance between each point and every house, represented as a multipolygon, and assign the point to the house with the shortest distance. However, this method is very slow. An alternative approach is to store the points in a `KdTree` and use a bounding box (Envelope) for each multipolygon. For each envelope, a spatial query is performed on the `KdTree`. Since the envelope covers the entire bounding area of a house, it may include points that do not belong to the building, such as nearby architectural features. Additionally, due to sensor inaccuracies in the synthetic dataset, LiDAR points may

be slightly misaligned, which can be problematic, especially in areas with terrace houses. Although using envelopes is faster, it can lead to incorrect matches. Therefore, the method of checking each point against every house was chosen as the most reliable approach.

See Algorithm G.1 for the pseudocode of this part.

Matching LiDAR Points to Faces Inside the House

This subsection describes the fourth step of the process: matching the synthetically generated LiDAR points to faces inside the house.

To assign labels to the LiDAR points, they need to be matched to the faces within each `HousePolygonsByLabel`, as these faces contain the information about whether and how a part of the house has changed. Figure 5.8 illustrates how the distance from a 3D point to a 3D polygon is calculated. First, a plane must be extracted from the polygon. This is done by selecting three non-collinear points from the polygon, creating two vectors between them, and calculating the normal vector using the cross product. The normal vector's X, Y, and Z components are denoted as A, B, and C, while D is computed using one of the polygon's points. This gives the plane equation:

$$Ax + By + Cz + D = 0.$$

The perpendicular distance from the point to the plane is then calculated using:

$$\frac{Ax_{\text{point}} + By_{\text{point}} + Cz_{\text{point}} + D}{\sqrt{A^2 + B^2 + C^2}}.$$

To find the closest point on the plane, this distance is subtracted in the direction of the normal vector:

$$\vec{p}_{\text{closest}} = \vec{p}_{\text{original}} - \frac{Ax_{\text{point}} + By_{\text{point}} + Cz_{\text{point}} + D}{A^2 + B^2 + C^2} \cdot \begin{bmatrix} A \\ B \\ C \end{bmatrix}.$$

This gives the projection of the point onto the plane. To check if this point lies inside the polygon, the method `IsPointInsidePolygon` is used. It applies a 3D ray-casting approach: a ray is cast from the point in a direction that lies in the same plane as the polygon and is perpendicular to the normal vector. For each polygon edge, the method checks if the ray intersects it. If the number of intersections is odd, the point is inside the polygon; if even, it is outside. If the projected point is outside, the shortest distance to the polygon is calculated by checking the distance from the point to each edge of the polygon.

See Algorithm G.2 for the pseudocode of this part. The distance to the face is explained above and not shown in the pseudocode.

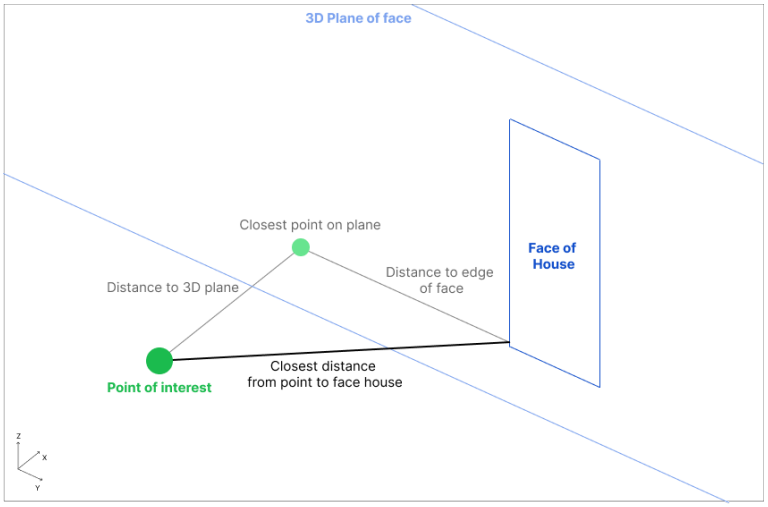


Figure 5.8.: Visualization of determining the distance between a point and a face of a house in 3D.

Matching OBJ Houses from Different epochs

This subsection describes the fifth step of the process: matching the houses from the two epochs.

The process of matching houses from two different epochs involves comparing each house from the first epoch with all houses from the second epoch using the convex hull of their respective polygons. For each house, the intersection area between its convex hull and that of the potential matches is computed. If the intersection area exceeds 60% of the area of the first house's convex hull, the intersection rate is calculated as the maximum ratio between the intersection area and the area of either of the two houses' convex hulls. When a match with a high intersection rate is identified, the two houses are added to the `matchedHouses` list. If no match is found, the house is marked as removed in the first epoch. After processing all houses from the first epoch, any remaining unmatched houses from the second epoch are marked as added.

See Algorithm G.3 for the pseudocode of this part.

Ground Points

As explained in Section 5.2, a flat surface (plane) is added below the houses to act as the ground. Then, a synthetic point cloud is generated for the ground without the houses. After that, all points located underneath a house are removed, and the remaining ground points are given a LiDAR classification value of 2.

Including ground points is important because they are needed to calculate the height difference feature. This feature is used by the random forest (RF) classifier in Section 5.5, especially for detecting houses that have been extended in width.

Output

From the matching process, the following information can be obtained for each house: whether it has been changed and the type of change that occurred. Additionally, for each point, the output includes its classification (e.g., wall or roof) and whether it has been altered compared to the other epoch, along with the type of change (added, removed, modified, or unchanged). A "modified" change refers to cases where a house has been removed and replaced by a new one, for example.

See Algorithm G.4 for the pseudocode of the general pipeline of the labelling process.

5.5. Point-Level Change Probability Estimation

This section explains how the random forest (RF) classifier is trained, to compute a point-level classification. This step is important for the research because:

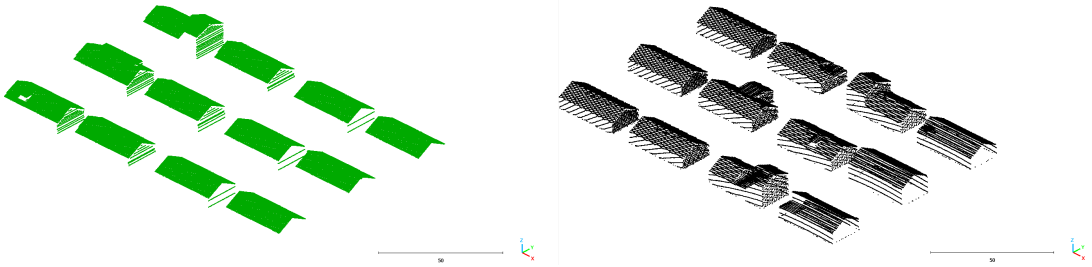
- it incorporates point features, both individually and in context, such as distance to the flightline and the stability factor,
- it takes into account dataset characteristics, like differences in point density or accuracy between the datasets,
- it gives each point a probability of being changed or unchanged.

In this section, all the features of the points are explained. These include the features as outline in Table 5.2.

Table 5.2.: Overview of input features used for change detection

Feature Type	**Description**
Point-Level Features	
Distance to flightline	Horizontal distance to the nearest flightline
Tilt Angle	Measures angle of a fitted plane through the neighbouring points
3D Density	The amount of points around the target point
stability factor around point	Measures the stability factor in the local neighborhood
Height difference	Vertical difference between epochs
stability factor difference	Change in local geometric consistency
3D point density difference	Change in 3D point density around the point
2D point density difference	Change in density when projected on 2D plane
Linearity difference	Change in the degree to which points form a line
Planarity difference	Change in the degree to which points form a plane
Sphericity difference	Change in the degree to which points form a sphere
Dataset-Level Features	
Beam divergence difference	Change in beam spread between datasets
Point density difference	Overall point density variation
Planimetric accuracy difference	Difference in horizontal accuracy
Height accuracy difference	Difference in vertical accuracy

To illustrate the features at each point, the scene shown below is used (Figure 5.9). Individual points are color-coded to represent their respective features.



(a) Simulated scene of the reference point cloud dataset.

(b) Simulated scene of the target point cloud dataset.

Figure 5.9.: Synthetic dataset showing two different epochs of the same scene. This example will be used again later to help visualize point-level features.

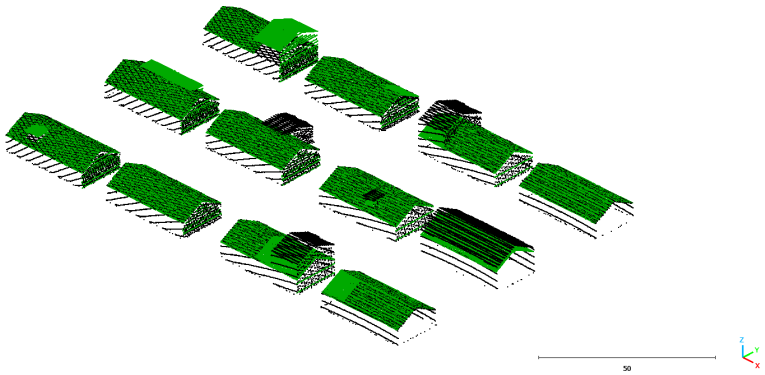


Figure 5.10.: Combined view of the scene from both epochs. Points from the reference point cloud dataset are shown in green, and points from the target point cloud dataset are shown in black. For the separate views, refer to Figure 5.9a and Figure 5.9b.

Intrinsic Point-Level Features

The first feature of the category `Intrinsic Point-Level Features` represents the **distance to the flightline** within the epoch when the point was recorded. This feature captures the increased uncertainty that appears when points are farther from the flightline. The calculation method for the distance between a point and a flightline is explained in Section 5.1.

The second feature is the **tilt angle**, which helps to differentiate vertical surfaces (such as walls) from horizontal surfaces (such as roofs). This angle is calculated based on the 3D positions of the nearest neighbouring points around the target point. First, the covariance

matrix of the local neighbourhood is computed. Then, eigenvalue decomposition is performed to find the main directions. The third eigenvector (corresponding to the smallest eigenvalue) estimates the surface normal, representing the direction with the least variation in point distribution. The angle between this surface normal and the vertical axis (Z-axis) gives the tilt angle of the local surface. A tilt angle near 0 or 180 degrees indicates a horizontal surface, while an angle near 90 degrees suggests a vertical surface. Results are presented in Figure 5.11.

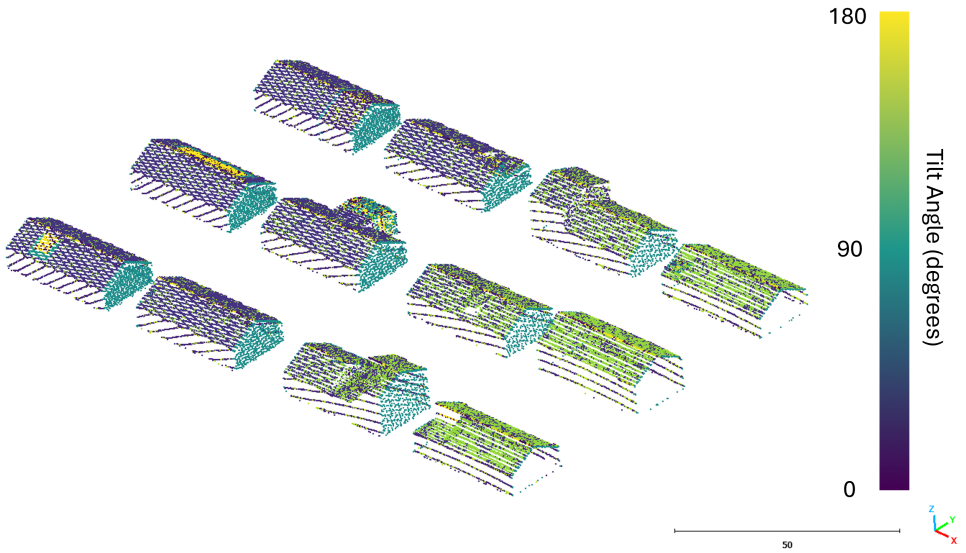


Figure 5.11.: Visualization of the second epoch of the synthetic dataset shown in Figure 5.9. Points are color-coded based on the `Tilt Angle` feature: yellow indicates a high angle (approximately 180 degrees), and dark blue indicates a low angle (approximately 0 degrees).

The third feature is the **3D density** around the point within its point cloud. This feature is important because the accuracy of the tilt angle depends on the number and distribution of neighbouring points. The result is illustrated in Figure 5.12.

The fourth feature is the **stability factor** around the point. As described in Section 2.6 and shown in Figure 2.11, this factor is the ratio between the number of points in a 2D neighbourhood and those in a 3D neighbourhood (Tran et al., 2018). It is calculated as:

$$\text{Stability Factor} = \frac{N_{\text{spherical}}}{N_{\text{cylindrical}}}, \tag{5.1}$$

Where $N_{\text{spherical}}$ is the number of points inside a sphere of fixed radius, and $N_{\text{cylindrical}}$ is the number of points inside a vertical cylinder of the same radius but extended in height. The result is illustrated in Figure 5.13.

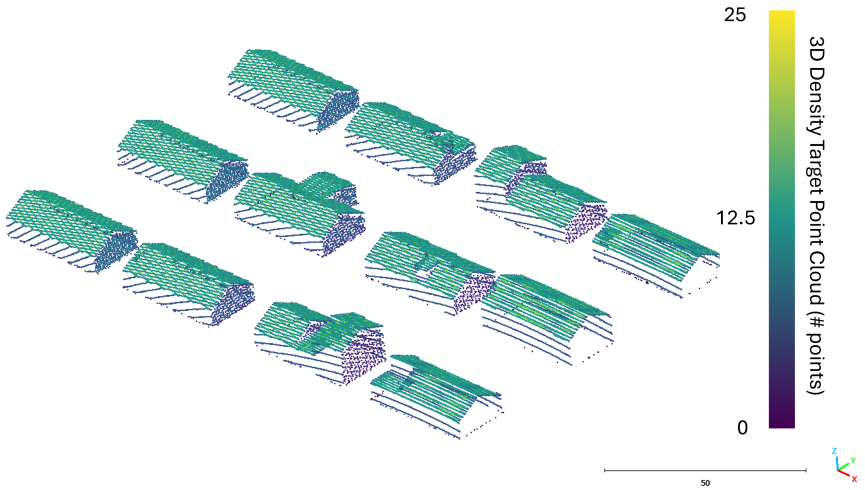


Figure 5.12.: Visualization of the second epoch of the synthetic dataset shown in Figure 5.9. Points are color-coded based on the `3D Density` feature: yellow indicates a high density (25 neighbouring points), and dark blue indicates a low density (0 neighbouring points).

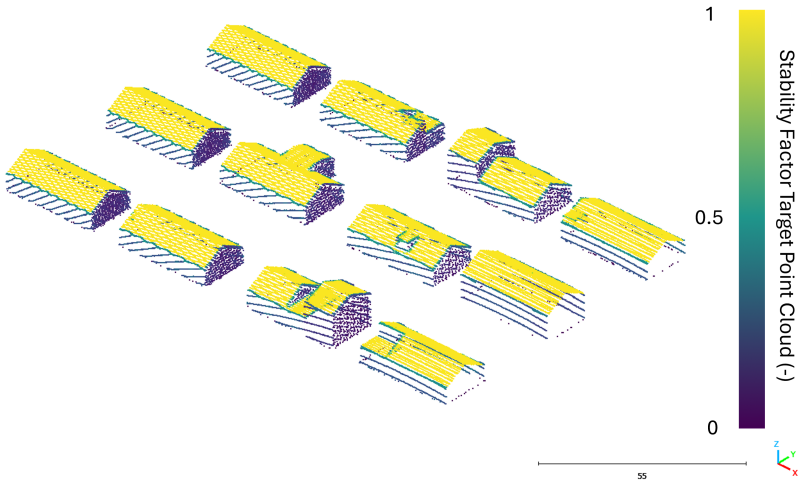


Figure 5.13.: Visualization of the second epoch of the synthetic dataset shown in Figure 5.9. Points are color-coded based on the `stability factor` feature: yellow indicates a high stability factor of 1, and dark blue indicates a low stability factor of 0.

Point Features Compared Between epochs

The first features of the category `Point Features Compared Between` epochs require finding the nearest neighbours in the other epoch, both in 2D and 3D space. A method based on the KdTree data structure was implemented to enable fast spatial queries. For each target point, a square search window in the 2D plane (X and Y coordinates) is defined using a given threshold distance. This window forms a bounding box used to query the `KdTree` for nearby points. The query is first executed on the KdTree containing only building points from the reference point cloud dataset. If no neighbours are found within the 2D threshold, the query is repeated on the full reference point cloud dataset, which includes all object types. To identify 3D neighbours, points found in 2D are filtered by height (Z coordinate). Only points with height differences smaller than the threshold remain. The same process is repeated for the target point cloud dataset. The chosen threshold distance is 0.5 meters.

The first feature is the **height difference**. This feature is based on observations from the exploration phase (see Section 6.1). To calculate it, 2D neighbours within 0.5 meters are considered. If none exist, the threshold increases by 1 meter repeatedly until at least one neighbour is found. Then, the 3D distances to these neighbours are calculated, and the smallest distance is used as the feature value. For the scene in Figure 5.9, the Height Difference is shown in Figure 5.14.

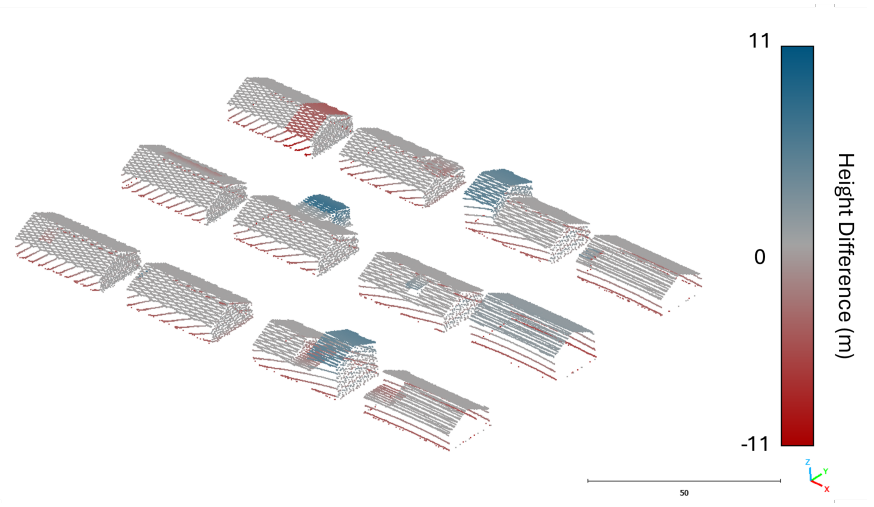


Figure 5.14.: Visualization of the second epoch of the synthetic dataset shown in Figure 5.9. Points are color-coded by the absolute `Height Difference` feature: blue indicates a higher height in the current epoch compared to the previous, red indicates a lower height, and gray indicates minimal difference.

The second feature is the **Difference in stability factor** relative to the nearest neighbour in the other epoch. This factor was explained earlier in this section. For the scene in Figure 5.9, the stability factor Difference is shown in Figure 5.15.

The third feature measures the **Difference in 3D point density** around the point. This is calculated by counting the number of neighbouring points within a set radius. The corresponding visualization is in Figure 5.16.

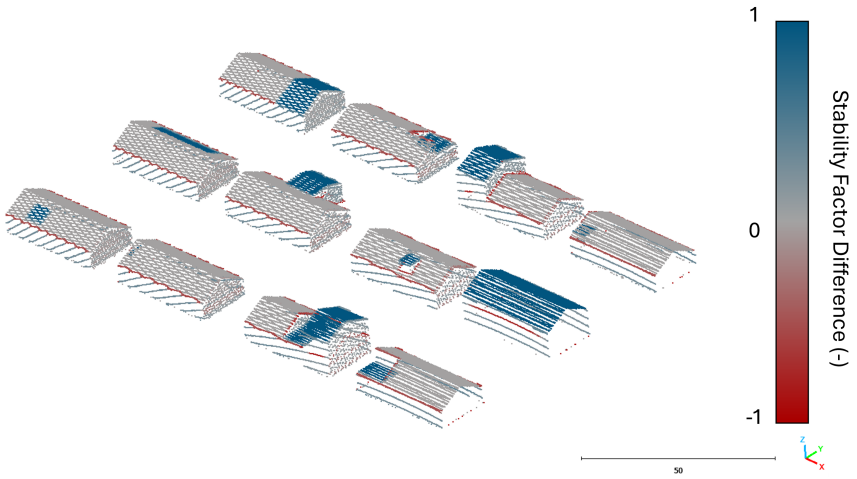


Figure 5.15.: Visualization of the second epoch of the synthetic dataset shown in Figure 5.9. Points are color-coded by the `stability factor Difference` feature: blue indicates a higher stability factor in the current epoch compared to the previous, red indicates a lower stability factor, and grey indicates minimal difference.

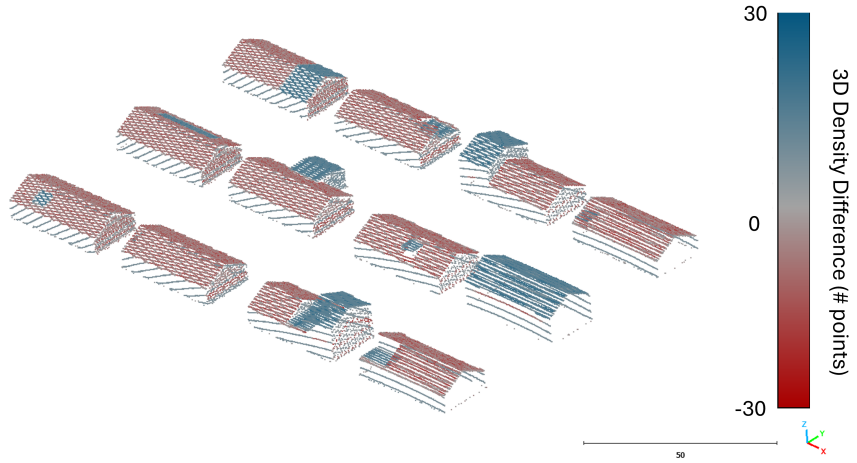


Figure 5.16.: Visualization of the second epoch of the synthetic dataset shown in Figure 5.9. Points are color-coded by the absolute `3D Density Difference` feature: blue indicates higher density in the current epoch, red indicates lower density, and grey indicates minimal difference.

The fourth feature represents the **Difference in 2D density** around the point. The corresponding visualization is in Figure 5.17.

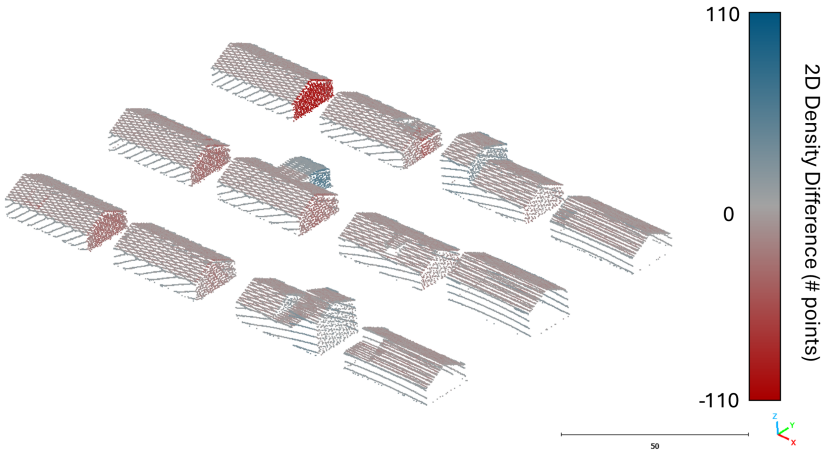


Figure 5.17.: Visualization of the second epoch of the synthetic dataset shown in Figure 5.9. Points are color-coded by the `2D Density Difference` feature: blue indicates higher density in the current epoch, red indicates lower density, and grey indicates minimal difference.

The fifth, sixth, and seventh features are the differences in **linearity**, **planarity**, and **sphericity**. These geometric features describe the local neighbourhood within the same epoch as the point.

- Linearity measures how closely the points align along a straight line. Higher values mean a more linear shape.
- Planarity measures how well the points lie on a plane. Larger values mean a flatter surface.
- Sphericity measures how close the points are to forming a sphere. Higher values mean a more spherical (isotropic) shape.

These features are computed from the eigenvalues of the covariance matrix of the neighbourhood points, sorted as $\lambda_1 \geq \lambda_2 \geq \lambda_3$. The formulas are:

$$\text{Sphericity} = \frac{\lambda_3}{\lambda_1}, \quad \text{Linearity} = \frac{\lambda_1 - \lambda_2}{\lambda_1}, \quad \text{Planarity} = \frac{\lambda_2 - \lambda_3}{\lambda_1}. \tag{5.2}$$

The eigenvalues λ_i describe:

- λ_1: Variance along the main direction, representing the dominant axis.
- λ_2: Variance in the second direction, showing the spread on a plane.
- λ_3: Variance in the smallest direction, indicating thickness or depth.

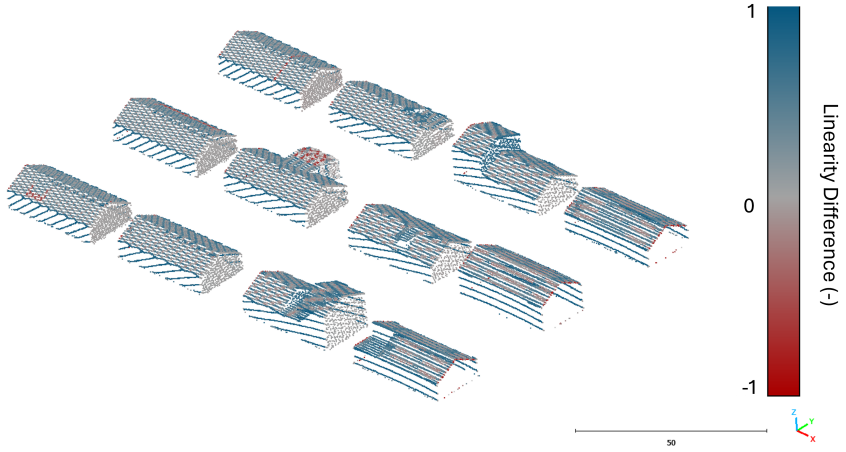


Figure 5.18.: Visualization of the second epoch of the synthetic dataset shown in Figure 5.9. Points are color-coded based on the `Linearity Difference` feature: blue represents low linearity, red represents high linearity.

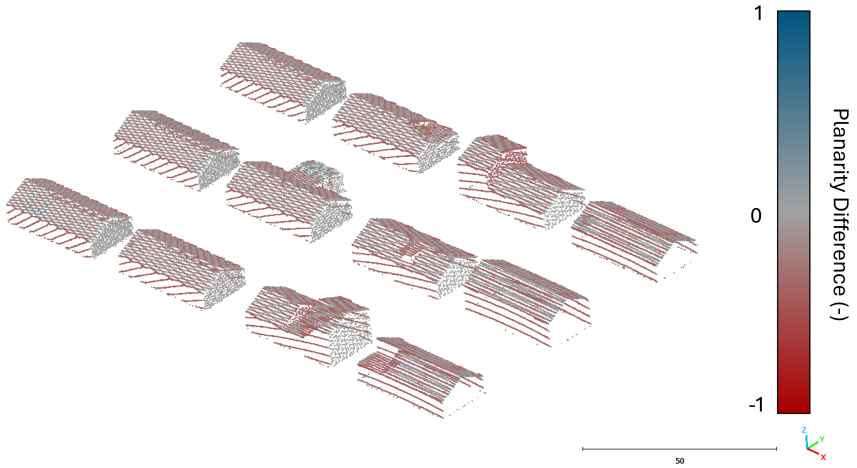


Figure 5.19.: Visualization of the second epoch of the synthetic dataset shown in Figure 5.9. Points are color-coded based on the `Planarity Difference` feature: blue represents low planarity, red represents high planarity.

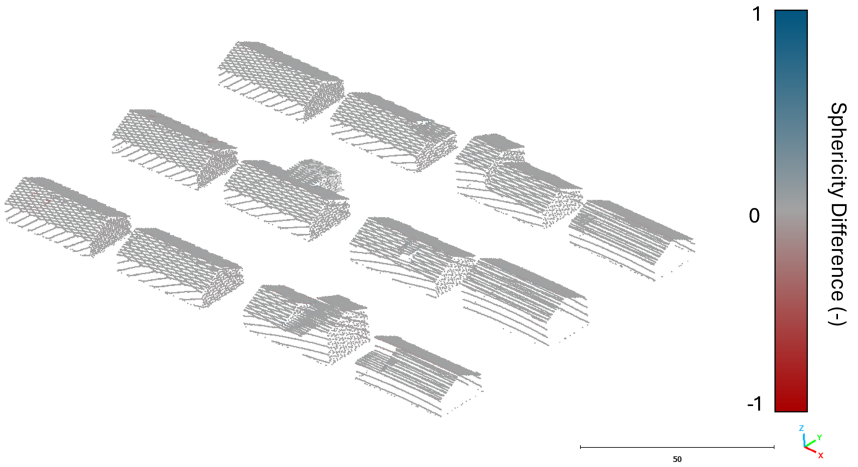


Figure 5.20.: Visualization of the second epoch of the synthetic dataset shown in Figure 5.9. Points are color-coded based on the `Sphericity Difference` feature: blue represents low sphericity, red represents high sphericity.

Dataset-Level Features

For each scene, several dataset-level characteristics are measured and employed as input features. So when a real dataset is the input, those features need to be calculated. The four calculated values for each scene include:

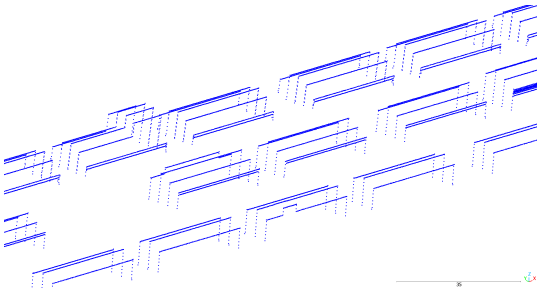
1. **Beam divergence difference**: The difference between the beam divergence values of the two scanners involved in the scene.
2. **Point density difference**: The difference in point density between epochs. In synthetic scenes, this is calculated by dividing the number of points by the surface area of the building polygons.
3. **Planimetric accuracy difference**: The difference in horizontal accuracy. With ground truth available for synthetic data, two buildings are selected, and the average horizontal offset of the wall points is computed.
4. **Height accuracy difference**: The difference in vertical accuracy. For the same two buildings, the average vertical offset of the roof points is calculated.

These values are listed in Table 5.3. Scene 6 corresponds to the matched national height model of the Netherlands number 4 (AHN4) national height model of the Netherlands number 5 (AHN5) dataset.

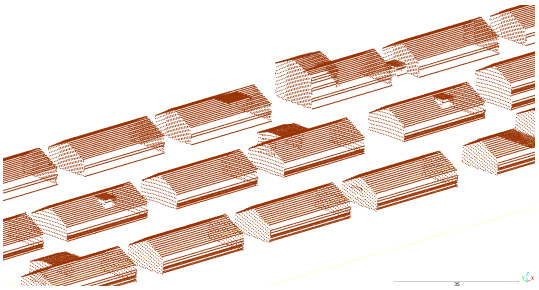
Scene 2 from Table 5.3 is excluded from the training and validation datasets due to low point cloud quality in the reference point cloud dataset. The lower pulse repetition rate (PRR) in the reference point cloud dataset leads to significantly sparser point coverage with large gaps, producing unrealistic patterns that do not represent real airborne scans. This sparse, uneven point distribution causes lower data quality, which supports the exclusion of the reference point cloud dataset in Scene 2 from training and validation due to its negative impact on model performance. This is visible in Figure 5.21.

	Scanner		Beam Divergence [mrad]		Density [pts/m²]		Planimetric Accuracy [cm]		Height Accuracy [cm]	
Name Scene	E1	E2	E1	E2	E1	E2	E1	E2	E1	E2
Scene 1	Riegl VQ-880g	Riegl VQ-880g	0.3	0.3	9.74	9.73	13.0	14.8	5.87	6.92
Scene 2	Riegl VQ-880g	Riegl VQ-880g	0.3	0.3	2.17	13.04	13.2	14.7	5.65	6.78
Scene 3	Leica ALS50-ii	Riegl LMS-q560	0.22	0.5	7.22	4.92	4.4	14.7	0.26	1.90
Scene 4	Riegl VQ-880g	Riegl VQ-880g	0.3	0.3	12.24	14.34	20.0	17.2	6.43	5.74
Scene 5	Optech 3100	Leica ALS50	0.424	0.33	7.82	7.55	8.9	7.50	1.22	0.58
Scene 6	Riegl VQ-1560i	Leica CityMapper-2	0.18	0.23	19.06	12.82	12.8	5.77	1.17	0.29

Table 5.3.: Differences in scanner specifications and accuracy between epochs E1 and E2 for each scene.



(a) The reference point cloud dataset of Scene 2, showing low PRR and resulting low point density.



(b) target point cloud dataset 2 of Scene 2, showing high PRR and resulting high point density.

Figure 5.21.: Maximum difference in pulse frequency between epochs in Scene 2.

Feature Selection, RF Parameters, and Evaluation

Before training the classifier, the dataset is balanced by applying stratified under-sampling to the changed points and uniform sampling to the unchanged points, resulting in approximately 58% unchanged and 42% changed points. This approach avoids naive over- or under-sampling while maintaining a representative distribution. Minor changes, such as the addition of a balcony, are included to ensure completeness. The data is then split into training, validation, and testing subsets.

The validation set is used to select the most effective input features and determine the best-performing feature combinations. Two methods are used to identify the most important features for point-level classification: permutation feature importance and RF feature importance. These approaches help to measure how much each feature contributes to model performance. Further details are provided in Section 2.4.

Different combinations of input feature types and change scenarios were tested on a validation dataset to determine which configurations perform best. The possible types of changes include balcony additions, dormers, height increases, complete house changes, house width extensions, and solar panels with thicknesses of 4, 8, 12, and 16 cm. The following input feature sets were evaluated:

1. **Set 1:** Includes all change types and all dataset-level features.
2. **Set 2:** Excludes intrinsic point-level features: tilt angle and 3D point density. This choice is motivated by visual inspection (Figure 5.11), where the tilt angle did not show clear discrimination power. Additionally, the 3D density feature was excluded

because tilt angle values can be misleading when few neighboring points exist, causing exaggerated tilt measurements.

3. **Set 3:** Excludes most dataset-level features: beam divergence difference, planimetric accuracy difference, and height accuracy difference. This set was used to evaluate whether the model effectively uses these features, which are constant for all points within the same dataset and thus have limited variability. The only dataset-level feature retained here is the density difference, as it showed the most promising results.
4. **Set 4:** Excludes all dataset-level features, including density difference, to further test the model's reliance on these features.
5. **Set 5:** Removes training data involving solar panels with thicknesses of 4 cm and 8 cm by filtering them out during data balancing. These small changes are excluded to examine whether they negatively impact the detection of unchanged areas. Since the accuracy of most datasets is limited to changes larger than 8 cm, this set tests if excluding thinner panels improves model performance.

For visualizing these results, two sets will be selected based on the found accuracy scores.

5.6. Occlusion Type Classification per Point

This section explains how it is checked if a point is hidden (occluded) in one of the epochs, and what type of occlusion it is. To know the different types of occlusions, see Figure 4.4. This step is important because:

- It helps to handle occlusion at the point level, which is a common problem in many change detection methods that causes false changes. Most methods detect these areas as changed since a lot of information (points) are missing and it is hard to compare. For this research these areas should be detected as unknown.

To decide which occlusion class a point belongs to (static occluded, dynamic occluded, or visible), two booleans (True/False) are checked:

- **Boolean 1:** Is the point hidden when using the point cloud and aircraft positions from the reference point cloud dataset? This is called **Occluded with reference point cloud dataset Aircraft Positions**.
- **Boolean 2:** Is the point hidden when using the point cloud from the reference point cloud dataset and aircraft positions from the target point cloud dataset? This is called **Occluded with target point cloud dataset Aircraft Positions**.

The occlusion class is assigned based on these checks:

- If Boolean 1 is `True`, the point is **Class 0 (Visible)**.
- If both Boolean 1 and Boolean 2 are `False`, the point is **Class 2 (dynamic occluded)**.
- If Boolean 1 is `False` and Boolean 2 is `True`, the point is **Class 1 (static occluded)**.

Each boolean is calculated in four steps:

1. Find the relevant flightlines. For the first boolean, the flightlines from the reference point cloud dataset are used. For the second boolean, the flightlines from the target point cloud dataset are used.

2. Get the aircraft positions along these flightlines.

3. Select all points that lie between the aircraft and the point we want to check.

4. From these, filter the points that block the view to the point.

Step 1: Identifying Flightlines That Could See the Point

First, the flightlines that could have observed the point are identified. This is done by representing each flight strip as a polygon and checking whether it contains the target point in the (x,y) plane.

See Algorithm G.5 for the pseudocode of this part.

Next, in step 2 the positions of the aircraft on the flightlines are determined. This step considers the viewing angles of the aircraft and the light detection and ranging (LiDAR) scanning pattern. As described in Section 3.1, national height model of the Netherlands number 4 (AHN4) uses a zigzag scanning pattern, while national height model of the Netherlands number 5 (AHN5) employs an oblique LiDAR scanner, requiring different approaches.

Step 2a: Determining Aircraft Positions on Flightlines in the Synthetic Dataset

In the synthetic dataset, each point and each coordinate along a flightline is associated with a GPS timestamp. This level of detail is not available in the AHN4 and AHN5 flightlines; it is only stored at the point. Due to the availability of GPS time information, the position of the scanner (or aircraft) at the moment a point was captured can be estimated.

The estimation process begins by identifying the closest flightlines to the target point. For each of these flightlines, neighbouring points are selected if their GPS times fall within the start and end times of the flightline. Among these, the 3D neighbour closest to the target point is identified.

Using the GPS time of this nearest neighbour, two consecutive coordinates on the flightline that enclose the GPS time are found. The position of the scanner is then calculated through linear interpolation between these two coordinates. This provides an estimated position of the aircraft at the time the neighbour point was recorded.

See Algorithm G.6 for the pseudocode of this part.

Step 2b: Determining Aircraft Positions on Flightlines Leica CityMapper-2

The scanning pattern for AHN5 is shown in Figure 3.4. Based on this pattern and the knowledge of which flightlines could potentially observe the point, there are two possible aircraft positions for each flightline. These positions are illustrated in the middle part of Figure 5.22.

To determine these positions, a buffer is created around the target point. The radius of this buffer corresponds to the scanning range of the oblique scanner. The points where the flightline's (multi)linestring intersects the boundary of this buffer represent the aircraft positions that could have scanned the point.

The radius of the scanning circle is calculated as follows:

- The Field of View (field of view (FOV)) for each flightline is extracted from the flight strip data.
- The aircraft height must be determined. This is done per flightline by taking the median of the elevation coordinates of the (multi)linestring representing the flightline.
- The radius of the scanning circle is then calculated as:

$$\text{radius} = \tan\left(\frac{\text{FOV}}{2}\right) \times \text{aircraft height}. \tag{5.3}$$

This ensures that the correct aircraft positions are identified based on the scanning geometry.

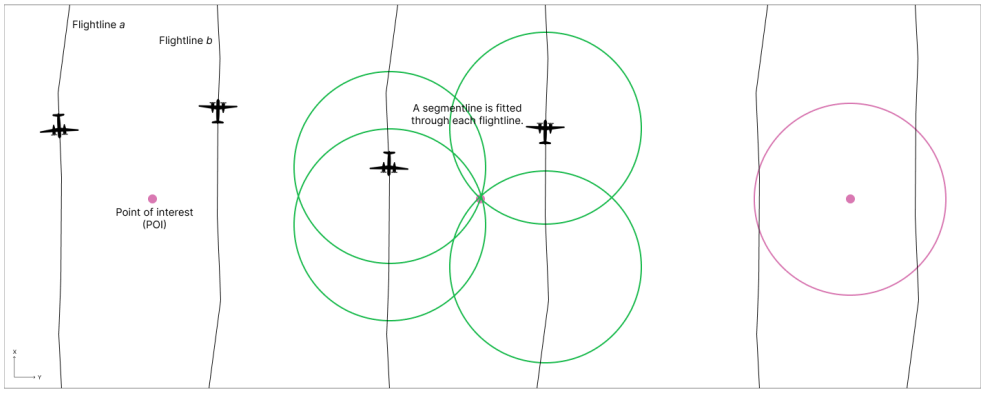
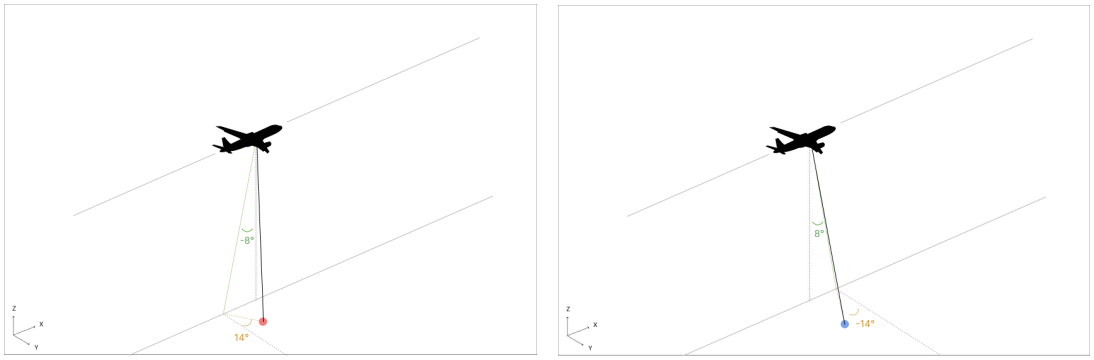


Figure 5.22.: The left part of the figure presents an example scenario. The middle part illustrates which oblique scans are capable of observing the point. The right part depicts the buffer around the target point, which is used to determine the possible aircraft positions.

See the first part of Algorithm G.7 for the pseudocode of this part.

Step 2c: Determining Aircraft Positions on Flightlines Riegl VQ-1560i Scanner

The scanning pattern for AHN4 is illustrated in Figure 3.2 and further visualized in 3D in Figure 5.23.



(a) Channel 1 conducting a point at the right side of the flightline. Angles are visible in one drawing. 2D drawings visible in Figure 3.2a.

(b) Channel 2 conducting a point at the right side of the flightline. Angles are visible in one drawing. 2D drawings visible in Figure 3.2b.

Figure 5.23.: For both scanners of the Riegl VQ-1560, the angles are shown in one drawing when seeing a point.

The process of determining the aircraft position from a given target point is illustrated in Figure 5.24 and Figure 5.25. Below is a detailed breakdown of the steps:

1. First, after identifying which flightlines could potentially see the point, a line segment is created for each flightline. Since flightlines are stored as (multi)linestrings, some computations of the next step can be challenging. To simplify the process, each flightline is temporarily represented as a straight line segment, using its first and last points. Later, the segment will be mapped back onto the original (multi)linestring. These segments exist in the (x,y) plane. This step corresponds to the first step of Figure 5.24.
2. The next step is to determine the closest point (C_i) on each of these line segments to the target point. This step is illustrated in the second step in Figure 5.24.
3. Since the LiDAR scanner is tilted at an angle of 14° in the (x, y)-plane, this needs to be accounted for. The correction is applied by multiplying the distance from the target point to C_1 and C_2 by the tangent of 14 degrees. This adjustment ensures that the position of the aircraft accounts for the tilted scanning angle. The resulting new points are labelled D_1, D_2, D_3, and D_4. This step corresponds to the third step in Figure 5.24.
4. Next, lines are drawn from the target point to D_1, D_2, D_3, and D_4. These lines are then extended until they intersect in the (x,y) plane with the flightlines. The intersection points, which represent the projected positions on the original (multi)linestrings, are labelled E_1, E_2, E_3, and E_4. This step is illustrated in the fourth step in Figure 5.25.
5. A buffer is then applied around E_1, E_2, E_3, and E_4 in the (x,y)-plane. The size of this buffer is calculated as the tangent of 8° multiplied by the z-coordinate of the point itself (i.e., the height of E_i). This accounts for the forward and backward scan angles when scanning in a non-nadir direction. The intersection points between these buffer boundaries and the flightlines represent the possible aircraft positions. This step generates eight possible positions, labelled F_1, ..., F_8, each having x, y, and z coordinates. This step corresponds to the fifth step in Figure 5.25.
6. For each flightline, the flight direction is determined by checking which coordinate is saved first and which is saved last in the (multi)linestring geometry. Additionally, it is

determined whether the target point is located to the left or right of the flightline's direction. This is calculated using the orientation index function applied to the flightline segment:

- `-1`: The point is to the right of the flightline.
- `1`: The point is to the left of the flightline.
- `0`: The point and the flightline segment are collinear.

This step corresponds to the sixth step in Figure 5.25.

7. Finally, to determine the true aircraft positions from the eight possible points (F_1, ..., F_8), three key factors are considered:
   a) The flight direction
   b) Whether the target point is left or right of the flightline.
   c) Whether the point was scanned by Scanner 1 or Scanner 2.

   The scanner type can be identified using Figure 3.2:

   - If the point is to the right of the flightline, the first aircraft position (in flight direction corresponds to Scanner 1.
   - If the point is to the left of the flightline, the first aircraft position (in flight direction corresponds to Scanner 2.

   Scanner 1 has a backward-facing scanner, so the last possible position (in flight direction is the correct one. Scanner 2 has a forward-facing scanner, so the first possible position (in flight direction) is the correct one. This final selection process is illustrated in the seventh step in Figure 5.25.

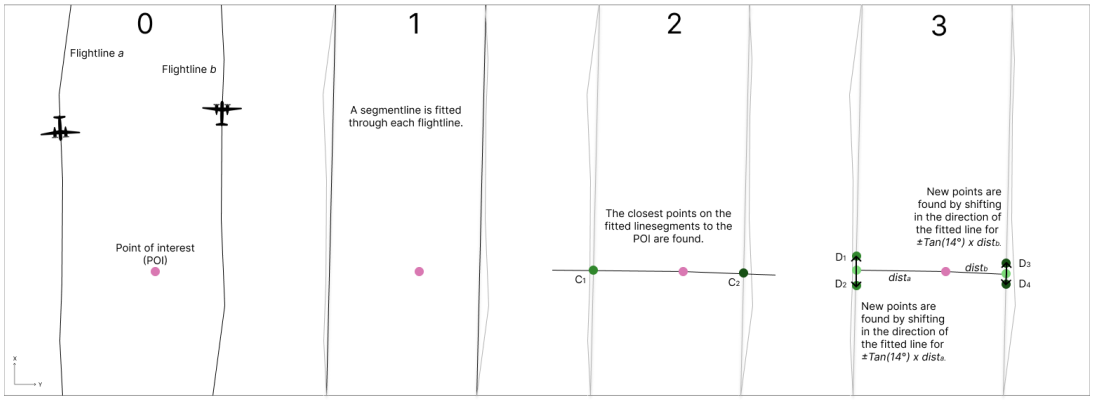


Figure 5.24.: The initial condition and the first three steps in determining the aircraft positions for a Riegl VQ-1560i scanner that could have observed the target point. The left section shows an example of two flightlines near the point. The middle section illustrates the first step, which involves fitting line segments. The next step identifies the closest point on the line segment to the target point. The right section depicts how the projected point shifts to account for the 14-degree tilt of the scan angle.

See the third part of Algorithm G.7 for the pseudocode of this part.

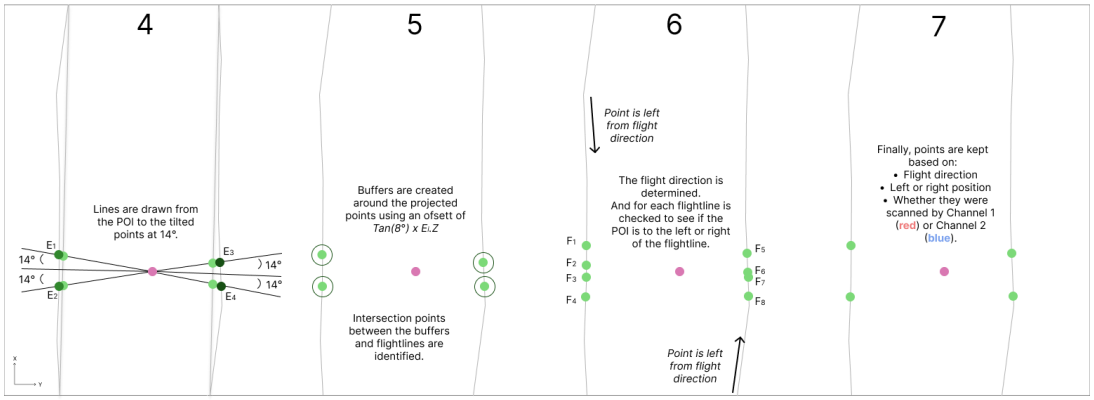


Figure 5.25.: The final four steps in determining the aircraft positions for a Riegl VQ-1560i scanner that could have observed the target point. The left section shows the step where lines are fitted from the target point to the projected points, followed by their intersection with the flightline. The next step applies a buffer to account for the forward and backward scan angles and finds intersection points between the buffer boundaries and the flightlines. The right two sections focus on selecting the correct points based on the flight direction, whether the point is to the left or right of the flightline, and whether it was scanned by channel 1 (Figure 5.23a) or channel 2 (Figure 5.23b).

Step 3: Selecting Relevant Points

With the known positions of the aircraft, a 3D ray is created from the target point to the aircraft. If the point cloud is stored in a NetOctree, all points located along this ray can be retrieved within a certain distance. The radius of the ray is calculated using Equation 5.4:

$$r_{\text{ray}} = 2 \cdot \tan\left(\frac{\theta_{\text{div}}}{2}\right) \cdot h_{\text{aircraft}} + 2 \cdot \sigma_{\text{hor}} + 2 \cdot \sigma_{\text{ver}}. \tag{5.4}$$

Where θ_{div} is the laser beam divergence angle (in radians), h_{aircraft} is the height of the aircraft, σ_{hor} is the horizontal accuracy of the scanner, and σ_{ver} is the vertical accuracy of the scanner.

The laser beam divergence determines the footprint size of the laser pulse on the ground. It represents the angle between the laser beam's normal and the boundary of its footprint. This divergence defines the effective width of the laser beam and is used to set a tolerance threshold around the visibility ray. For each aircraft position, a 3D ray is constructed from the target point towards the aircraft position. The direction vector is normalized, and the full distance to the aircraft is computed. An allowable error margin is calculated based on the laser divergence and the aircraft's height above the target point. This margin represents the radius around the ideal laser path within which potential occluders can exist. Using a spatial index (specifically a PointOctree), all points near the ray within the threshold margin are retrieved.

Only those with the same values for `Return Number` and `Number of Returns` are kept from the retrieved points. This ensures that the characteristic LiDAR behavior of penetrating through tree canopies is preserved.

Step 4: Determining Occlusion

The points selected in the previous step may belong to the same surface as the target point or to an obstacle that could block it. To determine whether the target point is occluded, a check is performed to see if any of the selected points are within a certain distance. If such points are present, the target point is not occluded. Otherwise, it is considered occluded.

Another approach involves fitting 3D planes through the selected points. If no well-fitting plane is found that includes the target point, it is assumed that it is occluded. To fit a 3D plane, points are first grouped based on their distance. Then, for each group, the best-fitting plane is computed following the method described in Eberly [2024]. The process consists of the following steps:

1. Compute the mean of each coordinate (X, Y, Z).
2. Center the points by subtracting the mean value.
3. Calculate the covariance matrix of the centered points.
4. Perform eigenvalue decomposition to obtain eigenvectors and eigenvalues.
5. Identify the eigenvector corresponding to the smallest eigenvalue, which represents the normal of the plane.
6. Compute the plane parameter d using the normal and mean values.

Ultimately, this method was not used because it requires a long computation time.

The third and final selected approach is carried out as follows. Each nearby point is evaluated using the following conditions:

- The point is located along the segment between the target point and the aircraft, based on its projection onto the direction of the laser ray.
- The point is positioned at least a small distance ahead of the target point, in the direction of the ray. This distance is defined by a threshold called `ErrorHeight`, which ensures the point is slightly in front of the target point and not on the same plane. The `ErrorHeight` is equal to the radius of the ray, as defined in Equation 5.4.

These checks confirm whether a point is physically in a position to block the line of sight between the aircraft and the target point. If three or more points meet the above occlusion conditions, the target point is classified as *occluded*. If no such points are found along the ray from any aircraft position, the target point is classified as *not occluded*.

See Algorithm G.8 and Algorithm G.9 for the pseudocode of this step and the previous step (step 3).

5.7. Integrating Occlusion into Certainty Scoring

This section describes how occlusion is used to estimate uncertainty and improve the reliability of the change detection algorithm. Two methods are explored. The first includes occlusion as an input feature for a second random forest (RF) model. The second method modifies the probability score from the first RF model to reflect uncertainty.

Model 1: Second RF Model With Occlusion as Input

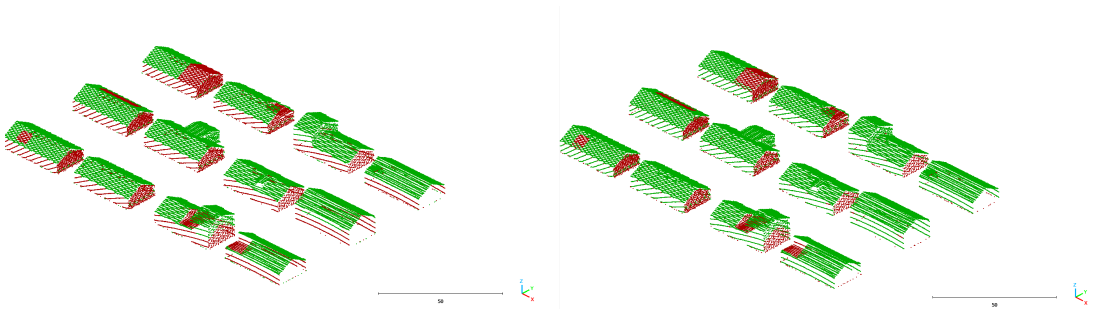
For the second RF model, several input features are used. The first is the **type of occlusion** for the target point. The types of occlusion are explained in Section 5.6 and illustrated in Figure 4.4. There are two occlusion boolean values, from which the occlusion type of each point is determined.

The second feature is the **probability score** from the first RF model for the same point. The third feature is the average probability score of neighboring points, referred to as the *ratio probability*. This feature helps to smooth predictions by including context from nearby points. It aims to reduce the influence of outliers and move from a purely local to a more global interpretation.

Model 2: Post-processing Probability Scores Using Occlusion

The second method adjusts the output probability score from the first RF model based on occlusion status. The idea is that if a point is not visible to the scanner, it is uncertain whether a change has occurred, because no reliable data is available for that region. In such cases, the prediction should be less confident, and the certainty score should be 0.5, representing uncertainty. This approach is explained conceptually in Figure 4.3.

Again, the types of occlusion are described in Section 5.6 and visualized in Figure 4.4. Figure 5.26 shows the binary occlusion labels for the synthetic scene. In Figure 5.26b, most wall points should ideally appear green (visible). However, walls facing the x-direction appear red, indicating dynamic occluded. According to the boolean rules in Section 5.6, these points are incorrectly labelled as dynamic occluded. This problem frequently occurs for wall points.



(a) Occlusion based on reference point cloud dataset aircraft positions.

(b) Occlusion based on target point cloud aircraft positions.

Figure 5.26.: Visualization of the second epoch of the synthetic dataset (see Figure 5.9). Points are color-coded according to the two occlusion feature values. Red indicates occlusion; green indicates visibility under the corresponding boolean condition.

Before applying occlusion-related rules, it is helpful to identify whether a point belongs to a wall. To make this distinction, the stability factor (defined in Equation 5.1) is used. Figure 5.27 visualizes this: points with a stability factor greater than 0.5 are shown in one color, while those with 0.5 or less are shown in another. This threshold effectively differentiates roof from wall surfaces.

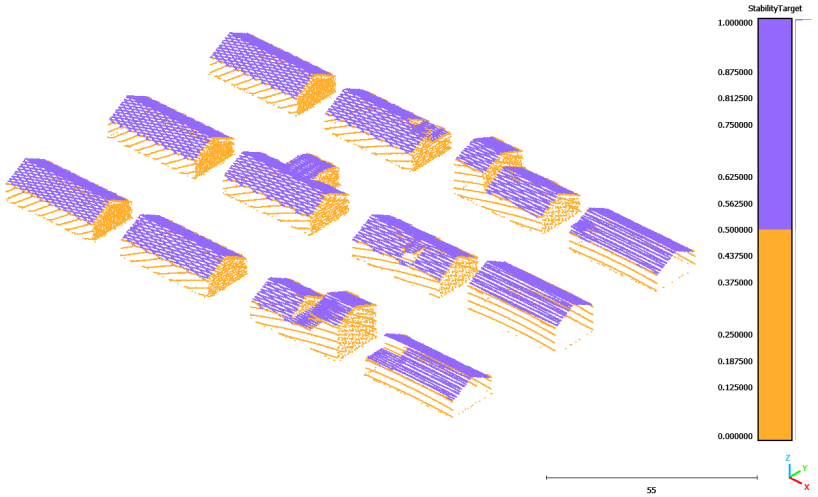


Figure 5.27.: stability factor threshold visualization. Points with a value above 0.5 are separated from those with a value of 0.5 or below. This distinction helps to differentiate between roof and wall surfaces.

To improve the reliability of method predictions, the probability scores from the first RF model are adjusted based on both the occlusion type and the stability factor of the point. This adjustment is only applied to points with an initial probability score greater than 0.5. The adjustment rules are as follows:

1. **Visible:** The score remains unchanged.
2. **Dynamic occluded with a stability factor $>$ 0.5:** The score remains unchanged.
3. **Static occluded with a stability factor $>$ 0.5:** The score is reduced to 0.5.
4. **Static occluded with a stability factor $\leq$ 0.5:** The score is reduced to 0.5.
5. **Dynamic occluded with a stability factor $\leq$ 0.5, and more than 20% of neighboring points are static occluded:** The score is reduced to 0.5. The 20% threshold was determined through visual inspection. In many wall regions, most points are incorrectly marked as occluded, while only a small portion is correctly identified as visible. Therefore, a relatively low threshold (20%) is used to trigger the adjustment.

These rules aim to reduce false positives in regions where visibility is compromised due to occlusion, particularly on wall surfaces where scanner line-of-sight is often blocked.

An overview of the rule-based adjustment process is shown in Figure 5.28.

5.8. Application to Real Datasets (AHN and Rotterdam)

This section details how the trained random forest (RF) classifiers and occlusion logic are applied to the national height model of the Netherlands (AHN) and Rotterdam datasets. This application serves three main purposes:

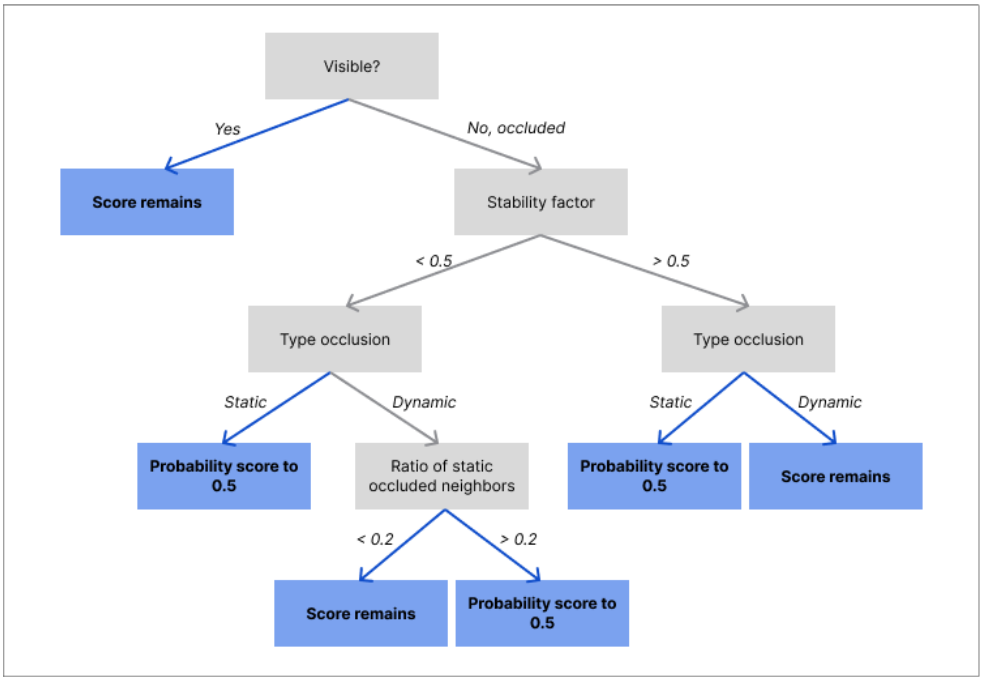


Figure 5.28.: Overview of conditions used to adjust the probability scores based on occlusion. Adjustments are only made for points with an initial probability score above 0.5. **dynamic occluded** refers to points that become occluded due to changes between epochs, while **static occluded** refers to permanent obstructions present in both epochs. A stability factor above 0.5 generally indicates roof surfaces; lower values correspond to walls.

- It demonstrates the performance of the trained models on real-world data.
- It highlights limitations of the current approach, providing insights that can inform improvements to the synthetic training data, making it more representative of datasets like national height model of the Netherlands number 4 (AHN4), national height model of the Netherlands number 5 (AHN5), and Rotterdam.
- It illustrates the generalizability of the algorithm across two distinct data sources.

Furthermore, this section outlines the method used to group light detection and ranging (LiDAR) points into individual building, a crucial step for building-level change detection.

Filtering Building Points in AHN

Because the classification differs per dataset, the first step is to group LiDAR points that belong to the same building. This is achieved using a raster-based clustering technique. The area is divided into a 2D grid of raster cells, each representing a fixed ground area based on a chosen resolution. Each cell stores the list of points it contains and a boolean flag indicating whether any of them are labelled as building points.

Clustering is performed by scanning each cell. For each unvisited cell containing building points, a flood-fill algorithm (using depth-first search) identifies all neighboring building-containing cells. These connected cells form a building cluster and are assigned a unique identifier, `BuildingGroup`. Two cells are considered connected if they share a horizontal or vertical edge.

The raster resolution is defined as four times the average intra-point distance of the dataset (see Section 5.1). To ensure consistent clustering across different epochs, the resolution is based on the largest intra-point distance across all datasets. This guarantees alignment between rasters, which is essential for comparing buildings across time.

A cell is marked as `ContainsBuildingPoints = true` if it contains at least one point labelled as a building. This approach compensates for inconsistencies in labelling, especially in AHN5, where wall surfaces are not labelled as building. Including such points ensures a more complete building representation. Figure 5.29 and Figure 5.30 illustrate the results of this clustering on the AHN4 dataset.

Matching and Change Classification

After clustering, each building cluster in the target point cloud dataset is matched to a corresponding cluster in the reference point cloud dataset. If no suitable match is found, the cluster is assumed to represent a newly constructed building. In that case, all points within its raster cells are labelled as changed with maximum certainty (score = 1).

Because of variations in point density, noise, and occlusion between epochs, buildings may appear split or merged. As a result, the matching strategy must accommodate both one-to-many and many-to-one relationships between clusters.

In some cases, a building extension present in the reference point cloud dataset may be removed in the target point cloud dataset. To handle such cases, the method also considers

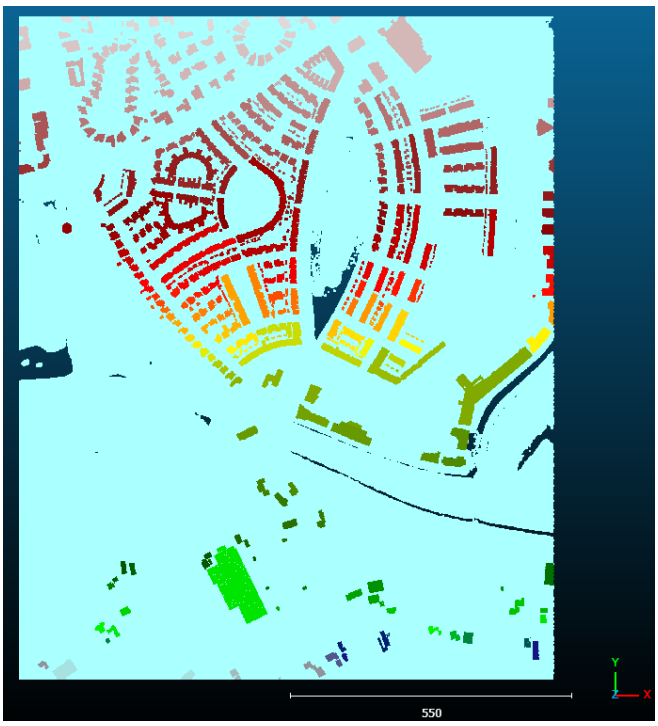


Figure 5.29.: 2D raster clustering result for Area 5 from Figure 4.1. Each color represents a unique building cluster. Light blue cells contain no building points.

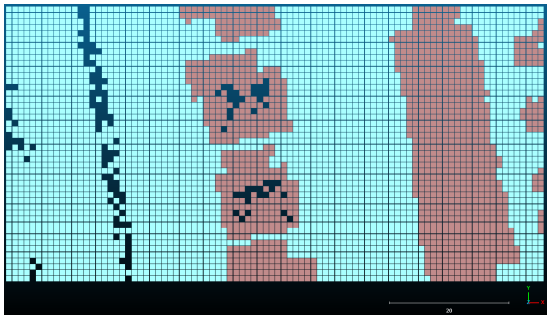


Figure 5.30.: Raster visualization of poorly captured buildings in AHN4 (see Figure 6.4a). These cases highlight challenges in clustering when input data quality is low.

points in the target point cloud dataset that are closest to any building points in the reference point cloud dataset, ensuring that removed segments are properly accounted for.

Once clusters are matched, each point within the building group is evaluated using the trained RF classifier. The classifier produces a binary prediction indicating whether a point has changed. For points classified as changed (i.e., with a score above 0.5), an occlusion-aware adjustment is applied to refine the certainty score. This refinement helps reduce false positives, especially in areas affected by occlusion.

Applying to Rotterdam

The Rotterdam dataset only includes ground and non-ground labels, which makes direct clustering infeasible. To overcome this, building outlines from the AHN dataset are reused to define building regions in Rotterdam. Consequently, the Rotterdam dataset is primarily used for qualitative evaluation rather than independent building segmentation.

Assessment

To assess the method's performance, multiple areas within the AHN and Rotterdam datasets are analyzed. These regions feature a variety of change types, allowing for qualitative evaluation of detection accuracy. Special attention is given to solar panels, which must be manually identified due to the lack of labelled data. This is done by visually inspecting roof intensity patterns across epochs, as shown in Figure 6.3.

The evaluation is carried out by visually comparing the predicted changes showed in the point cloud with orthophotos from Beeldmateriaal Nederland. A prediction is considered successful if the method detects a change that can be confirmed through visual inspection.

5.9. Conclusions Implementation

This chapter described the translation of the proposed methodology into a fully operational pipeline for detecting building changes in point cloud data. Beginning with an exploratory analysis of the national height model of the Netherlands (AHN) dataset, a synthetic urban scene was created to simulate airborne laser scanning (ALS) acquisition under varying scanner configurations. This enabled the generation of richly varied and automatically labelled datasets for training and testing.

The key steps of the pipeline included the estimation of the probability of point change using an random forest (RF) classifier, supported by a diverse set of handcrafted features capturing geometric and temporal properties. To improve the reliability of these predictions, an occlusion detection module was developed that distinguishes between dynamic occluded and static occluded. Two alternative strategies were implemented to incorporate occlusion information into the certainty score; their effectiveness will be assessed in the subsequent results chapter.

The complete pipeline was successfully applied to both synthetic and real-world datasets, including the AHN and Rotterdam datasets, confirming the feasibility of the approach. Design decisions, such as feature selection, automated labelling strategies, and the integration of

uncertainty modelling, were made with the goal of ensuring robustness and generalizability across datasets with differing acquisition properties.

Chapter 6 will evaluate the ability of the method to distinguish between changed and unchanged points, as well as its effectiveness in expressing uncertainty through the certainty index.

6. Results and Analysis

This chapter presents the results obtained from the different stages of the implementation. First, in Section 6.1, the outcomes of the exploratory analysis are discussed. These observations helped in tuning the parameters and making key design choices for the method. Then, in Section 6.2, the generated labelled point cloud datasets are evaluated in terms of quality and suitability for training and testing. Section 6.3 presents the performance of the occlusion detection module and illustrates how different types of occlusion were classified. In Section 6.4 and Section 6.5, the performance of the proposed method on the synthetic scenes is shown. The first section focuses on the outcomes of the RF classifier, while the second discusses how the inclusion of occlusion information influences the certainty index. Finally, in Section 6.6, the method is applied to real-world data from the national height model of the Netherlands number 4 (AHN4), national height model of the Netherlands number 5 (AHN5), and Rotterdam datasets, and the results are assessed visually.

6.1. Configuration of Features and Parameters

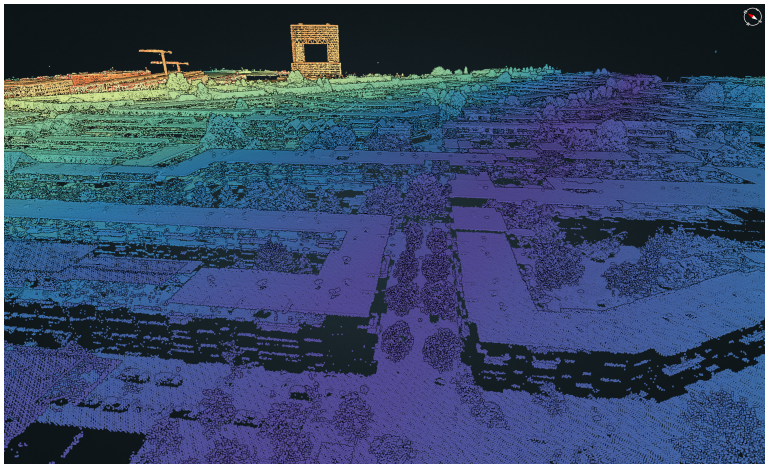
This section presents key findings from the exploratory analysis of the national height model of the Netherlands (AHN) datasets, as introduced in Section 5.1. These observations informed the selection of input features used in the change detection algorithm.

Effect of Distance to the Flightline

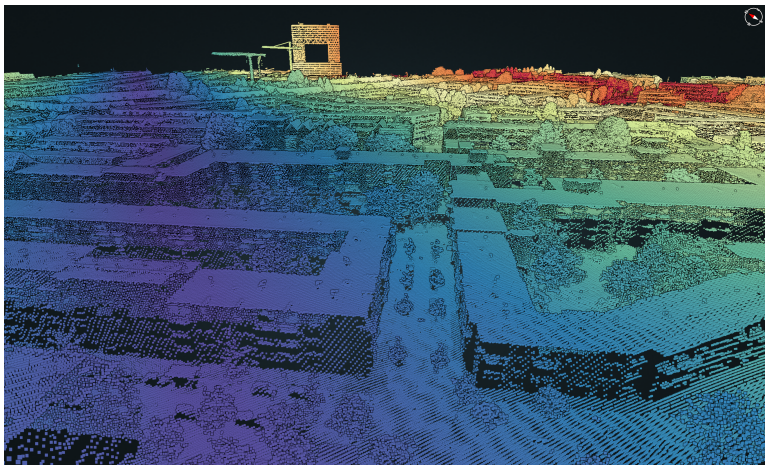
The feature `Distance to the Flightline` is relevant because it helps explain systematic differences in data quality and coverage across the point clouds. This subsection will show the results related to this feature.

As shown in Figure 6.1a, the national height model of the Netherlands number 4 (AHN4) point cloud exhibits missing or incomplete data on vertical surfaces located near the flightline. This effect is less prominent in national height model of the Netherlands number 5 (AHN5), as illustrated in Figure 6.1b. The variation is primarily attributed to differences in scanning technology: AHN4 was collected using a zigzag scanner that tilts forward and backward, while AHN5 employed an oblique light detection and ranging (LiDAR) scanner that better captures vertical elements.

In some areas close to the flightline, a linear or striped pattern is visible in AHN5, as shown in Figure 6.2a. By contrast, Figure 6.2b displays a more irregular point distribution further from the flightline. These patterns may result from limitations in the scanner's geometry or beam divergence. At greater distances, the laser footprint becomes wider, which introduces additional uncertainty in point positioning.

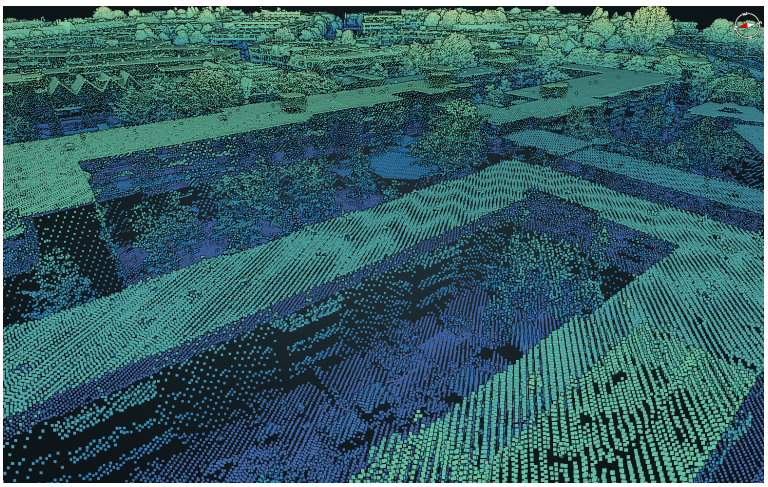


(a) Part of Area 1 in AHN4.

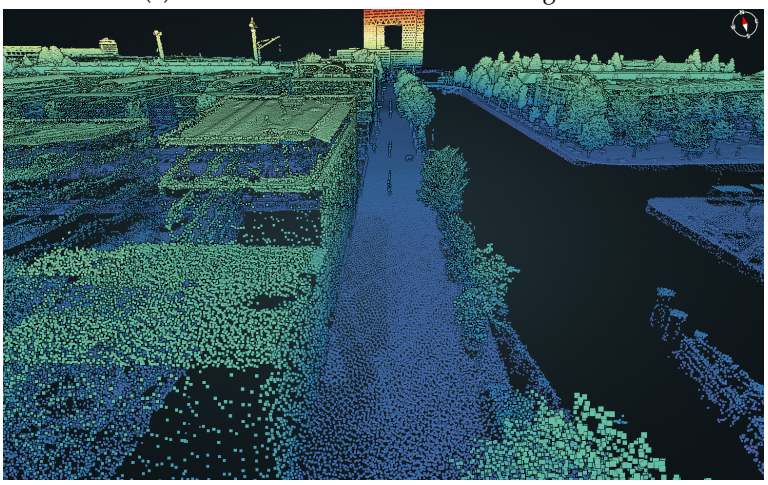


(b) Part of Area 1 in AHN5.

Figure 6.1.: Color indicates proximity to the flightline (purple: close, red: far). Vertical wall coverage near the flightline varies between AHN4 and AHN5.



(a) Part of Area 1 in AHN5 near the flightline.



(b) Part of Area 1 in AHN5 far from the flightline.

Figure 6.2.: Elevation visualized through color gradients.

Point Intensity

Intensity values in the point cloud are another important feature, as they may reveal material differences or changes between acquisition epochs. This subsection will show the results of this point characteristic.

Figure 6.3 shows a case where roofs with solar panels exhibit distinct intensity values compared to the rest of the roof surface. These differences are consistent across both AHN4 and AHN5, suggesting that intensity values can be used to identify added or removed solar panels. In this example, roof surfaces have intensity values between 700 and 1000 in AHN4 and between 1400 and 1900 in AHN5. In contrast, the solar panel areas show lower values, ranging from 100 to 400 in AHN4, and from 600 to 800 in AHN5.

In addition, another case highlights buildings that are poorly captured in AHN4, as illustrated in Figure 6.4. Some houses contain very few points, and those that are present show low intensity values. Although similar issues occasionally occur in AHN5, they are generally less severe. Aerial imagery from Beeldmateriaal confirms that the houses have not changed in structure and appear to have been built around the same time. One possible explanation for the poor capture quality is the presence of water near the buildings, which could have interfered with the laser scanner. This further supports the use of intensity as a diagnostic feature, particularly in identifying regions of poor data quality.

It is important to note that intensity values should only be interpreted within the same dataset, so the relative values. Differences in scanner type, sensor calibration, flight parameters, and environmental conditions result in significant variations between AHN4 and AHN5. As discussed in Section 2.6, these factors make direct comparison of intensity values across datasets unreliable. Intensity is also influenced by the angle of incidence, material reflectance, and distance to the flightline.

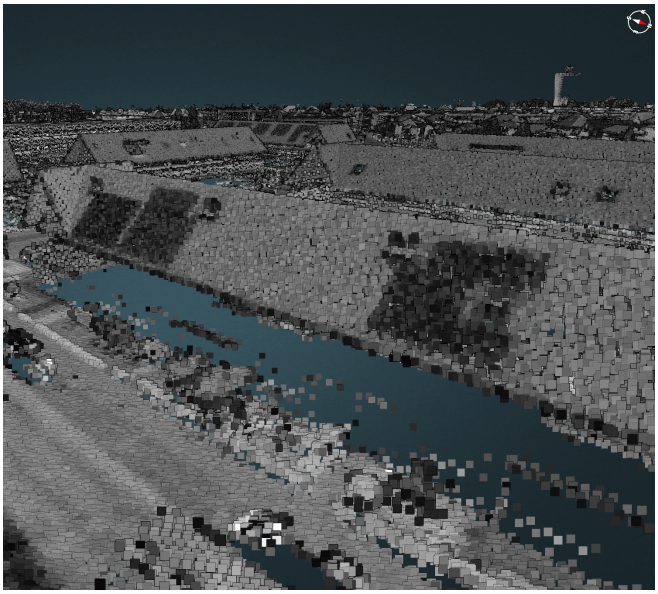
Although intensity demonstrates potential as a discriminative feature, it was ultimately excluded from the automated change detection algorithm. This decision was primarily due to difficulties integrating intensity into the synthetic training data. Further discussion of this limitation is provided in Section 7.1. However, intensity was still used during manual inspection to identify solar panels in both the AHN and Rotterdam datasets, and to qualitatively assess whether the algorithm correctly detected such changes.

Analysis of Nearest Neighbour Distances

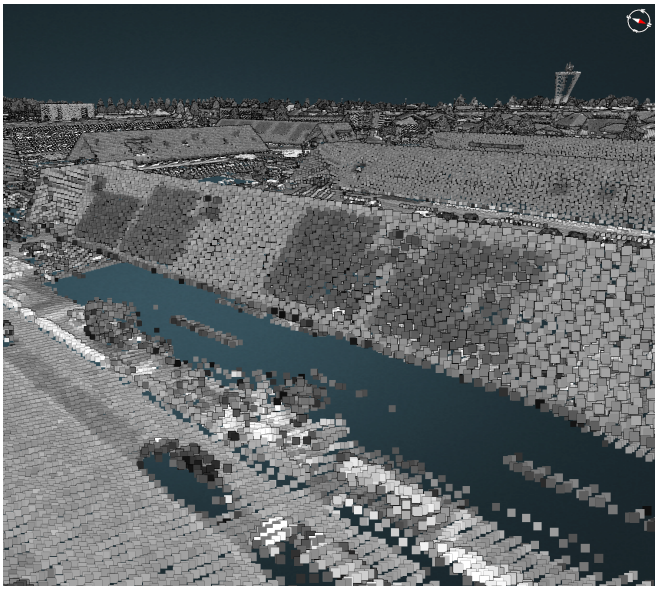
This subsection evaluates four different methods for computing nearest neighbour distances between corresponding point clouds. These methods inform the design of the height difference feature used in the random forest (RF) model.

The following variants were tested:

1. 3D nearest neighbour search, reporting the full 3D Euclidean distance.
2. 3D nearest neighbour search, reporting only the vertical (Z) height difference.
3. 3D nearest neighbour search within a 2D-bounded column (XY search area), reporting the full 3D Euclidean distance.
4. 3D nearest neighbour search within a 2D-bounded column (XY search area), reporting only the vertical (Z) height difference.



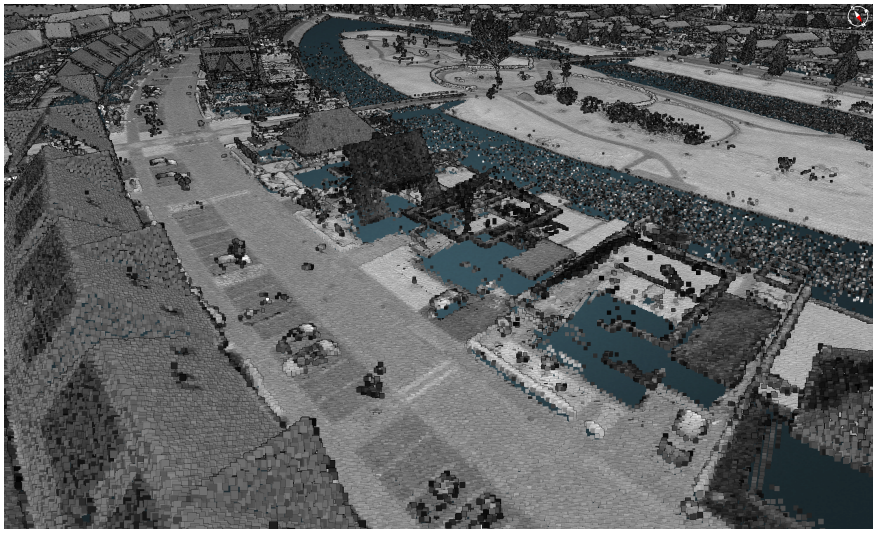
(a) Part of AHN4. Darker shades indicate lower intensity.



(b) Part of AHN5. Darker shades indicate lower intensity.

(c) Street view from Zaltbommel showing solar panels. Source: Google [2021].

Figure 6.3.: Part of Area 2 from Figure 4.1. Differences in intensity suggest the presence of solar panels.



(a) AHN4. Several houses are poorly captured. Darker colors represent lower intensity values.

(b) AHN5. Same area as in AHN4. Color indicates intensity.

Figure 6.4.: Part of Area 2 from Figure 4.1. Comparison of building coverage and intensity in AHN4 and AHN5.

All resulting visualizations are available in Appendix B. An example scene illustrating the differences is shown in Figure B.1, which includes both AHN4 and AHN5. Results from the 3D search methods are shown in Figure B.2, while the 2D methods are presented in Figure B.3.

Based on the comparison, the following observations were made:

- In the area shown (Figure B.1, Figure B.2, Figure B.3), a new apartment building was added. The 3D nearest neighbour search struggles to clearly define the edges of this new building, causing inconsistent distance values around it.
- Nearby buildings have some walls that were not visible in AHN4 but are present in AHN5. Even though these buildings did not change, the 2D nearest neighbour search reports large differences, especially near those newly visible walls. The 3D search is less affected but still picks up some of this noise.
- For small changes like an added balcony, the 3D Euclidean method gives a clearer signal of change. In contrast, 2D methods tend to overestimate the change because they ignore height differences.
- When dealing with vertical surfaces like walls, a 2D neighbour may be found at a very different height, which can lead to large and misleading vertical differences.

These insights directly informed the design of the height difference feature used in the point-level change detection. Specifically, neighbours are first identified within a 0.5-meter radius in the 2D (x, y) plane, after which the closest among these candidates is selected based on the full 3D Euclidean distance.

6.2. Evaluation of Synthetic Datasets as Model Input

This section presents the labelled synthetic datasets and evaluates their suitability for training and testing the change detection algorithm.

A synthetic scene was constructed to replicate the characteristics of the national height model of the Netherlands number 4 (AHN4) and national height model of the Netherlands number 5 (AHN5) datasets. For two selected areas, the real national height model of the Netherlands (AHN) point clouds were compared to synthetic point clouds generated from corresponding 3D building models of the Netherlands (3DBAG)-based scenes. Simulation parameters of the light detection and ranging (LiDAR) sensors were adjusted to closely match those of the real-world acquisitions. The evaluation focused on the following aspects (corresponding figures are provided in Appendix C):

- **Flight Strip Coverage:** The synthetic flight strips were designed to mirror the spatial coverage of the actual AHN flight paths. As shown in Figure C.1 and Figure C.3, both synthetic and real datasets encompass the same areas. In the real data, the flight paths exhibit minor deviations due to natural flight dynamics, whereas the simulated paths are perfectly linear. Despite this difference, the overall coverage is sufficiently comparable for training purposes.

- **Wall and Roof Coverage:** Figure C.2 and Figure C.4 compare the coverage of vertical and horizontal surfaces. Discrepancies are visible particularly near surface edges, where real data often exhibits missing points due to occlusions or scanner range limitations. The synthetic 3DBAG-based models, in contrast, lack finer architectural details and thus provide simplified geometric representations. As a result, alignment with the real point clouds is imperfect, particularly in areas affected by structural complexity or occlusions. Furthermore, the spatial distribution of points differs: real data tends to show a more scattered and irregular pattern, while the synthetic data exhibits a structured, grid-like arrangement.
- **Point Density:** Point density was measured in points per square meter over selected rooftop areas. The results show comparable densities between the real and synthetic datasets. This outcome must be the case, as both datasets were simulated using identical flight trajectories and similar sensor parameters.

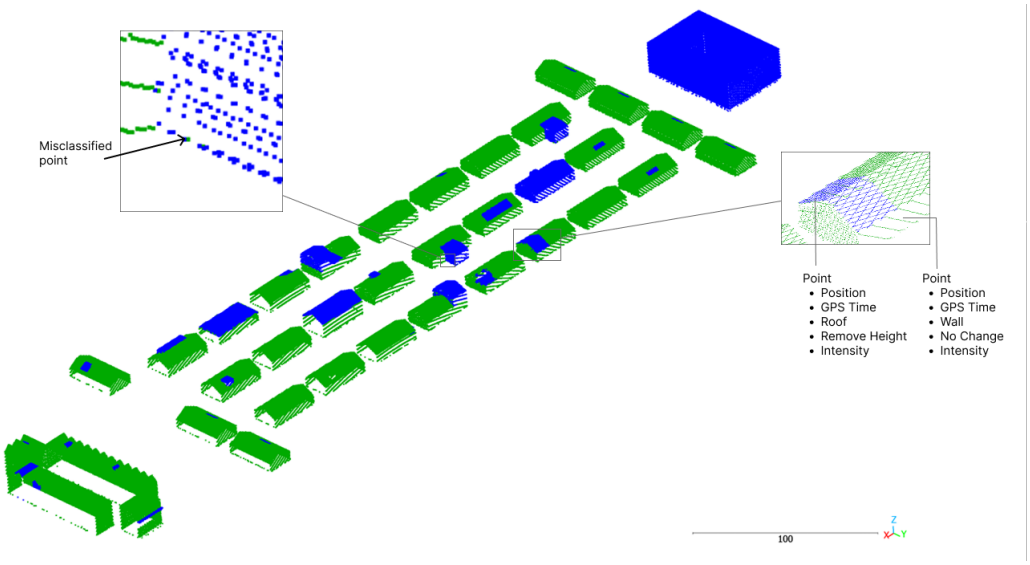


Figure 6.5.: Part of the labelled synthetic dataset. Green points represent unchanged areas; blue points are labelled as changed. The inset highlights an example of a potentially mislabelled point. The right-hand panel shows relevant point attributes.

6.3. Assessment of Occlusion Detection Accuracy

A key component of this research is the integration of occlusion detection into the change detection methodology. This section presents the results of the occlusion detection method and evaluates its performance.

Before examining the outcomes, it is important to recall the two types of occlusion introduced earlier (see Figure 4.4), as well as the two Boolean variables implemented to differentiate them (see Section 5.6).

Results Boolean 1: Occlusion Using Aircraft Positions from the reference point cloud dataset

This subsection evaluates the occlusion detection method based on the scanner positions from the reference point cloud dataset (Boolean 1).

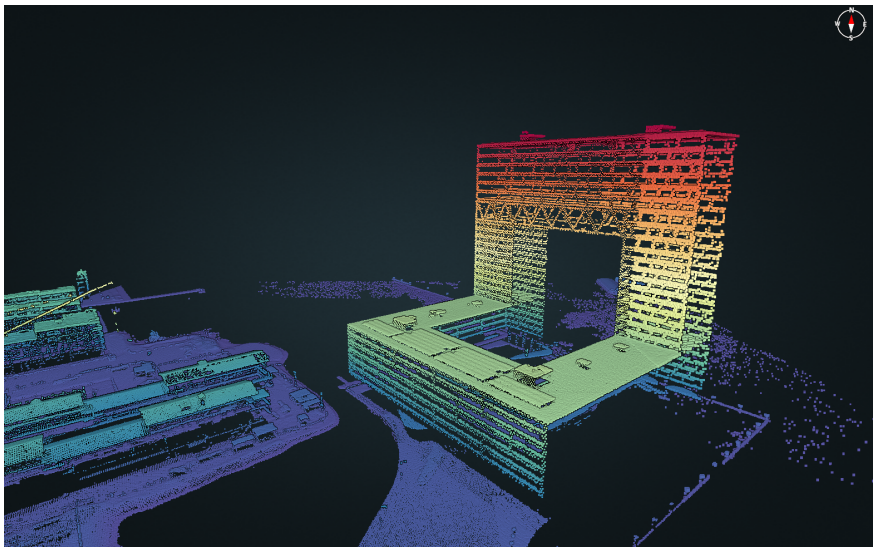
To assess the correctness of Boolean 1, the visibility of points in one epoch is compared to the actual point coverage in the second epoch. Specifically, points from the earlier dataset (reference point cloud dataset) are tested for visibility using the aircraft positions from that same dataset. These visibility predictions are then compared to the second dataset (target point cloud dataset) to verify consistency.

If a structure (e.g., wall or roof) is visible in the reference point cloud dataset, the same part should ideally be marked as visible (green) in the target point cloud dataset. Conversely, if an area is not captured in the reference point cloud dataset, e.g., due to occlusion by another object, then the corresponding area in the target point cloud dataset should be marked as not visible (red). This comparison helps validate whether the occlusion detection method accurately identifies regions that were blocked from the scanner's line of sight.

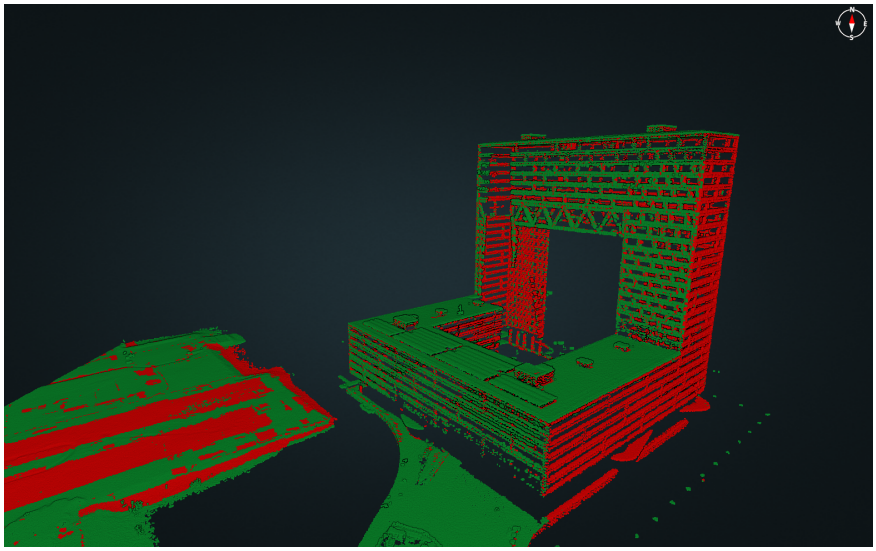
Figure 6.6b displays visibility results for national height model of the Netherlands number 4 (AHN4), with coloring derived from the scanner positions in national height model of the Netherlands number 5 (AHN5). Green points are marked as visible, red as occluded. For reference, the original AHN5 point cloud is shown in Figure 6.6a. Visual comparison between the two figures confirms the method's validity. For instance, missing wall segments in AHN5 align with red-labelled occluded areas in AHN4, while walls present in both epochs appear green. Overall, the method performs well. Ground points in AHN4 that were later blocked by new buildings in AHN5 are correctly labelled as occluded. The "Pontsteigergebouw," a large apartment complex, also demonstrates effective detection, its unscanned walls in AHN5 are accurately marked red in AHN4. Some inaccuracies exist, such as visible walls incorrectly labelled as occluded, likely due to minor inconsistencies in the point cloud, including multi-layer wall representations.

Figure 6.7a and Figure 6.7b show similar results for another test area, this time using the zigzag scanning pattern from AHN4. Here, the red-labelled occluded points in Figure 6.7b match the missing structures in the AHN4 scan (Figure 6.7a). For instance, a partially captured house wall in the foreground is correctly divided between visible and occluded regions, confirming the method's performance in this context.

Figure 6.8a and Figure 6.8b show results from the synthetic dataset. The first figure presents the reference point cloud dataset, while the second illustrates the visibility status of the target point cloud dataset based on scanner positions from the reference point cloud dataset. While

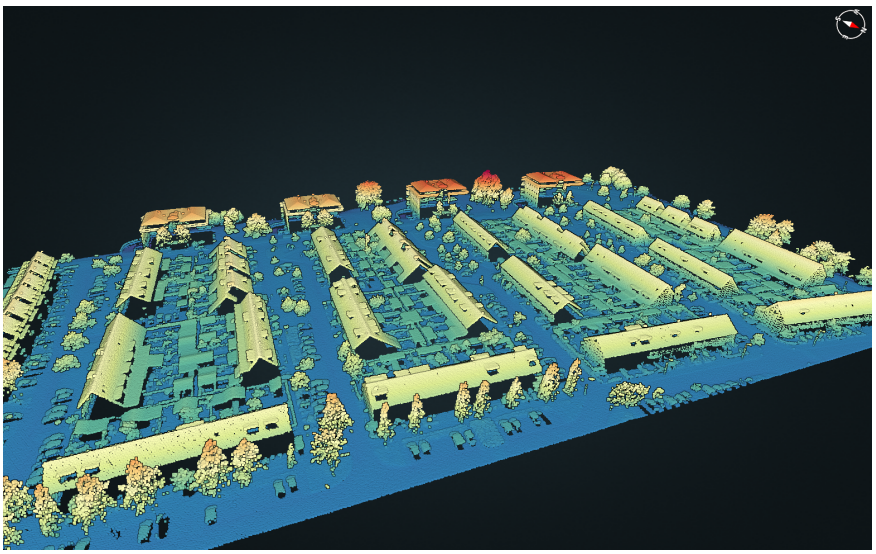


(a) Houthaven, Amsterdam visualized using AHN5. Coloring represents elevation.

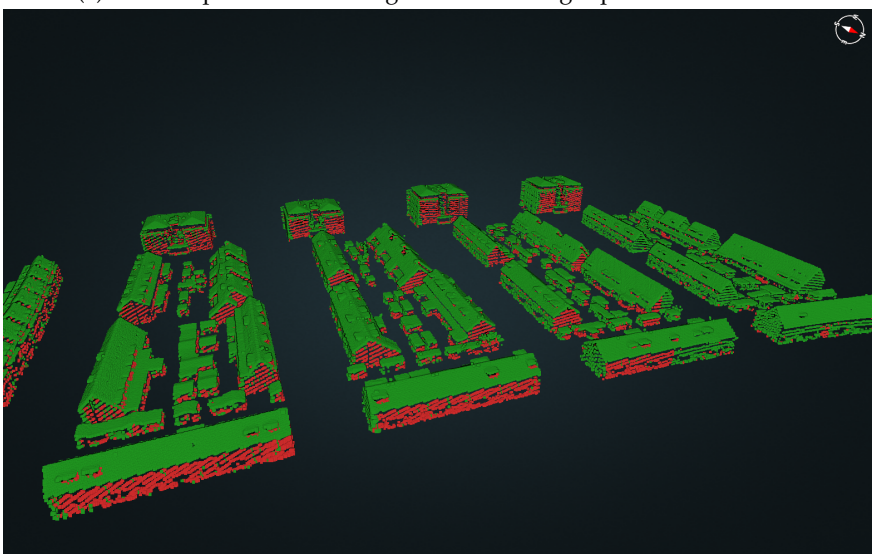


(b) Houthaven, Amsterdam visualized using AHN4. Points are colored based on occlusion status in AHN5. Green = visible, red = occluded.

Figure 6.6.: Occlusion detection results using AHN5 as the reference point cloud dataset. Red points in AHN4 indicate areas not visible from scanner positions in AHN5.



(a) Nootdorp visualized using AHN4. Coloring represents elevation.



(b) Nootdorp visualized using AHN5. Points are colored by occlusion status in AHN4. Green = visible, red = not visible.

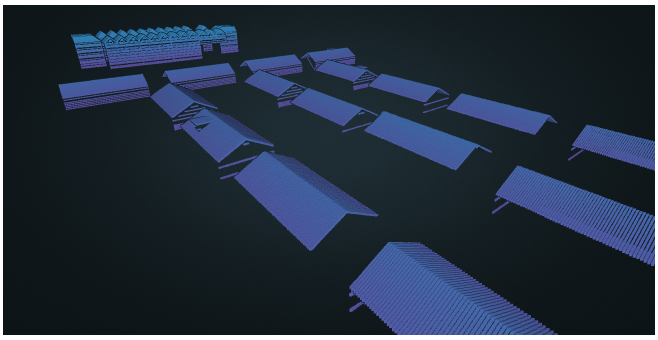
Figure 6.7.: Occlusion detection results using AHN5 as reference point cloud dataset. Red points in AHN4 show areas not visible in AHN5.

roof surfaces are generally well detected, many walls that should be visible are incorrectly marked as occluded.

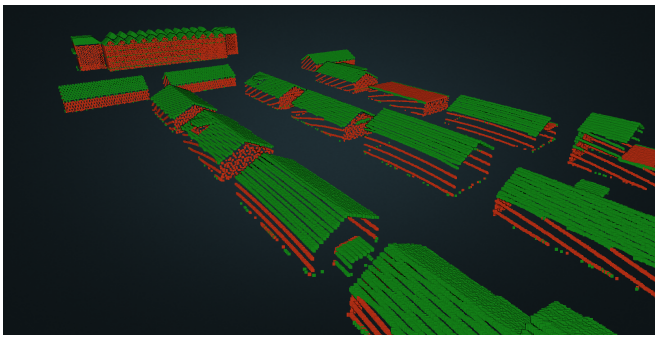
There are two primary reasons for this difference:

1. The method of calculating scanner positions differs for synthetic data compared to real datasets (see Section 5.6).

2. In this region, the incidence angle between scanner and wall surfaces is small. At shallow angles, parts of a wall may block other segments of the same wall, even though they are theoretically within scanner range.



(a) Synthetic scene, reference point cloud dataset. Coloring shows elevation.



(b) Synthetic scene, target point cloud dataset. Points are colored by occlusion status in the reference point cloud dataset. Green = visible, red = occluded.

Figure 6.8.: Occlusion detection results in a synthetic scene. Red points in the target point cloud dataset denote areas not visible from the scanner positions in the reference point cloud dataset.

Results Boolean 2: Occlusion Using Aircraft Positions from the target point cloud dataset

This subsection evaluates the occlusion detection method based on the scanner positions from the target point cloud dataset (Boolean 2).

The second occlusion check (Boolean 2) assesses the visibility of points from the first epoch based on the aircraft positions of the second epoch. This approach differs from Boolean 1, as it uses the scanner positions that actually captured the target point cloud dataset.

In most cases, points from the first epoch should be visible (green), since the second epoch's aircraft positions were able to observe them. Only areas affected by real changes, such as newly introduced obstructions, should be marked as not visible (red).

Figure 6.9 illustrates the occlusion detection results for a selected area in Nootdorp using the national height model of the Netherlands (AHN) dataset. Green points are considered visible, while red points are classified as occluded. The results show that several wall surfaces are incorrectly labelled as occluded, even though they are clearly captured in both datasets. This may be due to the aircraft's flight path being almost directly above the buildings.

Figure 6.9b highlights a similar issue on roof surfaces, where some areas are incorrectly marked as occluded, despite no actual change having occurred. This is likely the result of local misalignment between the two datasets, as nearby houses do not display the same error.

In both cases, the misclassified wall and roof points, the occlusion threshold parameter may be too strict. This threshold defines the maximum number of points allowed between the aircraft and the surface before a point is marked as occluded. If set too low, it can cause visible surfaces to be incorrectly classified as hidden.

Results for the synthetic dataset are shown in Figure 6.10. In this area, no actual changes occurred between the epochs, meaning all points should ideally be marked as visible. However, all west-facing walls are labelled as occluded, while most south-facing walls are correctly marked as visible. This difference is likely due to the geometry of the scan angle. When the angle between the scanner and a wall is small, segments of the wall itself may occlude neighboring points, resulting in incorrect classifications.

6.4. Performance of Change Probability Estimation on Synthetic Data Using a Random Forest Model

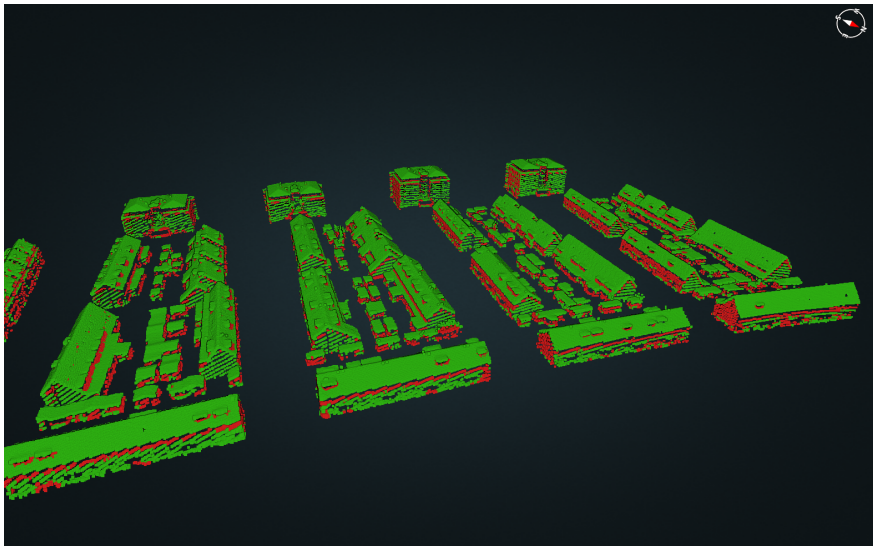
This section presents the performance of the random forest (RF) model on the synthetic dataset. The model assigns each point a probability score between 0 (very likely no change) and 1 (very likely a change). In the next section, occlusion-aware results will be discussed.

As explained in Section 5.5, five different feature and training set configurations were evaluated. Table 6.1 shows the classification accuracy for each change type across these five configurations.

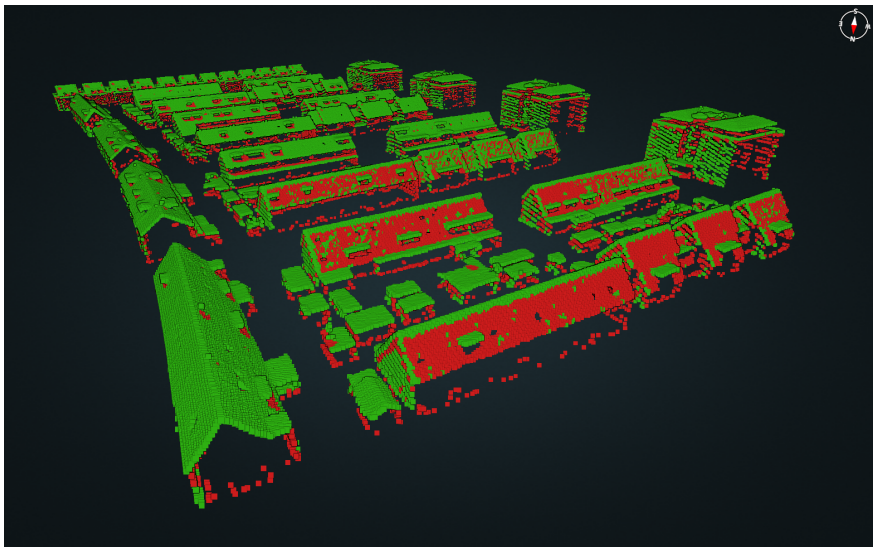
Feature importance for Set 1 is shown in Figure 6.11. The most informative features are the stability factor difference and height difference.

Although tilt angle and 3D density were intended to help distinguish between roof and wall surfaces, Figure 5.11 shows they do not perform this task effectively. In contrast, the stability factor (Figure 5.13) proves more reliable. Since excluding tilt angle and 3D density does not reduce accuracy, they were removed for simplicity.

Furthermore, Table 6.1 shows that excluding most dataset-level features generally improves accuracy. These features are not informative (Figure 6.11) and may even introduce noise. A



(a) Part 1.



(b) Part 2.

Figure 6.9.: Boolean 2 occlusion results for Nootdorp using AHN5. Green points are visible to the scanner; red points are incorrectly labelled as occluded due to possible misalignment or threshold settings.

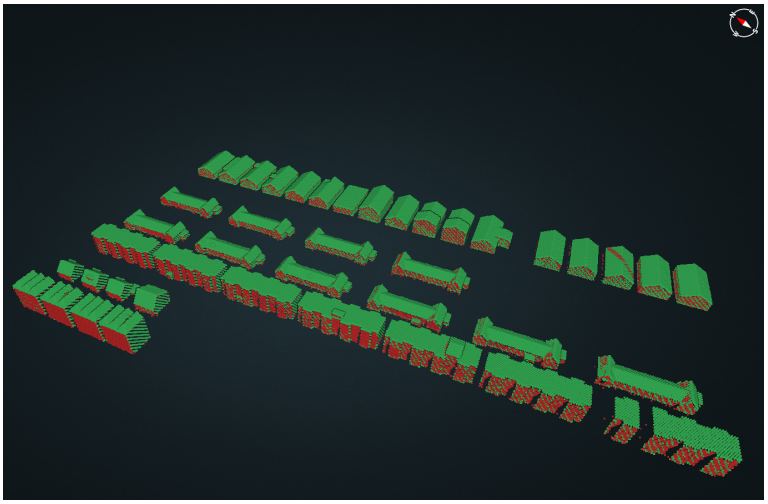


Figure 6.10.: Boolean 2 occlusion results for the synthetic scene. No structural changes occurred in this area. Red points are marked as occluded, and green points are visible to the scanner. Errors appear primarily on west-facing walls due to low-angle occlusion.

Change Type	**Total Points**	**Set 1**	**Set 2**	**Set 3**	**Set 4**	**Set 5**
Balcony Add/Remove	3591	85.6%	86.9%	87.0%	87.0%	86.9%
Dormer Add/Remove	13663	89.4%	89.3%	89.0%	89.3%	88.6%
Height House Add/Remove	15000	93.8%	94.7%	94.9%	94.9%	95.2%
House Add/Changed	15000	89.9%	91.2%	88.5%	89.0%	89.2%
SP04 Add	6837	48.1%	53.1%	53.4%	53.0%	7.6%
SP08 Add	6696	56.6%	59.7%	60.4%	60.9%	19.2%
SP12 Add	6933	67.1%	70.4%	70.2%	71.2%	48.8%
SP16 Add	3117	69.5%	70.4%	71.4%	70.4%	59.9%
Width House Add/Remove	7706	93.9%	94.8%	95.5%	96.5%	96.4%
No Change	105000	94.5%	94.4%	94.4%	94.3%	97.5%
Total Accuracy		88.4%	88.5%	88.2%	88.6%	87.1%

Table 6.1.: Classification accuracy for each change type using Feature and Input Sets 1 to 5. The percentage indicates the proportion of points of each change type that were correctly classified. Feature Set 1 includes all features. Set 2 excludes tilt angle and 3D density. Set 3 removes most dataset-level features, keeping only density difference. Set 4 excludes all dataset-level features. Set 5 excludes training data containing solar panel changes of 4 cm and 8 cm thickness. The ground truth is based on the labelled synthetic dataset.

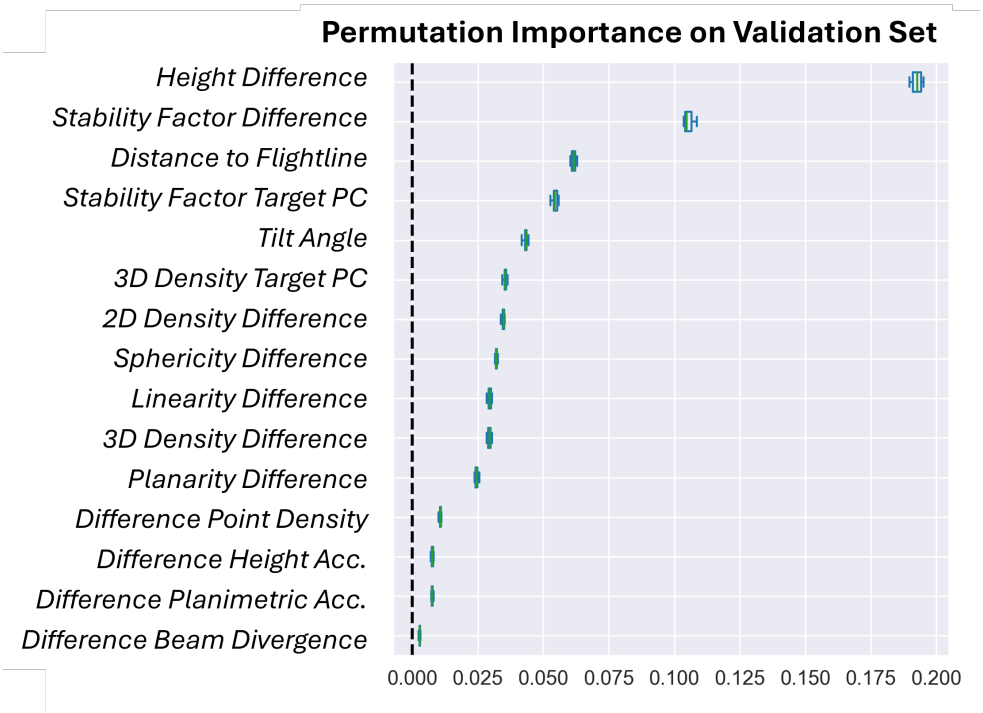


Figure 6.11.: Permutation importance of point- and dataset-level features in the RF model trained on Set 1.

likely reason is that they lack per-point variability, which limits the model's ability to learn from them. Therefore, they are omitted in the final model.

Two final models were selected:

1. Model 1: Trained with all change types.
2. Model 2: Trained without solar panel changes of 4 cm and 8 cm. These changes are still used in testing.

Model hyperparameters were optimized using RandomizedSearchCV. Final settings:

- `Number of trees`: *393* (Set 1), *121* (Set 2).
- `Max tree depth`: *19* (Set 1), *5* (Set 2).

Table 6.2 reports precision and recall for both classes using the synthetic test dataset. Precision and recall scores on the test dataset show that Model 1 generally performs better than Model 2. In particular, the recall of Model 2 for detecting changed points is low. This means that many actual changes were incorrectly classified as unchanged. One possible reason is that Model 2 was not trained on examples of small solar panel additions (4 cm and 8 cm), even though these were part of the test dataset.

Metric	**Unchanged Class (0)**		**Change Class (1)**	
	Model 1	**Model 2**	**Model 1**	**Model 2**
Precision	0.87	0.76	0.93	0.98
Recall	0.95	0.99	0.81	0.58

Table 6.2.: Precision and recall for changed and unchanged points for both models.

The following figures compare Model 1 and Model 2 across three synthetic scenes:

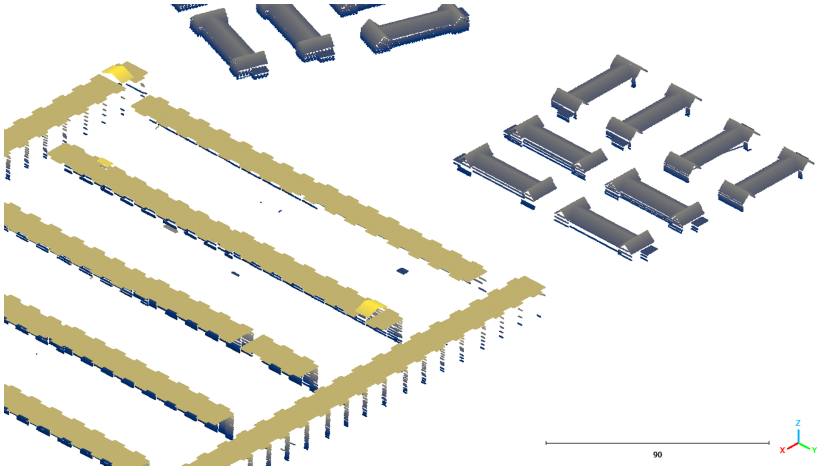
- Figure 6.12: General performance and sensitivity to different change types.
- Figure 6.13: Influence of flight path distance on prediction.
- Figure 6.14: Effects of occlusion by tall buildings.

F1 scores for both classes across five synthetic scenes are listed in Table 6.3. Each scene uses different scanner parameters. Descriptions of the scenes can be found in Table 5.3. While Table 6.3 shows that Model 2 often achieves higher F1 scores per scene, these scores do not provide the full picture. So figures will be shown below.

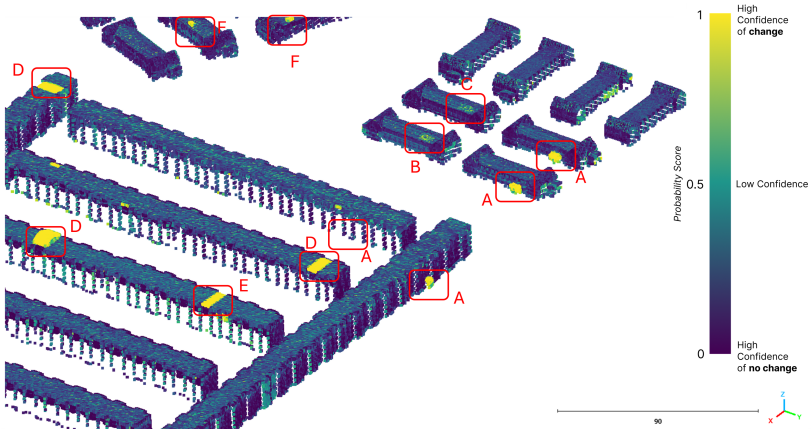
Scene	**Description**	**F1 Score – Unchanged Class**		**F1 Score – Change Class**	
		Model 1	**Model 2**	**Model 1**	**Model 2**
Scene 1	Same scanner settings	0.99	1.00	0.70	0.83
Scene 2	Different pulse repetition rate (PRR)	0.79	0.82	0.11	0.11
Scene 3	Different beam divergence	0.97	0.99	0.47	0.64
Scene 4	Different flightlines	0.97	0.99	0.50	0.73
Scene 5	Different scanner accuracy	0.96	0.98	0.51	0.70

Table 6.3.: F1 scores for both models across synthetic scenes with different scanner settings.

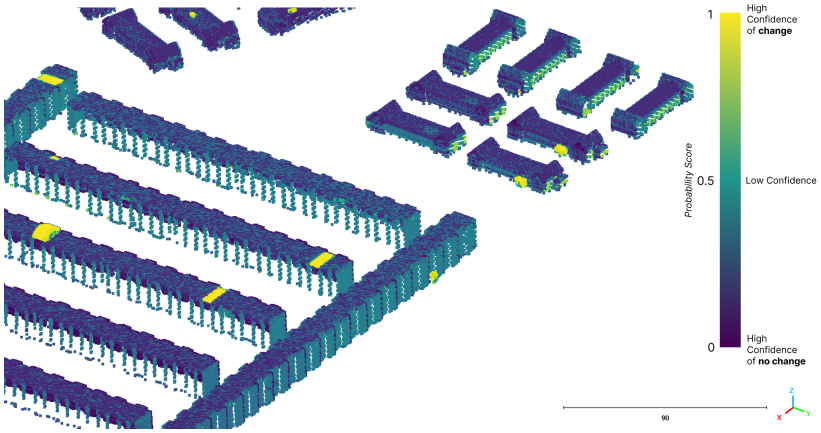
In the figures, both models identify similar types of changes. Additions of solar panels with heights of 8 cm and 12 cm are detected by Model 1 and are partially visible in Model 2. There is also a notable difference in the predicted probability scores. Model 1 tends to assign



(a) Reference point cloud dataset. Coloring based on height of the point.

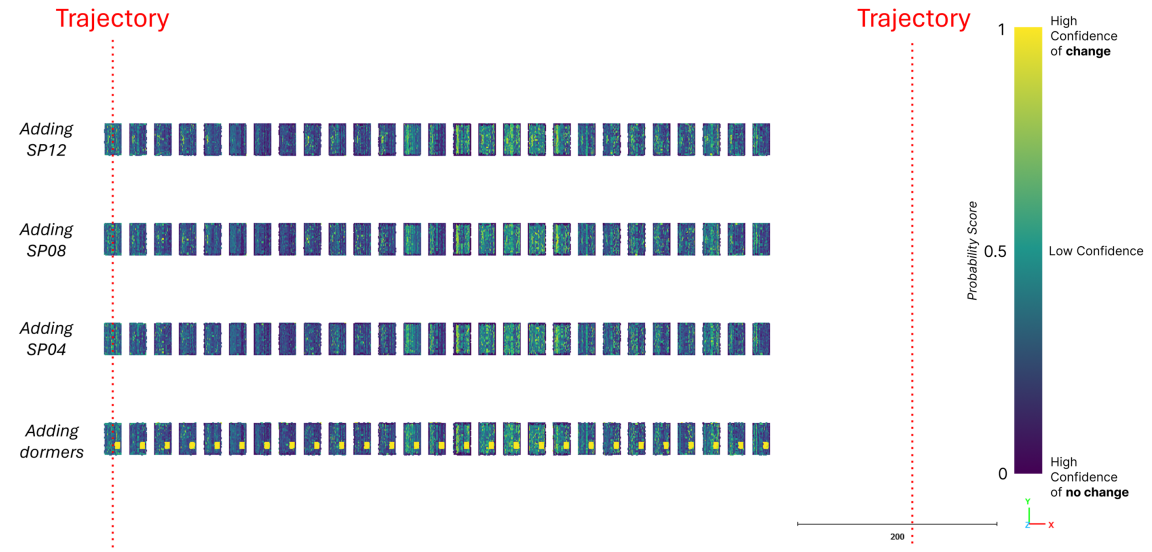


(b) Part of the synthetic scene, colored by the probability score output from RF Model 1. The visualized changes include: (A) house extension or removal, (B) addition of a solar panel with a thickness of 12 cm, (C) addition of a solar panel with a thickness of 16 cm, (D) height increase or decrease, (E) construction of a new house, and (F) addition of a dormer.

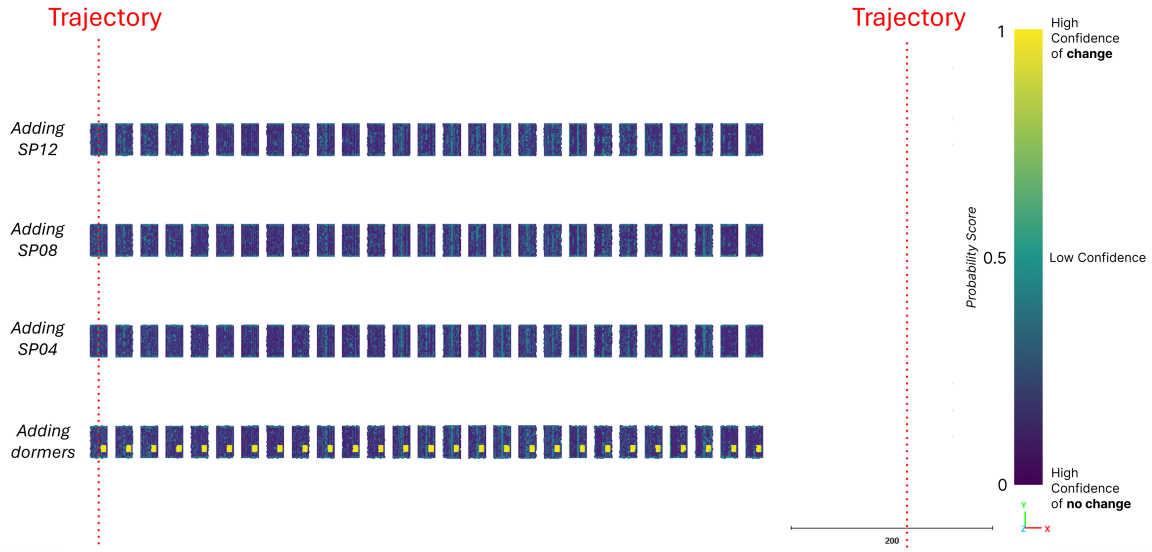


(c) Part of the same synthetic scene, colored by the probability score from RF Model 2.

Figure 6.12.: Probability scores from RF classifiers. Yellow indicates high change probability; dark blue indicates low. Based on national height model of the Netherlands number 4 (AHN4)/national height model of the Netherlands number 5 (AHN5) settings. Subfigures show the same area for Model 1 and Model 2, highlighting changes A–F.

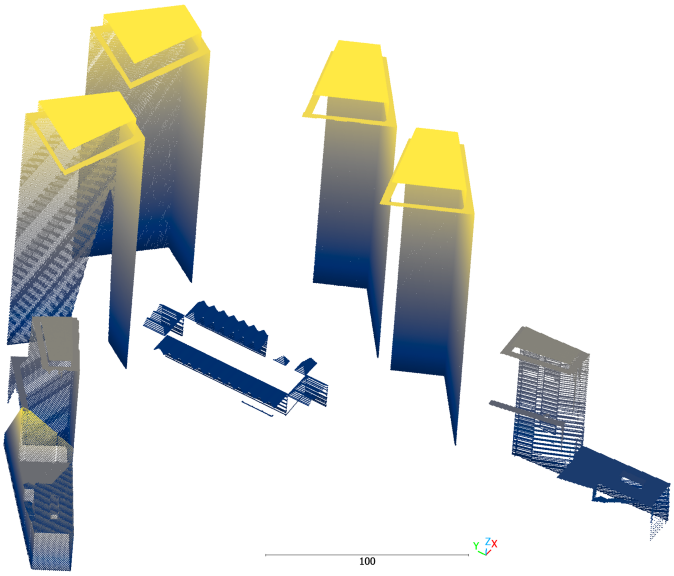


(a) Probability scores from RF Model 1.

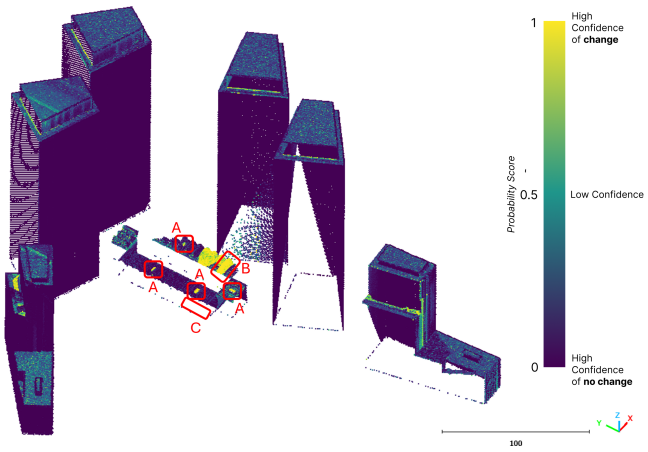


(b) Probability scores from RF Model 2.

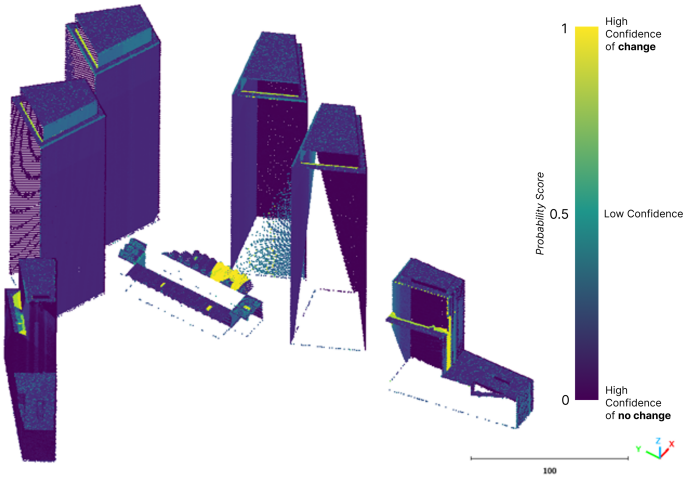
Figure 6.13.: Probability scores for identical house types at varying distances from the flight-line. Scanner settings match AHN4/AHN5. Results illustrate the influence of scanner position on predictions.



(a) The reference point cloud dataset. Coloring based on height of the point.



(b) Part of the synthetic scene, colored by the probability score from RF Model 1. The visible changes include: (A) addition of a dormer, (B) construction of a new house, and (C) removal of a balcony.



(c) Part of the same synthetic scene, colored by the probability score from RF Model 2.

Figure 6.14.: Probability scores from RF classifiers for a synthetic scene with tall buildings. Occlusion can result in false positives due to missing data.

probabilities around 0.5 for unchanged roof surfaces. In Model 2, probability scores around 0.5 occur more frequently on unchanged walls, particularly when those areas are occluded in one of the epochs. According to Figure 6.13, the performance of Model 1 declines as the distance from the flightline increases. In those areas, Model 1 tends to classify more unchanged points as changed. Both models have difficulty with occlusion but that will be covered when incorporating occlusion. Model 2 will be used to test on the real datasets.

Comparison Between Scenes

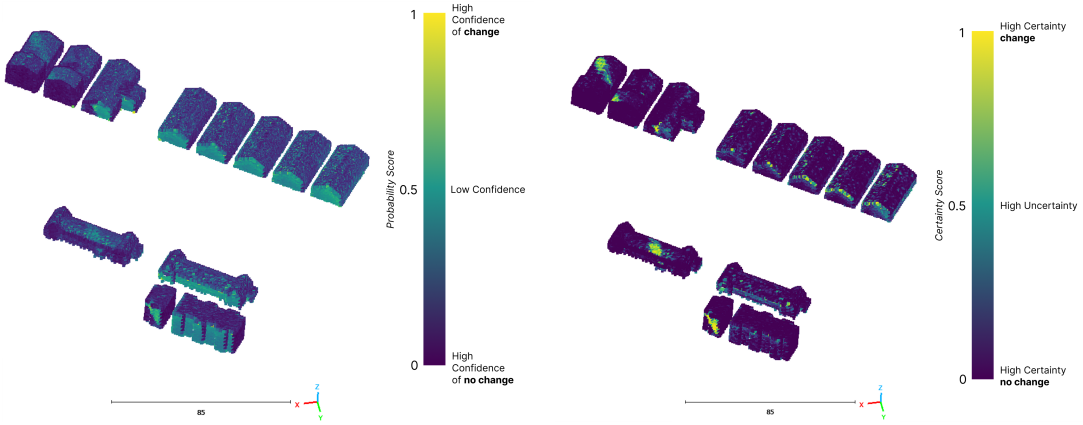
Five synthetic scenes were generated with varying scanner configurations, as shown in Table 5.1. The following section presents a comparison between these scenes.

When comparing across scenes, Scene 2 shows the weakest performance. This scene had a different PRR, which appears to affect the quality of the point cloud and the model's ability to detect changes. This is consistent with Figure 5.21, where the house in Scene 2 was poorly captured. The best results are found in Scene 1, where the scanner settings were identical between the two epochs. This scene also had minimal occlusion, contributing to the higher performance. Scenes 3, 4, and 5 show similar performance based on the F1 score.

6.5. Effect of Occlusion-Based Uncertainty on Synthetic Scene

As described in Section 5.7, two strategies were explored to incorporate occlusion information into the change detection approach.

In Model 1, occlusion was introduced as an additional input feature for a second random forest (RF) classifier. However, this approach proved ineffective, as the classifier is designed to predict categorical labels, such as change or no change, while occlusion instead provides information about prediction confidence or uncertainty. As such, including occlusion as a feature did not improve classification accuracy. The second RF model also included derived features such as ratio probability and ratio occlusion. Including these features resulted in predictions shifting further toward the extremes (0 or 1), leading to overconfident classifications. Since the initial predictions from the first model were already near these extremes, this effect was amplified, decreasing the model's ability to reflect uncertainty in ambiguous regions, particularly those affected by occlusion or data inconsistencies. Consequently, this approach was not adopted. Figure 6.15 illustrates this effect in a scene with no real changes.



(a) Probability score from RF 1.

(b) Certainty score from RF 2.

Figure 6.15.: Comparison between RF 1 and RF 2 in a scene without actual changes. Yellow indicates high certainty of change; dark blue indicates high certainty of no change.

The results indicate that incorporating occlusion directly as a model feature leads to overconfident predictions that do not accurately represent uncertainty in some regions. To mitigate this limitation, an alternative approach, Model 2, was developed.

Model 2 uses two criteria to determine whether a prediction should be modified to reflect uncertainty: the occlusion type and the point's stability factor. If these indicate a high level of uncertainty, the probability score is adjusted to a certainty score of 0.5, representing maximal uncertainty. This process is illustrated in Figure 5.28.

To demonstrate the application of this method in the synthetic scene, two examples are provided below:

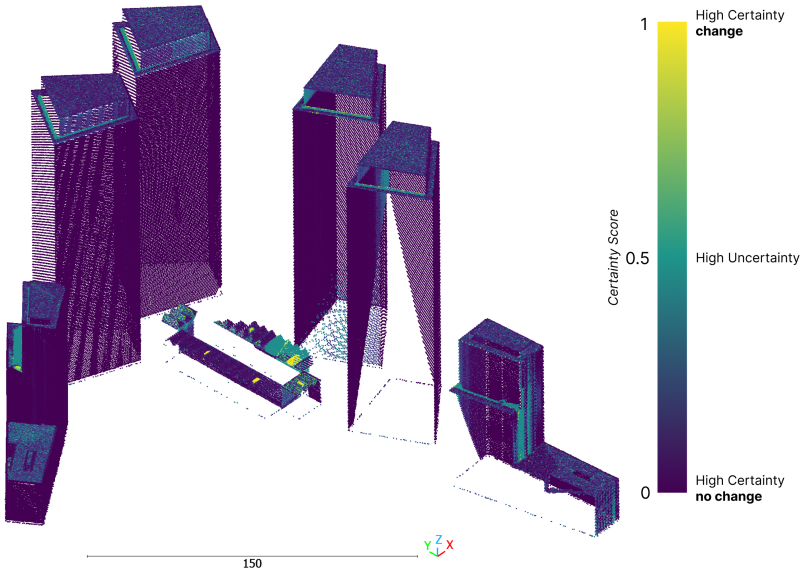
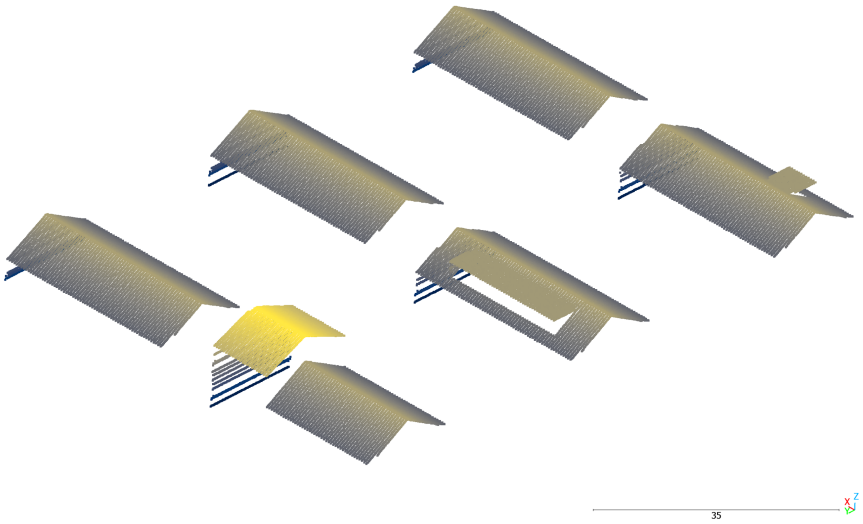
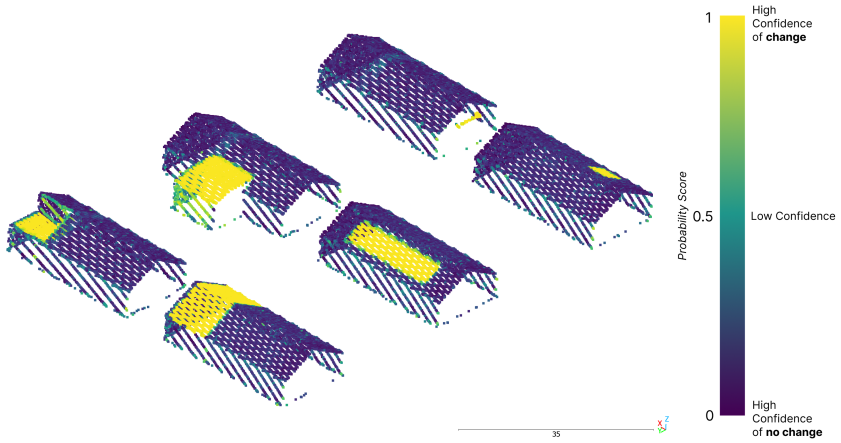


Figure 6.16.: Section of the synthetic scene. Same area as in Figure 6.14. Colored by the final certainty score.

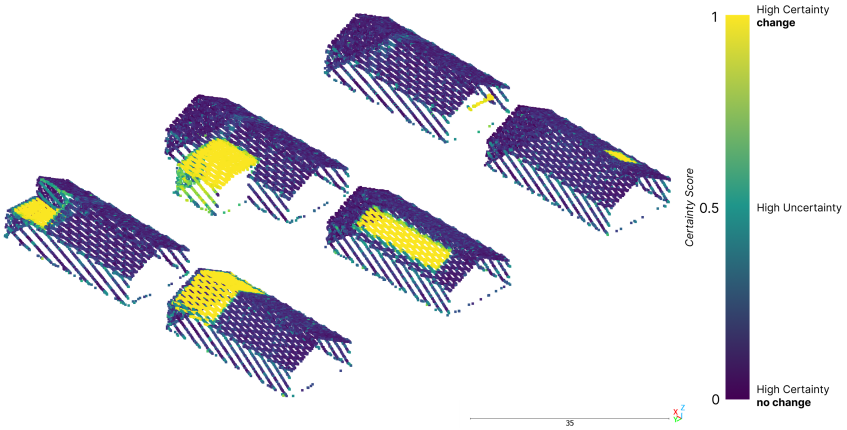
- Figure 6.16 presents the certainty score for the same region shown in Figure 6.14. Compared to Figure 6.14c, a subset of points, particularly along the roof edges, are now assigned a certainty score of 0.5, indicating uncertainty. This adjustment is particularly effective in identifying uncertainty on the roofs of terraced houses that are partially occluded by adjacent taller structures. In addition, some roof edge points are marked as uncertain due to self-occlusion, which is a correct assessment.
- Figure 6.17 shows three views of a second area: (a) the original reference point cloud dataset colored by height, (b) the initial probability scores from the RF model, and (c) the certainty scores after incorporating occlusion. In the final image, some boundaries of changed regions have been adjusted to 0.5 to reflect uncertainty. For instance, for the building on the left, where a change in roof height occurred, the walls were initially assigned a high probability of change, but are now marked as uncertain in the certainty score. While this adjustment is not entirely correct, the roof area still retains a high change score, which supports human interpretation to see a change has happened. Additionally, other dynamic occluded effects, such as the appearance or disappearance of dormers, continue to produce high change probabilities, which aligns with expectations.



(a) Reference point cloud dataset. Colored by point height.



(b) Initial probability scores from the RF model.



(c) Final scores after incorporating occlusion.

Figure 6.17.: Comparison of predictions before and after integrating occlusion information.

6.6. Method Performance on Real Datasets

The performance of the change detection algorithm was visually assessed using the national height model of the Netherlands number 4 (AHN4) as reference point cloud dataset and national height model of the Netherlands number 5 (AHN5) as target point cloud dataset. Comparison was done with ortophotos from Beeldmateriaal. Points in the target point cloud dataset are color-coded according to their certainty score: values near 0 or 1 indicate strong confidence in no-change or change, respectively, whereas values close to 0.5 reflect uncertainty. The findings are presented below, organized by object type and observed change pattern. The corresponding figures can be found in Appendix D. The below observations are visually supported by colored points of AHN5, what has been changed from AHN4.

Building Modifications and Structural Changes

- **Dormers:** Newly constructed dormers are clearly detected, with high certainty at the relevant points (see Figure D.1).
- **Extensions:** Lateral and vertical building extensions are consistently identified with high confidence (see Figure D.2).
- **New Buildings:** Newly build buildings are reliably classified as changed, even if similar in shape to previous structures. These are often more apparent in the point cloud data than in orthophotos (see Figure D.6).
- **Removed Structures:** Removed sections of buildings are detected, typically showing high change probabilities in the corresponding ground-level points (see Figure D.7).
- **Incomplete reference point cloud dataset:** In cases where buildings are poorly captured in the AHN4 dataset (which is happening for Figure 6.4, the algorithm tends to predict high change probabilities due to insufficient comparison data (see Figure D.8).
- **Infills Between Existing Structures:** For newly added buildings situated between pre-existing structures, roof points typically show high certainty of change, while wall points often remain uncertain due to low stability factor values in the reference point cloud dataset (see Figure D.9).

Roof Objects and Temporary Elements

- **Solar Panels:** Angled solar panels mounted on roofs generally receive scores around 0.75, contrasting with unchanged roof areas that score near 0. Smaller panels aligned with roof slopes are detected less reliably (see Figure D.3).
- **Temporary Objects:** In Amsterdam, temporary objects such as cars or furniture on rooftops are often detected as structural changes, given their presence in only one epoch (see Figure D.10).
- **Scaffolding:** The removal of scaffolding between epochs can result in adjacent ground points being marked with high certainty of change (see Figure D.11).

Vegetation and Ground-Related Observations

- **Trees in Building Raster Cells:** Tree points misclassified within building-labelled raster cells may result in falsely detected changes (see Figure D.4).

- **Opaque Vegetation and dynamic occluded:** If dense vegetation in AHN4 blocks all laser pulses and is later removed in AHN5, the occluded areas may remain classified as changed due to the persistent lack of visibility (see Figure D.4).
- **Ground Points:** Some individual ground points are falsely detected as changes. These are generally isolated and not visually meaningful (see Figure D.12).

Occlusion and Systematic Uncertainty Patterns

- **Occluded Walls:** Wall segments initially detected as changed often become more uncertain after applying occlusion corrections (see Figure D.5).
- **Distance to Flightline Effects:** Unchanged points located approximately 129–138 meters from the flightline have an increased certainty score, with scores ranging between 0.5 and 0.7. This suggests a potential systematic effect related to the scanning geometry (see Figure D.13).

The performance of the change detection method was also evaluated using the Rotterdam 2023 dataset as the reference point cloud dataset and AHN5 as target point cloud dataset. Additionally, a second setup used AHN4 as reference point cloud dataset and the Rotterdam 2024 dataset as target point cloud dataset. Visual comparison was supported by orthophotos from Beeldmateriaal and satellite imagery from Topotijdreis. Since many results are similar to those discussed above in this section, this part focuses on the main differences introduced by using the Rotterdam datasets.

First, the setup with Rotterdam 2023 as reference point cloud dataset and AHN5 as target point cloud dataset is discussed. It is worth noting that these datasets differ by only one year, as the AHN5 data for Rotterdam was acquired in 2024. Examples of the results are shown in Figure D.14 and Figure D.15.

- Unchanged areas are predicted with high confidence, especially where neighbouring points have similar values. This results in a clear visual distinction for actual changes, due to strong contrast in point cloud color differences. The dense point cloud of the Rotterdam dataset as reference point cloud dataset contributes to this clarity.
- Occlusions are minimal in the reference point cloud dataset, again due to the high density of the Rotterdam point cloud.
- One geotile had to be subdivided into smaller tiles for processing because of long runtime. In certain smaller tiles, the occlusion check incorrectly labelled all points as occluded, a result that did not occur in other tiles. This inconsistency may indicate a processing mistake. This issue manifested as a repeated misclassification across tiles containing the same building geometry.

Next, the setup with AHN4 as reference point cloud dataset and the Rotterdam 2024 dataset as target point cloud dataset is discussed. In this case, the temporal gap is four years. Relevant figures are shown in Figure D.17 and Figure D.18. Key differences are summarised below:

- Small architectural features such as church towers are not always well captured in AHN4, resulting in changed parts of that building in the colored Rotterdam 2024 point cloud dataset.
- Compared to the previous setup, prediction certainty is lower. Probabilities vary more across neighbouring points, as illustrated in Figure D.16.

The previous sections provided visual comparisons and qualitative validations of the change detection results. Although no ground truth is available for the real-world datasets, Table 6.4 presents a numerical summary based on the distribution of certainty scores. These scores are grouped into 0.1 intervals to give insight into the model's confidence levels across different dataset pairs.

Table 6.4.: Distribution of Certainty Scores in 0.1 intervals for each dataset pair.

Certainty Score Range	**AHN4–AHN5**	**Rotterdam2023–AHN5**	**AHN4–Rotterdam2024**
$0.0 \leq$ score < 0.1	28.40%	68.79%	50.47%
$0.1 \leq$ score < 0.2	14.99%	21.97%	23.28%
$0.2 \leq$ score < 0.3	29.18%	4.69%	10.00%
$0.3 \leq$ score < 0.4	9.79%	2.18%	5.45%
$0.4 \leq$ score < 0.5	5.44%	0.94%	3.21%
$0.5 \leq$ score < 0.6	3.34%	0.64%	6.99%
$0.6 \leq$ score < 0.7	0.85%	0.19%	0.15%
$0.7 \leq$ score < 0.8	0.78%	0.13%	0.08%
$0.8 \leq$ score < 0.9	0.57%	0.09%	0.07%
$0.9 \leq$ score ≤ 1.0	3.46%	0.39%	0.30%

Looking at the table, some clear patterns appear. First, for high certainty in no change scores (below 0.4), the datasets that include Rotterdam data, whether as the reference point cloud dataset or target point cloud dataset, have the highest percentages in the most certain category ($0.0 \leq$ score < 0.1). After that, the percentages steadily decrease as certainty goes down (meaning the score increases towards 0.5). In contrast, the AHN4–AHN5 pair peaks in the $0.2 \leq$ score < 0.3 range, and it also shows relatively high percentages across other uncertain ranges.

For certainty scores of change (above 0.6), all three dataset pairs reach their highest peak in the most certain range ($0.9 \leq$ score ≤ 1.0). Interestingly, for all pairs, the second highest percentages appear in the $0.6 \leq$ score < 0.7 range, but these are still very small compared to the peak at the highest certainty category.

It is important to note that structures present exclusively in the Rotterdam dataset but absent in the corresponding national height model of the Netherlands (AHN) dataset cannot be detected. Since the Rotterdam point clouds are unclassified, the algorithm can only compare objects classified as buildings in the AHN datasets.

7. Discussion & Limitations

This chapter presents the meaning of the results and outlines their limitations. It focuses on four main aspects of the research and ends with a theoretical comparison to existing change detection methods for point clouds. The four aspects are: the interpretation of the synthetic scene generated, the design choices made for the random forest (RF) model, the detection and integration of occlusion into the method, and the evaluation of the performance of the method on real-world datasets. It will end with a theoretical comparison between the developed method and existing methods.

7.1. Synthetic Dataset

In general, the synthetic scenes exhibit reasonable alignment with the real national height model of the Netherlands (AHN) datasets. It should be noted that the national height model of the Netherlands number 4 (AHN4) synthetic scene was generated using a different scanner from the one used to acquire the real AHN4 data. As a result, some variation in point capture is expected.

A significant limitation in the scene is the simplified composition. Only buildings were included in the synthetic model, excluding other common urban elements such as trees, vehicles, and street furniture. Moreover, due to time constraints, no post-processing was applied. Several enhancements are proposed for future iterations of the synthetic dataset:

- Introduce minor horizontal perturbations to reduce linear patterns in point distributions, especially for surfaces with uniform geometry.
- Add a small number of intentionally misclassified points to emulate classification errors.
- Incorporate Gaussian noise to simulate realistic measurement uncertainties.

To show the performance of Helioss++, a noteworthy example is Scene 1 in Table 5.3, where both epochs employed identical scanner settings and flightlines. Despite this, minor differences in accuracy and point density are observed, as indicated in the same table. This suggests that `Helios++` inherently introduces a certain degree of noise during simulation.

Several issues arose during the simulation process. For example, certain parts of buildings, particularly surfaces with complex geometry, such as faces with five or more vertices, were not correctly captured by the scanner. This issue is depicted in Figure 7.1, where a section of a roof was missed and points were generated incorrectly on the ground level.

Another issue relates to the simulated intensity values, which do not align with realistic expectations. While the simulation incorporates effects related to scan angle and distance from the flight path, it omits material properties and surface reflectance. Some model surfaces were assigned different material types (e.g., glass, metal, or rough surfaces), but only

the `metallic percentage` parameter in Blender influenced the resulting intensity values. These factors (material properties and surface reflectance) are crucial for realistic intensity simulation.

In real datasets such as AHN4 and national height model of the Netherlands number 5 (AHN5), intensity values vary with the distance from the flight path. This behavior was not replicated in the synthetic dataset. However, in the synthetic scene the intensity differed when the surface was angled instead of flat, which is correct.

Based on these observations, it is concluded that the simulated intensity values do not faithfully represent real-world conditions. Therefore, intensity data from the synthetic scene was excluded from both the dataset and the developed model.

The labelling results are encouraging. As illustrated in Figure 6.5, the majority of points in the synthetic dataset were correctly classified. Furthermore, the dataset includes useful class labels that distinguish various types of changes. These labels can be employed to filter the data, evaluate the performance of the change detection method, and analyze which types of changes are accurately identified and which are not.

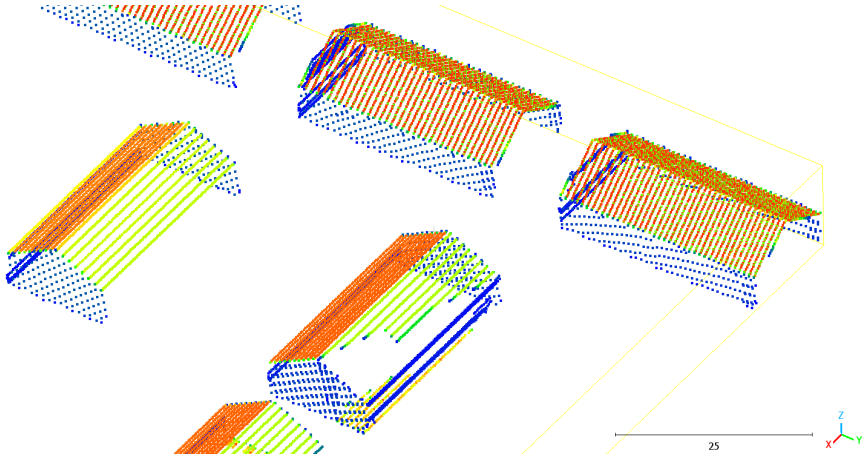


Figure 7.1.: Generated points illustrating a missing roof segment, with points incorrectly projected onto the ground.

7.2. Design Decisions in the Random Forest Model

The final method was not trained to detect small height changes of 4 and 8 cm. Including these small differences introduced uncertainty, even in areas without actual change. This is likely because the scanner accuracy is close to these values, making it inherently difficult to distinguish such small geometric differences using only shape-based features. Additional data attributes, such as intensity or color, could improve the detection of subtle changes like the addition of thin solar panels. For this reason, the smallest height difference included during training was 12 cm. In hindsight, it would have been better to include smaller changes consistently, so that the minimum detectable change would be clearly defined.

During the development of the random forest (RF) model, it became evident that standard performance metrics such as precision and recall do not fully represent the practical quality of change detection in point clouds. This is due to two main reasons:

- In certain areas, insufficient data from one of the epochs prevents the method from making a confident prediction. Precision and recall evaluate only binary outcomes ('yes' or 'no') and do not account for such uncertain regions.
- Visual interpretation of change relies heavily on contrast between changed and unchanged areas. For example, if 20 points have truly changed but only 12 are classified as such, and the remaining points receive a certainty score around 0.4, the area will still be visually perceived as changed, even though classification accuracy is only 60%.

When labelled data is available, visual evaluation is often more informative than relying solely on precision, recall, or accuracy metrics.

The impact and behavior of specific features are discussed below:

- **Stability factor difference:** This feature often reaches a value of 1 in changed areas and -1 near building edges, where fewer points are captured in the reference point cloud dataset. The sensitivity of this feature depends strongly on the radius used during its calculation. Smaller additions may be missed due to the default 0.5 m radius, although added dormers are still detectable. The method assumes no points beneath buildings, so in cases where windows reveal interior walls or floors, the feature becomes unreliable. For newly constructed buildings, walls often receive incorrect stability factor values, so the feature mainly reflects roof changes. Boundaries between changed and unchanged regions are very distinct in this feature, and neighboring points tend to show consistent values.
- **Height difference:** This feature is more effective than the stability factor difference for detecting added structures, as it identifies both wall and roof changes. However, it offers less contrast at the transition boundaries. The feature can be either positive or negative, making it informative about the nature of the change. However, it is more variable among neighboring points and more sensitive to occlusions.
- **Distance to flightline:** As shown in Figure D.13, a clear jump in certainty score occurs between 129 m and 138 m from the flightline. This feature is not used to detect change directly but instead helps method uncertainty. Laser beams widen with distance, and coverage differs depending on overlap from multiple flightlines. This feature captures those acquisition effects. Although it would have been interesting to evaluate the performance without this feature, this was not tested. According to Figure 6.11, overall performance would decrease if the feature were removed.
- **Density differences (2D and 3D):** These features had low importance. Visualizations show strong variation in overlap zones between flightlines and in areas directly underneath the scanner path. These features are also highly sensitive to occlusion and vary across datasets. They were retained in the model as they provide complementary information to the stability factor.
- **Linearity, planarity, and sphericity differences:** These features also had low importance but were included to capture scan inconsistencies. Differences can be observed even on the same roof surface when captured from different scan lines, while wall

surfaces tend to show more consistent values. These features mainly highlight misalignment between scan lines and are illustrated in Figure 5.18, Figure 5.19, and Figure 5.20. The sphericity feature might have been more useful had trees been included in the synthetic scene, as it could help model their complex geometry.

- **stability factor of the target point:** This feature gives insight into the geometry of a point within its own epoch. It helps distinguish roofs from walls and supports the interpretation of the stability factor difference.

Several dataset-level features were initially considered, with the idea of helping the method to determine thresholds (e.g., minimum height difference for change) based on dataset properties. However, since these values were constant across all points in a dataset, they showed little variance and were not useful for the model. Instead, training on a wide variety of dataset types was considered a better approach for improving generalization across different scenes.

7.3. Occlusion Detection

Before interpreting the results, it is important to recall the two approaches used for occlusion detection: one based on aircraft positions from the reference point cloud dataset (Boolean 1), and the other using positions from the target point cloud dataset (Boolean 2). In both approaches, occlusion is assessed relative to the reference point cloud dataset. For both occlusion models, similar observations can be made:

- In general, areas that are expected to be occluded, especially roof surfaces, are detected correctly. This is evident in Figure 6.6 and Figure 6.7. However, some exceptions occur. In Figure 6.9b, certain roof areas are incorrectly classified as occluded. A likely explanation is misalignment of these buildings with respect to surrounding structures. Increased roof tilt can also reduce occlusion detection accuracy, for the same reason that walls generally perform worse: visibility is harder to assess at steeper angles.

- For wall surfaces, errors occur when multiple point layers are present or when the scanner's view angle is limited. In such cases, visible points are sometimes incorrectly marked as occluded. This is visible in Figure 6.10. The smaller the angle between the scanner direction and the vertical axis, the more difficult it becomes to determine visibility accurately.

- On the synthetic dataset, the occlusion detection performs less reliably than on real data, as shown in Figure 6.8b. Several wall surfaces that should be visible are mistakenly classified as occluded. One likely reason is the way aircraft positions were defined in the simulation. Additionally, the simulated viewing angle is steep, almost vertical, for these parts of the scene, which further complicates visibility assessment.

- Vegetation introduces additional challenges. When a tree is located between the aircraft position and the surface of the target point, points underneath are often still marked as occluded. This occurs even though the method is designed to consider differences in the number of returns as a visibility cue. In these cases, points should have been retained but are still discarded.

In general, the occlusion detection method performs reasonably well and contributes meaningfully to the overall change detection pipeline. The ability to distinguish between wall and roof surfaces, based on the stability factor, adds another layer of logic. Still, several limitations remain:

- The 20% threshold used in one of the conditions in Figure 5.28 in Section 5.7 is currently based on visual inspection. This threshold might not generalize well to other datasets and could benefit from optimization, even within this dataset.
- The use of the stability factor to differentiate walls from roofs is not always reliable. In greenhouses, for instance, points below glass roofs lower the stability factor, which can cause the roof to be misclassified as a wall. Similarly, when trees overhang rooftops, the reduced stability factor leads to incorrect classification. Unconventional building designs, such as inward-sloping walls, can also distort stability factor values and affect classification accuracy.

7.4. Interpretation of the Method's Performance on Real Datasets (AHN4, AHN5, Rotterdam)

This section reflects on the visual interpretation of the results produced by the trained random forest (RF) model and the integration of occlusion logic in the final certainty score. The real-world datasets considered are national height model of the Netherlands number 4 (AHN4), national height model of the Netherlands number 5 (AHN5), and the Rotterdam 2023 and 2024 point clouds.

Evaluation Setup and Limitations

Only visual assessment was performed on the change detection results; no quantitative ground-truth comparison was used. As a result, definitive performance claims (e.g., "all dormers are detected") cannot be made, only cautious observations like "most dormers are detected." Additionally, only selected areas were analyzed, and thus, the findings do not generalize to the full datasets.

The method is computationally intensive, especially during occlusion analysis and flightline proximity calculations. This significantly limited the number and size of areas that could be processed within a reasonable timeframe. Moreover, the performance characteristics indicate that this method is not suitable for real-time or on-the-fly applications.

Observed Change Detection Behavior

Building additions, such as dormers, roof expansions, and new constructions, are consistently detected with high certainty. Even minor features, like tilted solar panels, are often recognized, while flat or low-profile objects are more likely to be missed.

Building removals or reductions, including partial demolitions, are generally visible in the point cloud data. However, interpreting the type of change requires additional context, such as the reference point cloud dataset, since the point cloud alone is insufficient.

The smallest trained change was 12 cm, and these small alterations, such as subtle roof modifications, are typically detected. The model also picked up temporary objects like traffic cones, confirming high spatial sensitivity, as illustrated in Figure 6.12c.

Contextual Clarifications and Data Fusion

Temporary objects (e.g., parked cars, rooftop furniture) are sometimes misclassified as permanent changes. Orthophotos can clarify such ambiguities, suggesting an efficient workflow: use the point cloud to flag changes, and verify with orthophotos.

Point clouds can reveal changes that are obscured in orthophotos by shadows or vegetation, particularly for small or complex changes. This emphasizes the complementary strengths of 3D and 2D data. This approach is likely faster than manually inspecting orthophotos and is particularly helpful for identifying small or geometrically complex changes. In addition, the 3D geometry of point clouds often makes it possible to look around the object.

Challenges from Data and Model Limitations

Some structures in AHN4, like church towers or houses, are poorly captured, leading to false change detection. These cases expose a model weakness where certainty is based on unreliable reference data. A building-level pre-check could flag such structures and reduce certainty scores accordingly.

Integrating occlusion into the model significantly improves reliability. Occlusion is detected based on mismatches in `return number` and `number of returns` between the aircraft and the target point. This approach generally reduces false positives in shadowed or obscured regions. However, a systematic misclassification occurs when a building or tree is removed from the reference point cloud dataset to the target point cloud dataset. If this removed object had previously caused occlusion for a nearby building, the model incorrectly marks the newly visible region as dynamic occluded. This issue is visualized in Figure D.4 and reflects a limitation of the occlusion integration.

A recurring pattern was observed: unchanged points located 129–138 meters from the flightline received disproportionately high certainty scores. This indicates an issue with the distance-to-flightline feature, which needs more critical integration with occlusion handling.

Impact of Dataset Characteristics

Datasets with higher point densities as the reference point cloud dataset, like Rotterdam 2023, tend to produce higher stable certainty scores and clearer change detection results. As shown in Table 6.4, including a dataset with higher point density makes the certainty scores more confident in unchanged areas. This effect is strongest when the denser point cloud is used as the reference point cloud dataset.

Glass roof surfaces sometimes allow internal points to be captured, lowering the stability factor and degrading results.

Due to differences in classification schemes between AHN5 and the Rotterdam datasets, all points within the raster cells labelled as buildings were treated as such. This led to some misclassifications (e.g., cranes or overhanging vegetation).

7.5. Theoretical and Qualitative Comparison to Existing Methods

This section discusses how the developed method compares to several existing approaches for point cloud change detection, as introduced in Section 2.3. While no direct quantitative benchmarking could be performed, due to time constraints, the following qualitative comparison highlights the key differences in methodology, assumptions, and observed behavior.

Raster-based Differencing. A commonly used approach involves converting point clouds into raster grids (e.g., digital surface model (DSM)s) and subtracting them to identify changes. While computationally efficient, this process reduces the 3D structure into a 2.5D representation, where each cell stores a single height value. As a result, fine-grained geometric details are lost, and object-level interpretation becomes more difficult.

Point-to-Point Comparison. Traditional methods such as cloud-to-cloud (C2C) or cloud-to-mesh (C2M) comparisons rely on nearest-neighbour distances and require threshold tuning to distinguish real changes from noise. These approaches are sensitive to occlusion and variations in point density. In contrast, the method developed in this research includes a stability factor feature and integrates occlusion reasoning, reducing false positives in occluded areas. For example, our method avoids labelling walls as changed if they were occluded in one epoch.

Mesh-based Approaches. Some methods convert point clouds to surface meshes before comparing epochs. While this adds continuity, it also introduces interpolation artifacts and smoothing parameters that can obscure smaller changes. By working directly with raw point clouds, our approach avoids these simplifications and retains the ability to detect localized and high-frequency changes.

Deep Learning Methods. Deep learning approaches have shown strong performance, especially in structured environments. However, they require large, diverse training datasets that are typically unavailable for airborne laser scanning (ALS) building change detection in the Netherlands. Moreover, their predictions often lack interpretability. Our approach, by using an random forest (RF) classifier trained on synthetic data, is more transparent and better suited to datasets with limited labelled examples. Certainty values at the point level provide interpretable outputs and support visual verification.

Advantages of the Proposed Method

In summary, the developed method of this researc offers several qualitative advantages over existing approaches:

- **Occlusion Handling:** By explicitly modelling occlusion, the method reduces false positives, especially on building façades.

- **Interpretability:** Certainty scores are assigned at the point level, helping users understand where predictions are reliable. Visualization is consistent, making small and large changes equally visible.
- **Threshold-Free Learning:** The model learns decision boundaries from the data, avoiding manual tuning of height or distance thresholds, which are often dataset-dependent.

8. Conclusions

This chapter provides answers to the research sub-questions. At the end, a brief summary of the main research question is presented.

How can realistic urban ALS point cloud datasets be simulated to support the training and evaluation of change detection algorithms, including variations in data quality and occlusion?

Simulating realistic urban ALS point cloud datasets is essential for developing and evaluating reliable change detection algorithms. To achieve this, a synthetic urban scene was created with a variety of building modifications, such as adding or removing balconies, dormers, houses, and solar panels. This controlled setup enables automatic labelling of changes, which is often difficult in real-world datasets.

Using `Blender` allowed the incorporation of various types of buildings and architectural styles, enhancing the representativeness of the synthetic environment. Moreover, incorporating realistic challenges such as occlusions caused by tall buildings and varying distances from the flight path helped simulate key factors that affect data quality and change detectability in airborne laser scanning.

The `Helios++` simulator further contributed by realistically modelling the scanning process, including noise and various parameters of the scanner and flight path. This flexibility enables the generation of datasets with diverse scanning characteristics, allowing the training and evaluation of algorithms under different conditions and improving their generalizability.

Although the synthetic datasets retain certain idealisations, such as linear point distribution and the absence of natural elements like vegetation, they offer a valuable foundation for controlled experimentation. The comparable performance of the developed method on both synthetic and real-world scenes suggests that training with synthetic data is effective in this context. Moreover, the framework is designed for flexibility, allowing for straightforward extension with additional building types, scanning configurations, or environmental features.

In summary, the developed simulation approach balances control and realism, providing richly labelled and varied datasets that explicitly account for data quality variations and occlusion effects. This approach supports the development of change detection methods that can better generalise to real-world ALS data, while leaving room for future enhancements towards even more realistic scenarios.

How can a change detection method be developed that accurately identifies building changes while accounting for uncertainties in point cloud data?

A set of point-level features was developed for the model, divided into three groups: (1) basic features extracted from each point cloud, (2) features comparing the two epochs, and (3) dataset-related features. The third group was tested but later removed since it did not improve performance. The use of multiple synthetic datasets during training, however, helped the model learn from data with varying quality and uncertainty.

The final feature set captures both local information and differences between epochs. The most informative features were height difference and the difference in the stability factor (see Figure 2.11). The model learns decision boundaries from labelled examples, avoiding manual threshold selection.

Small changes below 12 cm were not included in training because they fall within typical scanner accuracy limits, making it difficult to design reliable geometric features for them.

Uncertainty was addressed in two ways: by training on data with different scanning properties and by including a feature measuring the distance to the flightline, since points further away generally have lower quality.

The output probability score provides richer information than just height differences. Combined with occlusion information, this score forms the certainty index, which will be discussed in the answers of the next subquestion.

How can occlusion be incorporated into the change detection process to improve prediction certainty?

Occlusion was handled by determining whether each point was visible from the scanner in each epoch. Points were assigned one of three occlusion categories:

- **Visible:** The point was visible.
- **static occluded:** The point was blocked in both epochs by an unchanged object.
- **dynamic occluded:** The point was blocked due to a change, for example when an object was removed or lowered.

This occlusion classification was combined with the RF model's output probabilities to adjust change likelihoods, resulting in a certainty score.

The certainty score improves visualization compared to just showing height differences. Since the certainty values range between 0 and 1, all detected change, small or large, are clearly visible through color mapping. Height difference alone can be dominated by outliers, making subtle changes hard to see.

Moreover, the certainty score reflects both the point properties and occlusion status, incorporating uncertainty directly. This is important because some areas simply cannot confirm change from point cloud data alone. Using height difference without this uncertainty measure leaves ambiguous areas unclear.

Beyond change detection, the occlusion detection framework developed here offers potential applications in several other areas, such as:

1. **Navigation in point clouds:** By highlighting occluded areas, such as walls or barriers, this framework can improve path planning by marking zones that are unknown or blocked.

2. **Building mesh generation:** For projects like 3D building models of the Netherlands (3DBAG), which build detailed models of buildings from multiple epochs, the framework can identify facades that remain occluded. This information can guide flight planning to ensure those areas are fully captured in future scans.

3. **Validation of point cloud epoch alignment:** As demonstrated in Figure 6.9b, this method can highlight misalignments between epochs. Here, misalignment means the epochs differ beyond a defined threshold, measured by the `ErrorHeight` metric, which corresponds to the radius defined in Equation 5.4.

How well does the developed change detection algorithm perform when applied to real-world datasets such as national height model of the Netherlands (AHN) and Rotterdam?

The model performs effectively on real-world datasets, accurately detecting building extensions, dormers, and new constructions in national height model of the Netherlands number 4 (AHN4), national height model of the Netherlands number 5 (AHN5), and Rotterdam point clouds. Subtle changes, such as small roof elements, are identified when they exceed the 12 cm detection threshold established during training. The method also captures removed structures, though it cannot independently differentiate between temporary and permanent changes.

False positives primarily arise from sparse data coverage, glass surfaces, or complex occlusion caused by vegetation. The assigned certainty scores assist in interpreting these uncertain cases and help guide manual visual verification. Integrating orthophotos into the workflow is recommended to improve semantic understanding of detected changes.

This research also demonstrates the value of synthetic point cloud datasets for training change detection algorithms applied to real data. Key differences between synthetic and real datasets include the absence of elements like trees (which could be added in future work) and variations in point feature intensity.

Another important observation is the improved performance when using a denser point cloud dataset, such as the Rotterdam dataset. The results indicate that optimal outcomes are achieved when the denser point cloud is used as the reference point cloud dataset, enabling the target point cloud dataset to produce more accurate predictions.

Main Research Question: To what extent can a building change detection method between two epochs of aligned point cloud datasets be developed to maximize reliability using a certainty index, incorporating dataset characteristics, applied to the AHN and Rotterdam datasets?

Using point-level certainty scores and by explicitly incorporating occlusion reasoning, the developed method effectively addresses common challenges in ALS-based building change detection. Although some limitations remain, particularly in handling vegetation and temporary objects, the results demonstrate that combining 3D feature engineering, supervised learning, and occlusion-aware logic enables reliable and interpretable change detection.

This research is innovative and novel for the combination of the following:

- **Automatic Bi-Temporal Change Labelling:** The approach generates synthetic bi-temporal point cloud datasets with automatically labelled building changes, including detailed change types.
- **Integration of Occlusion:** Occlusion is incorporated both in the synthetic data simulation and directly into the change detection method by transforming the random forest model's probability output into a certainty score that accounts for occlusion.
- **Certainty Score Visualization:** Instead of only showing the magnitude of change, the method provides a certainty score reflecting confidence in the change detection. The visualization uses consistent coloring for all detected changes, ensuring that small and large changes are equally visible.
- **Use of Raw Point Clouds:** The method operates directly on raw point clouds at the individual point level, preserving the full 3D data without conversion or simplification.
- **Threshold-Free Decision Making:** Rather than relying on manually set thresholds, the model learns decision boundaries automatically from the training data.
- **Trained on Diverse Scanning Conditions:** The algorithm was trained on synthetic datasets generated with varying scanning parameters. This diversity enables the method to generalize well and perform reliably across different ALS datasets.

In conclusion, this research contributes to the larger goal of maintaining reliable and up-to-date urban data. Detecting structural changes in buildings is essential for applications such as urban planning, cadastral maintenance, and environmental analysis. The developed method introduces a new framework that combines synthetic, labelled training data with occlusion-aware reasoning and a certainty-based output. It improves automated change detection in 3D point clouds and demonstrates good generalizability to real-world datasets. The resulting pipeline is reproducible and can serve as a foundation for future research and operational workflows involving digital twins and urban monitoring.

This work also fits into the wider context of the Master Geomatics programme and the societal challenges it addresses. It covers the full geomatics pipeline: from generating and simulating spatial data to extracting valuable information. The project involved working with complex geometries, visualizing 3D data, and applying the results to the built environment. Accurate and timely 3D information supports spatial planning. The method was developed through an iterative process, balancing concept design, algorithm development, and real-world validation. By integrating both simulated and real datasets, the approach remained adaptable and realistic within the time constraints of the thesis.

Data Availability: The trained RF algorithm, selected results from the AHN4–AHN5 datasets, and the synthetic datasets generated in this study are publicly available at the following repository: here or https://drive.google.com/drive/folders/1z6u_rhH-baqwG_XcdseNyPGTykUJ6uRe?usp=sharing.

9. Future Research

This chapter outlines potential directions for future research, based on the limitations encountered and findings derived during this study. The first section discusses ideas that could have been implemented within the scope of this research, given more time. The second section presents broader suggestions for future studies that build upon or complement this work.

9.1. Within the Scope of This Study

Several extensions to the current approach were identified during the research process. These are listed below and could serve as valuable directions for further refinement.

- **Occlusion Analysis in the Rotterdam Dataset:** Occlusion detection did not yield the expected results for several areas in the Rotterdam dataset. A more detailed investigation is needed to determine whether this is due to incorrect aircraft position predictions or inappropriate code settings. Insights from this analysis could improve the occlusion model.
- **Expanding Scene Variability:** The synthetic scene could be made more representative by introducing a wider range of urban elements. For instance, trees located near buildings should not be misclassified as structural changes. Features like sphericity may help distinguish such cases. Additional elements, such as crane parts or semi-transparent features like roof windows, could be included to simulate more complex real-world conditions.
- **Pre-Evaluation at Building Level:** Some buildings are poorly captured in national height model of the Netherlands number 4 (AHN4), resulting in incomplete representations. A building-level pre-check could address this by adjusting the certainty scores of buildings with sparse point coverage (e.g., less than 90% of raster cells populated). This would extend the occlusion correction mechanism to a broader building-level assessment. An example is shown in Figure 5.30.
- **More Variation in Flightline Configuration:** Deviations observed around 129–138 meters from the flightline suggest limitations in the current random forest (RF) model. To mitigate overfitting to specific distance intervals, synthetic scenes should incorporate greater variation in flightline configurations.
- **Increased Number and Diversity of Synthetic Scenes:** Currently, only five synthetic scenes were generated, each with controlled acquisition parameter variations. Expanding the dataset to include more scenes, each combining multiple parameter variations, would provide deeper insight into the effects of point density, scan angle, and occlusion on classification confidence. It would also enable more context-sensitive adjustments to certainty thresholds.

- **Further Evaluation of Feature Importance:** After implementing the above extensions, features such as planarity, sphericity, and linearity should be re-evaluated to assess their actual contribution to classification accuracy. Some features may prove more useful under increased scene complexity and with the addition of pre-filtering steps.

9.2. Future Directions for Other Researchers

Beyond the current scope, several directions are suggested for researchers interested in advancing change detection in airborne point clouds.

Graph Neural Networks for Context-Aware Classification

A key limitation of the current method is the point-wise classification approach, where nearby points may receive different predictions even though they belong to the same object. This can lead to fragmented or unstable results. An alternative would be to incorporate spatial context during classification. Graph Neural Networks are designed to capture local and global relationships between points by representing the point cloud as a graph. Each point becomes a node, and edges represent spatial proximity or similarity. Prior research by de Gélis et al. [2021b] and Kharroubi et al. [2022] shows that Graph Neural Networks can improve consistency in predictions and enhance structural understanding, especially in complex urban environments.

Solar Panel Detection via Intensity Change

Visual inspection indicated noticeable intensity changes associated with solar panel installations. This suggests potential for using intensity as a proxy for detecting added or removed solar panels. A dedicated study could evaluate how consistent and reliable this intensity shift is under various acquisition conditions and whether it can be used for specialized change detection.

Use of Intensity and Color Information

Both intensity and color contain valuable semantic information. While these were not available in the synthetic data used here, future work could focus on generating or annotating real datasets where these attributes are present. Combining aerial imagery with point clouds could facilitate semi-automatic labelling. A hybrid approach, starting with image-based change detection followed by manual refinement, could result in a high-quality dataset for training intensity-aware or color-sensitive classifiers.

Extent Certainty Index

The certainty index introduced in this research offers a promising way to express confidence in change detection outcomes. Several extensions could further enhance its reliability and interpretability.

One possible improvement is to incorporate the distance to the flightline directly into the certainty index, rather than using it as a standalone feature. This requires a more detailed understanding of how distance influences point density, occlusion, and scanning quality. Controlled experiments that vary only the flightline geometry could help isolate these effects and support a more data-driven integration.

Additionally, if more detailed knowledge becomes available about how specific scanner types and flight parameters influence the classification results, the certainty index could be tailored to those acquisition settings. This would make the index more adaptable and trustworthy across datasets with varying characteristics.

Introduce a spatial component by examining the labels of neighbouring points. If a point is surrounded by points with the same label, the certainty can be increased. If neighbouring labels are inconsistent, the certainty should be decreased accordingly.

Runtime Considerations

The current implementation of the method is computationally expensive and time-consuming. For future research, it would be valuable to explore a multi-resolution or hierarchical approach to change detection. As discussed in Section 2.6, an initial coarse-level detection could be applied over large areas to identify potential changes, followed by a more detailed analysis when zooming in on specific regions of interest. This could significantly reduce processing time while preserving accuracy where it matters most.

A. Reproducibility Self-Assessment

A.1. Marks for Each of the Criteria

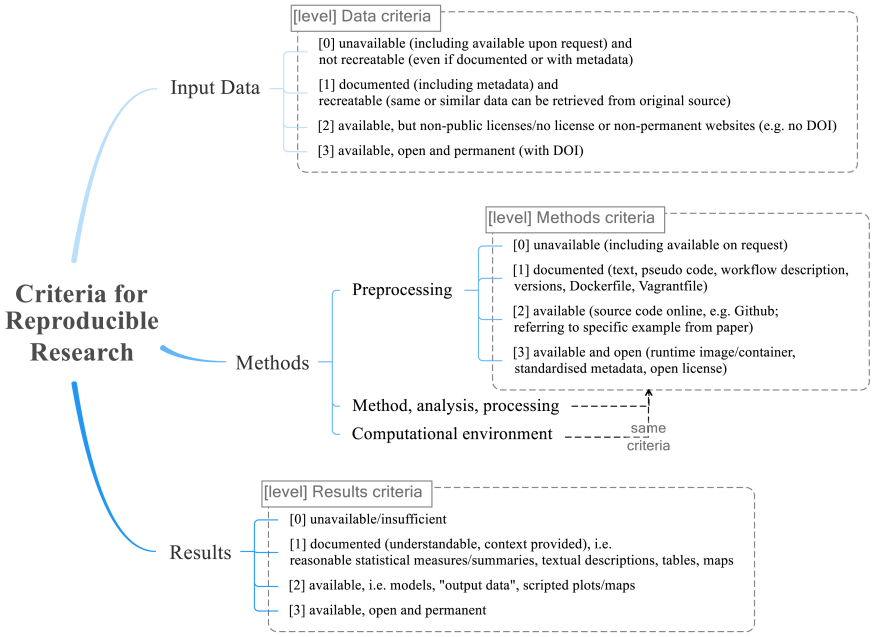


Figure A.1.: Reproducibility criteria to be assessed.

The reproducibility of this research is self-assessed based on five criteria, with a score of 0 to 3 for each:

1. **Input Data:** 1
2. **Preprocessing:** 3
3. **Methods:** 3
4. **Computational Environment:** 3
5. **Results:** 2

A.2. Self-Reflection

Input Data. The workflow of this research uses the national height model of the Netherlands number 4 (AHN4) and national height model of the Netherlands number 5 (AHN5) datasets,

which are open and publicly available. For validation and testing, the Rotterdam point cloud datasets were used. These are not open datasets and were only accessible through Geodelta. This limits full reproducibility for this part of the research.

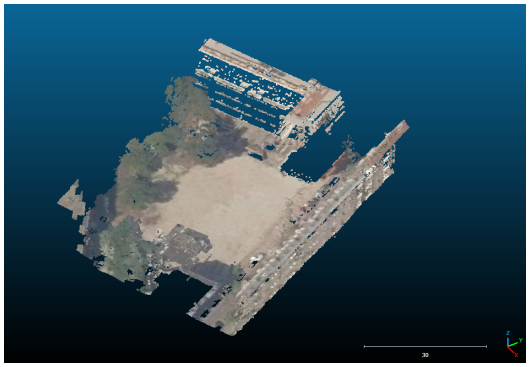
Preprocessing. The steps taken for preprocessing the data are explained in detail, including how the features were generated and labelled. The synthetic data generation pipeline is described and supported by custom figures to improve understanding.

Methods. The methodology and implementation are explained step-by-step in Chapter 4 and Chapter 5. All key software used, `Helios++`, `Blender`, `Python` (with open packages), `Potree`, `CloudCompare`, and `QGIS`, are open-source. One closed-source C# library provided by Geodelta was used to read `.laz` files, but the same functionality could be replicated in Python using the open `pylas` package.

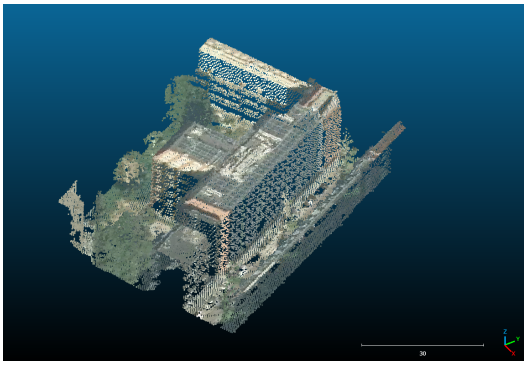
Computational Environment. The tools used are well-documented and freely available, making it possible to recreate the computational environment with limited setup. No proprietary software was used for processing or analysis, aside from the optional C# module mentioned above.

Results. The thesis document, including all figures, results, and discussion, will be made publicly available. The trained Random Forest model and some processed national height model of the Netherlands (AHN) tiles are shared here. However, results based on the Rotterdam dataset cannot be published due to restrictions set by the data provider (Geodelta).

B. Figures Nearest Neighbour Distances

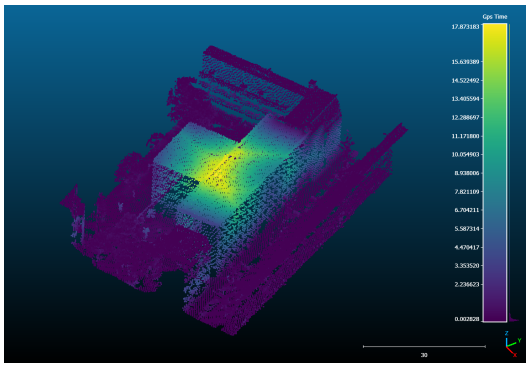


(a) Part of Area 1 in AHN4.

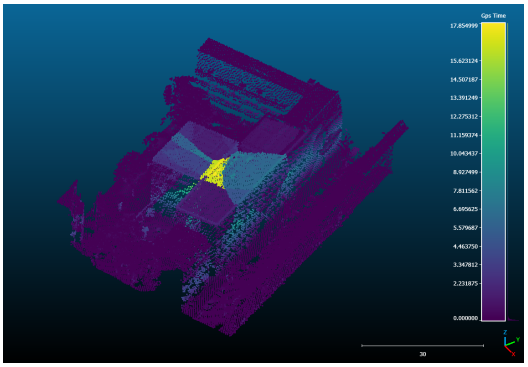


(b) Part of Area 1 in AHN5.

Figure B.1.: Visual comparison between AHN4 and AHN5 for evaluating the four nearest neighbour methods.

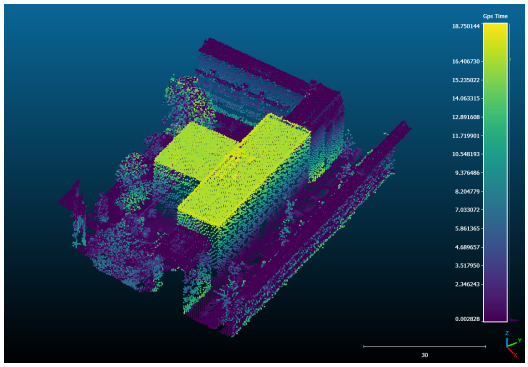


(a) AHN5 points colored by their 3D Euclidean distance to the nearest AHN4 point (3D search).

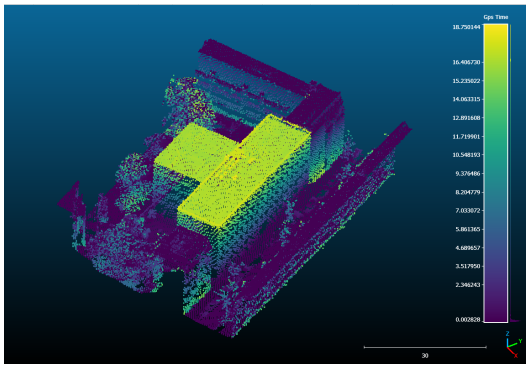


(b) AHN5 points colored by their vertical (Z) difference to the nearest AHN4 point (3D search).

Figure B.2.: 3D nearest neighbour search: comparison of Euclidean vs. height difference.



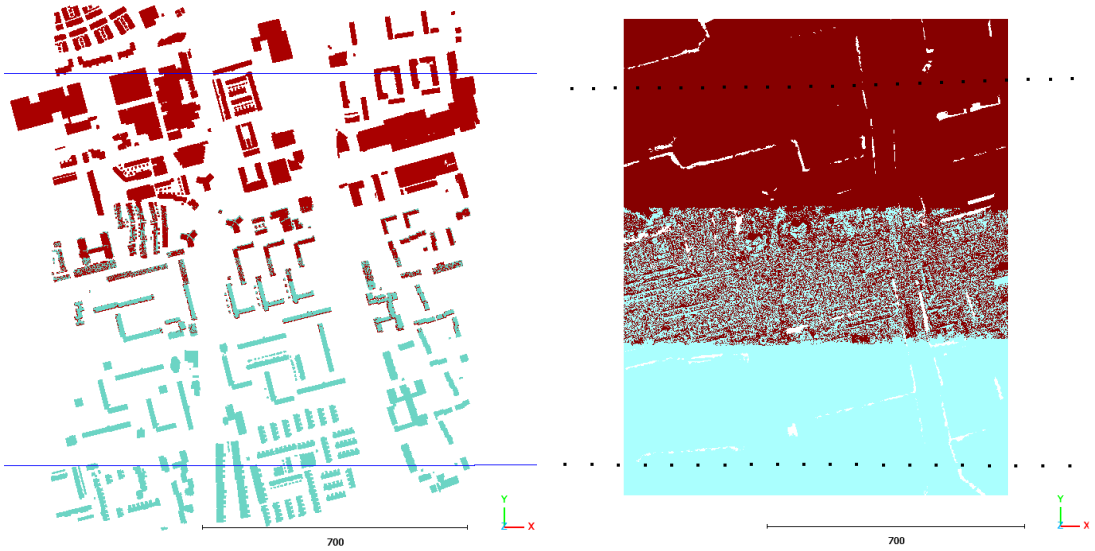
(a) AHN5 points colored by their 3D Euclidean distance to the nearest AHN4 point (2D search).



(b) AHN5 points colored by their vertical (Z) difference to the nearest AHN4 point (2D search).

Figure B.3.: 2D nearest neighbour search: comparison of Euclidean vs. height difference.

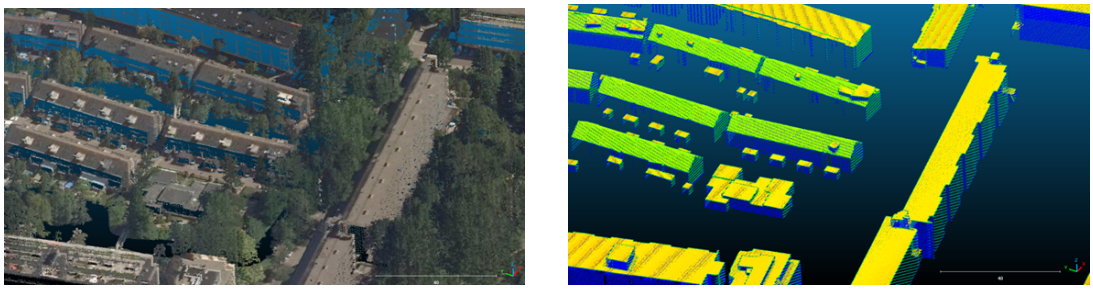
C. Figures Evaluation of the Model Input: Synthetic Datasets



(a) Simulated scene (top view), showing both flightstrips (red and light blue). Dark blue represents the flightlines.

(b) AHN4 dataset (top view), colored by `source id` indicating flightline. Black lines are aircraft positions.

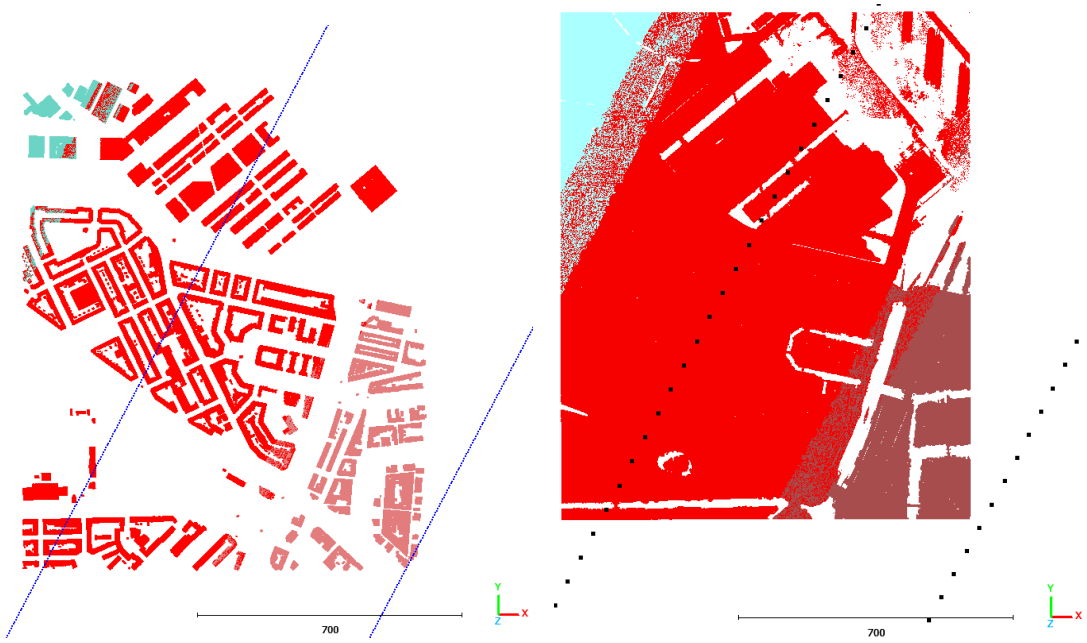
Figure C.1.: Comparison of simulated and real flight strips in the AHN4 dataset for Area 3 (see Figure 4.1). Keep in mind, the bounding boxes are different.



(a) Building roofs and facades in the AHN4 dataset.

(b) Building roofs and facades in the simulated dataset.

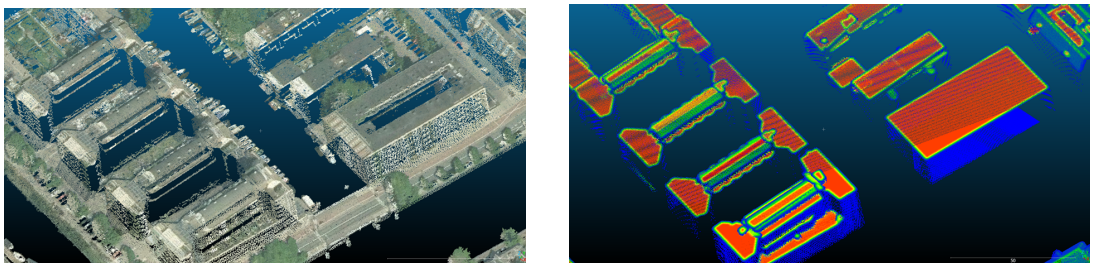
Figure C.2.: Comparison between AHN4 and the simulated scene for Area 3.



(a) Simulated scene (top view), showing all three flightstrips (light blue, red and pink). Dark blue represents the flightlines.

(b) AHN5 dataset (top view), colored by `source id` indicating the originating flightline. Black lines are aircraft positions.

Figure C.3.: Flight strip visualization in the simulated AHN5 dataset for Area 1. Keep in mind, the bounding boxes are different.



(a) Building facades in the AHN5 dataset.

(b) Building facades in the simulated dataset.

Figure C.4.: Comparison between AHN5 and the simulated scene for Area 1.

D. Figures AHN4 - AHN5 Results

D.1. AHN

Figure D.1.: Change detection results for a building where a dormer was added between the national height model of the Netherlands number 4 (AHN4) (reference point cloud dataset) and national height model of the Netherlands number 5 (AHN5) (target point cloud dataset) epochs. The added structure is identified with high certainty in change. Orthophotos from 2021 and 2024 are provided for context, source: Beeldmateriaal Nederland.

Figure D.2.: Detection of a lateral extension to a building structure, constructed between AHN4 and AHN5 epochs. The newly added portion is highlighted with high certainty of change. Orthophotos support the structural modification, source: Beeldmateriaal Nederland.

D.2. AHN-Rotterdam

Figure D.3.: Angled solar panels installed on a roof between AHN4 and AHN5 are detected with moderate to high change probabilities. Panels aligned with the roof slope are less reliably identified. Orthophotos from 2021 and 2024 are provided for context, source: Beeldmateriaal Nederland.

Figure D.4.: False change detection due to tree points within building-labelled raster cells (blue circle). In addition, removed opaque vegetation between epochs results in regions consistently labelled as changed due to persistent occlusion in the reference point cloud dataset (red circle). Orthophotos from 2021 and 2024 are provided for context, source: Beeldmateriaal Nederland.

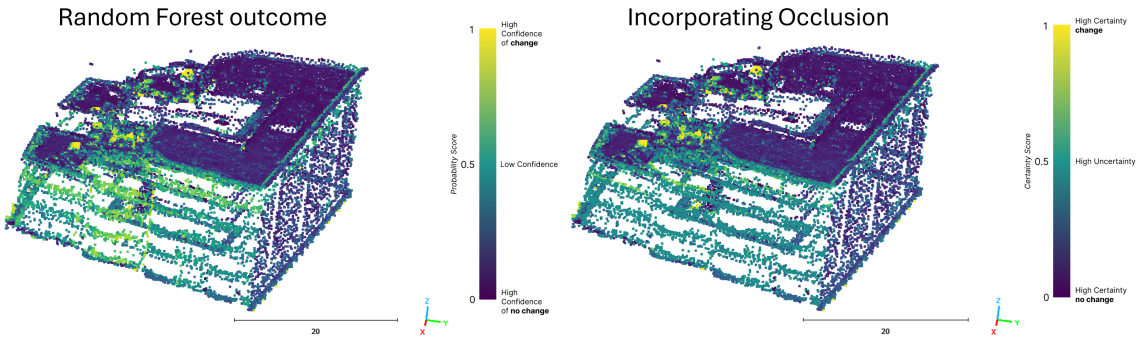


Figure D.5.: Initially detected changes on occluded walls become more uncertain after applying occlusion corrections. This reflects the model's capacity to adjust incorporate occlusion into the certainty score. Orthophotos from 2021 and 2024 are provided for context, source: Beeldmateriaal Nederland.

Figure D.6.: Newly constructed buildings are identified between AHN4 (reference point cloud dataset) and AHN5 (target point cloud dataset) point clouds. The detection algorithm assigns high change probability values to the new structures. Orthophotos from 2021 and 2024 are provided for context, source: Beeldmateriaal Nederland.

Figure D.7.: Removal of a building section detected between AHN4 and AHN5. The ground points in the former building footprint show high change probability, accurately reflecting the structural disappearance. Orthophotos from 2021 and 2024 are provided for context, source: Beeldmateriaal Nederland.

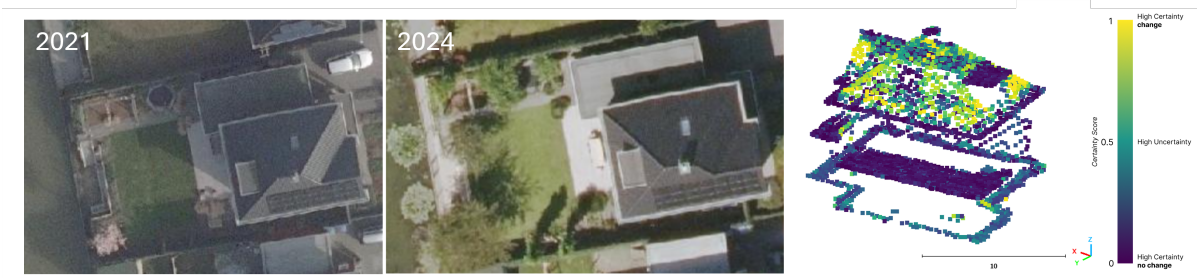


Figure D.8.: A building poorly captured in the AHN4 (reference point cloud dataset) scan results in false positive detections of change in AHN5 due to incomplete data. Change probabilities are high in areas where comparison is unreliable. Orthophotos from 2021 and 2024 are provided for context, source: Beeldmateriaal Nederland.

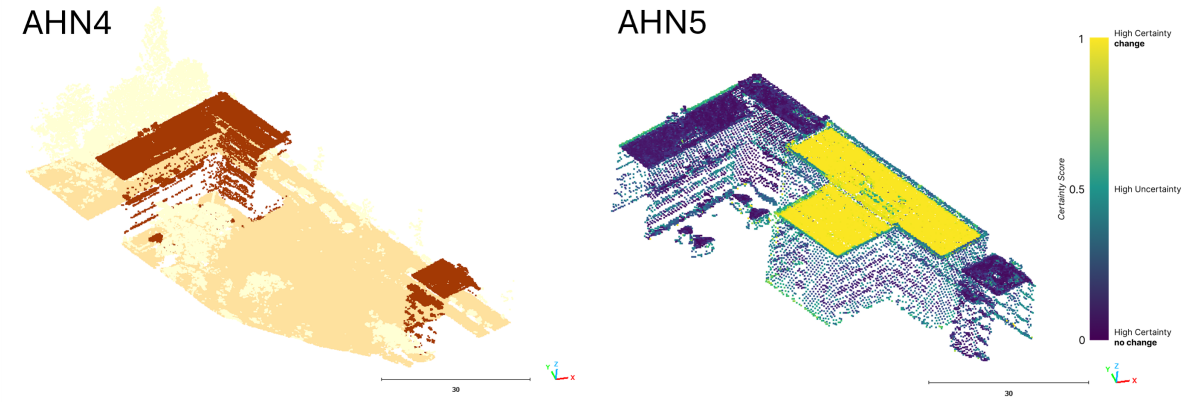


Figure D.9.: A building inserted between two existing structures is partially detected. High certainty is observed on the roof, while walls remain uncertain.

Figure D.10.: Temporary rooftop objects (e.g., cars, furniture) present in only one scan epoch are classified as changes due to their transient nature. These high-probability detections are visible in Amsterdam's dense urban areas. Orthophotos from 2021 and 2024 are provided for context, source: Beeldmateriaal Nederland.

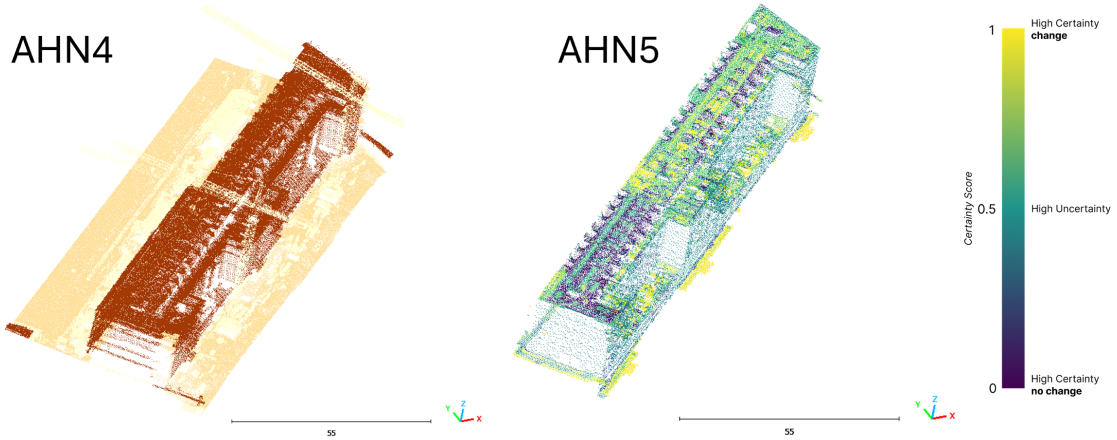


Figure D.11.: Scaffolding removed between AHN4 and AHN5 is interpreted as a structural change. The change detection algorithm highlights adjacent points with high probability where the scaffolding previously existed.

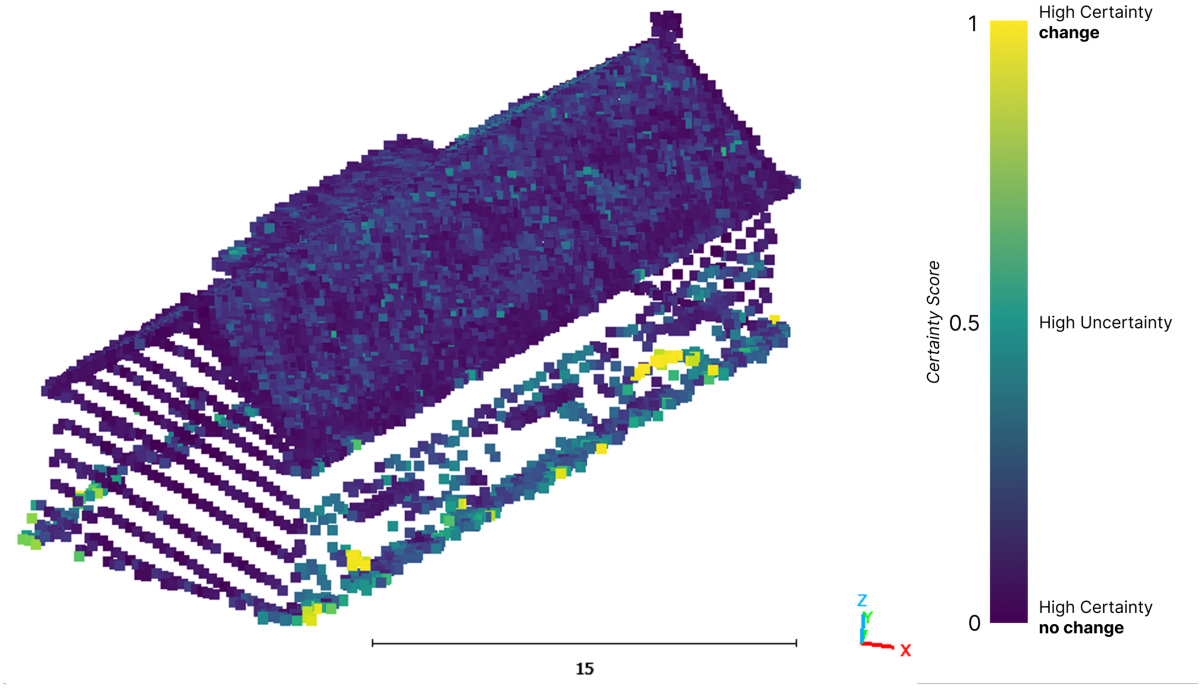


Figure D.12.: Isolated ground points misclassified as changes. These points are sparse and not visually significant.

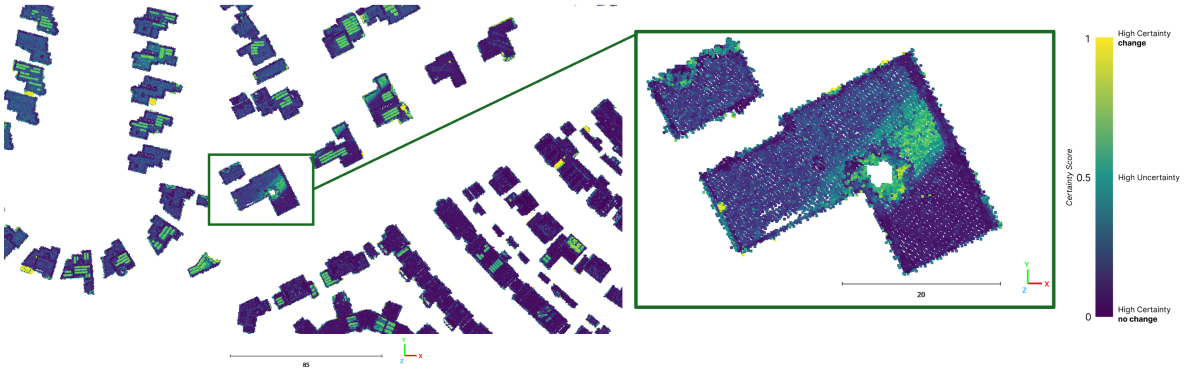


Figure D.13.: Points located approximately 129–138 meters from the flightline show reduced certainty, with change probability values between 0.5 and 0.7. This may suggest systematic uncertainty related to scan angle or density.

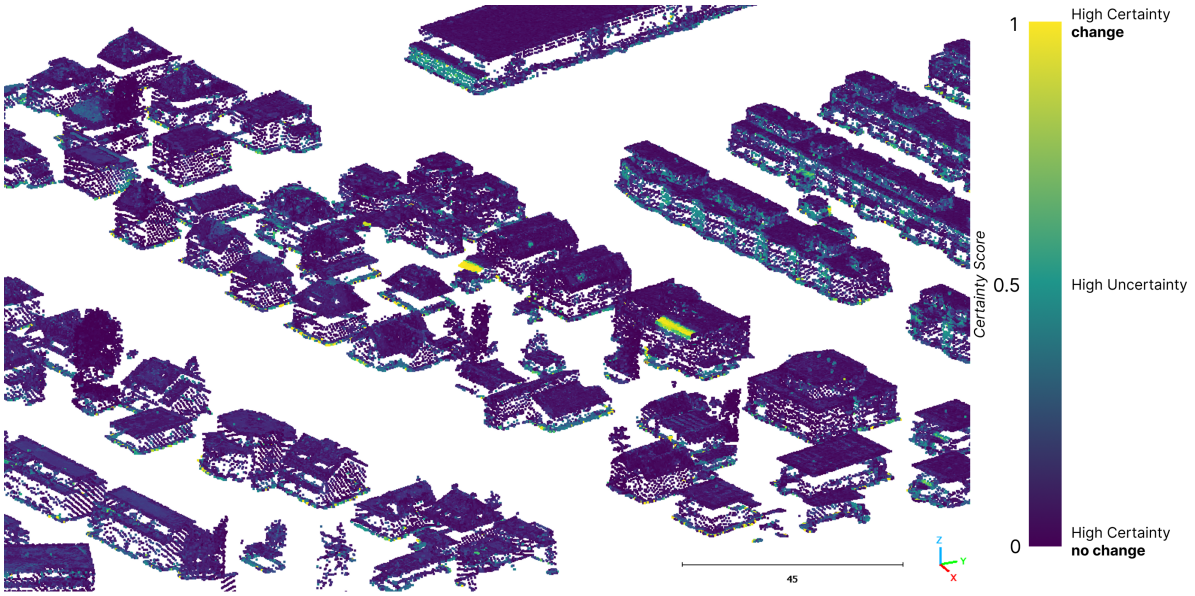


Figure D.14.: Overview of change detection between Rotterdam 2023 (reference point cloud dataset) and national height model of the Netherlands number 5 (AHN5) (target point cloud dataset).

Figure D.15.: Detection of angled solar panels installed between Rotterdam 2023 and AHN5. Panels placed at an angle are detected with moderate to high certainty of change. Those aligned with the roof slope are less reliably detected. Orthophotos from 2022 and 2024 are shown for reference point cloud dataset. Source: Beeldmateriaal Nederland.

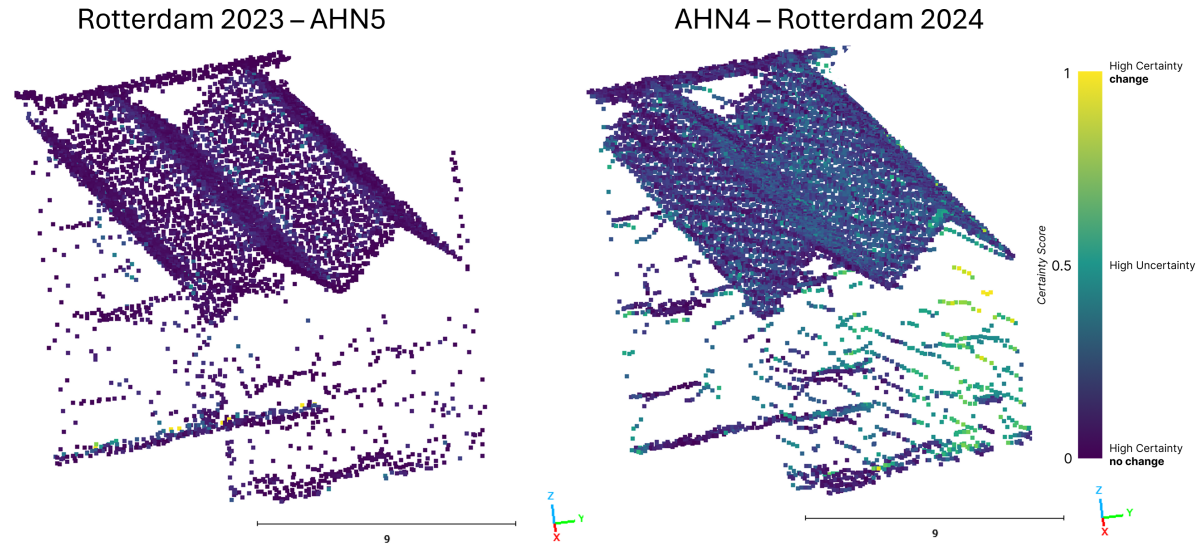


Figure D.16.: Comparison between the two dataset pairs. Left: Rotterdam 2023 (reference point cloud dataset) and AHN5 (target point cloud dataset). Right: national height model of the Netherlands number 4 (AHN4) (reference point cloud dataset) and Rotterdam 2024 (target point cloud dataset). Coloring is based on the certainty score.

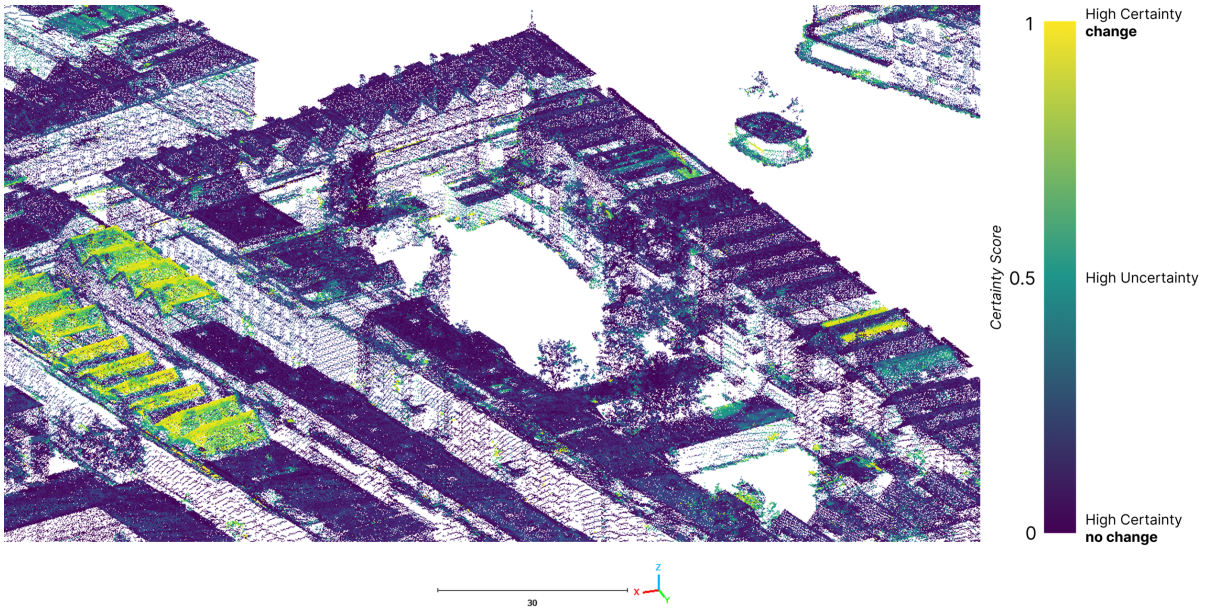


Figure D.17.: Overview of change detection between AHN4 (reference point cloud dataset) and Rotterdam 2024 (target point cloud dataset).

Figure D.18.: Change detection result showing a dormer added between AHN4 (reference point cloud dataset) and Rotterdam 2024 (target point cloud dataset). The added structure is clearly detected with high certainty of change. Orthophotos from 2021 and 2024 are included for context. Source: Beeldmateriaal Nederland.

E. Scanner Information Leica Citymapper2

Listing E.1: Custom Leica Citymapper 2 Definition in XML

```
<?xml version="1.0" encoding="UTF-8"?>
<document>
  <scanner id="leica_citymapper2"
           accuracy_m="0.05"
           beamDivergence_rad="0.00023"
           name="Leica CityMapper-2"
           optics="conic"
           pulseFreqs_Hz="2000000"
           pulseLength_ns="4"
           rangeMin_m="300"
           rangeMax_m="5500"
           scanAngleMax_deg="36"
           scanAngleEffectiveMax_deg="36"
           scanFreqMin_Hz="60"
           scanFreqMax_Hz="150"
           wavelength_nm="1064">
    <beamOrigin x="0" y="0.085" z="0.06">
        <rot axis="x" angle_deg="0" />
        <rot axis="z" angle_deg="0" />
    </beamOrigin>
    <headRotateAxis x="0" y="0" z="1"/>
  </scanner>

</document>
```

F. House Level Classifying Synthetic Dataset

This appendix describes an implementation that classifies changes at the house level using synthetic data. The goal was to reduce computational complexity by first identifying potentially changed houses, before applying detailed point-level analysis. In the end, this approach was not included in the final pipeline. First, the implementation is explained, followed by the results, and then a short discussion.

F.1. Implementation

The classification process first groups light detection and ranging (LiDAR) points by house, using the closest building from the synthetic city model (OBJ houses). Two main approaches are used: rule-based requirements and geometric distance metrics.

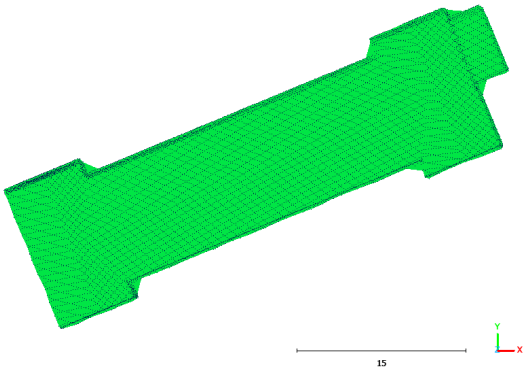
Rule-Based Requirements.

Three rule-based checks are applied to detect changes between epochs:

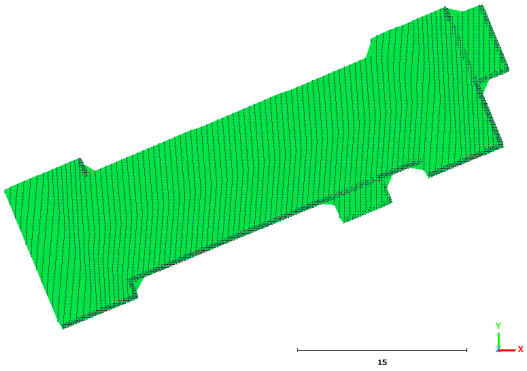
1. **Area of Concave Hull** — This requirement checks whether the 2D concave hull area of a house has changed between epochs. A significant area difference suggests a change in the house's footprint, such as an extension. An example is shown in Figure F.1.
2. **Maximum Height Difference** — This check compares the highest LiDAR point of a house across epochs. A large height difference indicates a vertical change, such as adding an extra floor. Visual results are shown in Figure F.2.
3. **Roof Shape Similarity** — This method compares the height distributions of LiDAR points using histograms. Walls are challenging to analyze because of their vertical orientation. Due to occlusion and their orientation, they are often only partially captured in point clouds, which can negatively impact the detection results. To reduce their impact, bins associated with wall surfaces are removed. A Pearson correlation coefficient is then calculated between the two normalized histograms. The full process is illustrated in Figure F.3 to Figure F.5.

The correlation coefficient r is computed as:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}, \tag{F.1}$$

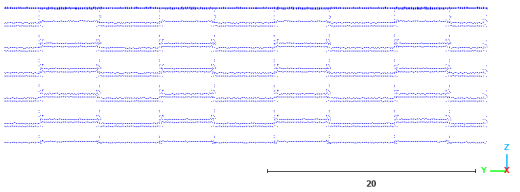


(a) LiDAR points and the 2D concave hull of a synthetic house in the first epoch, showing the original house.

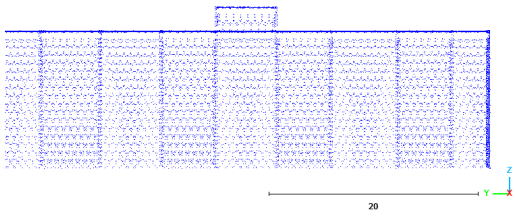


(b) LiDAR points and the 2D concave hull of the same synthetic house in the second epoch, showing an expansion.

Figure F.1.: Comparison of 2D concave hull areas between two epochs for a synthetic house with a width expansion.



(a) Side view of LiDAR points for a synthetic house in the first epoch, showing the original house.



(b) Side view of LiDAR points for the same synthetic house in the second epoch, showing an increase in height.

Figure F.2.: Height difference between two epochs for a synthetic house with a height expansion.

where x_i and y_i are the normalized histogram bin values, and $\bar{x}$, $\bar{y}$ are their respective means.

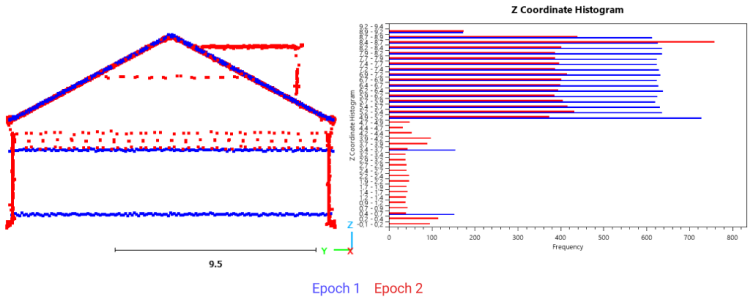


Figure F.3.: Histogram of height (z-value) distributions for a synthetic house in two epochs. Blue represents epoch 1; red represents epoch 2.

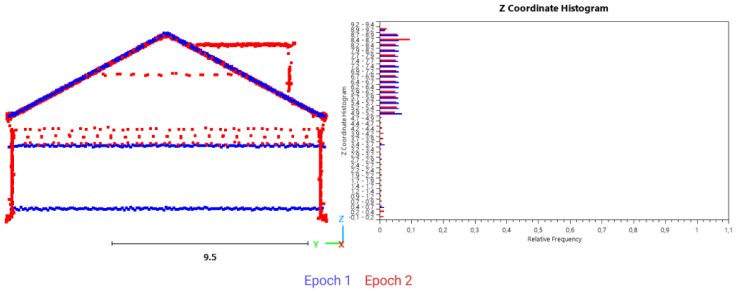


Figure F.4.: Normalized height histograms from Figure F.3, accounting for variations in point count.

Distance-Based Metrics.

Three geometric distance metrics are also calculated to measure differences between point clouds of the same house in two epochs:

- **Chamfer Distance** — Calculates the average distance between each point in one epoch and its nearest neighbor in the other.
- **One-sided Hausdorff Distance** — Measures the largest distance from a point in one epoch to the closest point in the other.
- **Point Cloud Alignment (ICP)** — Uses Iterative Closest Point (ICP) to align the two point clouds. The resulting fitness value and inlier root mean squared error (RMSE) indicate how well the clouds align.

These methods are implemented using the Point Cloud Utils and Open3D libraries.

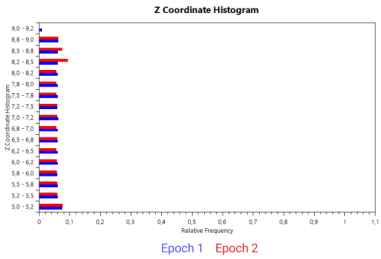


Figure F.5.: Final height histograms after removing bins representing wall surfaces, based on a frequency change threshold.

F.2. Results

The results of the three rule-based checks are shown in Figure F.6. Houses are grouped by type of change: footprint extension, height increase, or roof modification (e.g., dormers or solar panels). Changed houses are shown on the left, and unchanged houses on the right. A logarithmic scale is used in the left two graphs for clarity.

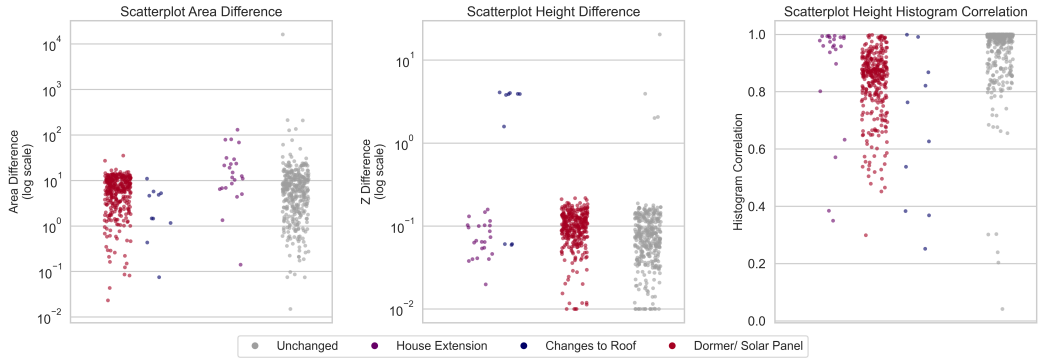


Figure F.6.: Scatterplots showing the outputs of the three rule-based requirements. Each dot represents one house. Changed houses are grouped by type: extensions, roof changes, and rooftop additions. Each requirement targets one of these types. The left side of each graph shows the changed houses; the right side shows the unchanged ones.

The six distance-based metrics are visualized in Figure F.7. Red dots represent changed houses; green dots are unchanged. A logarithmic scale is used for clarity.

F.3. Discussion

The results from the house-level classification suggest several findings:

- **Footprint Extensions (Area Change)** — The method struggled to separate changed houses from unchanged ones using the concave hull area. With the lowest possible threshold, a lot of unchanged houses were falsely marked as changed. This may be due

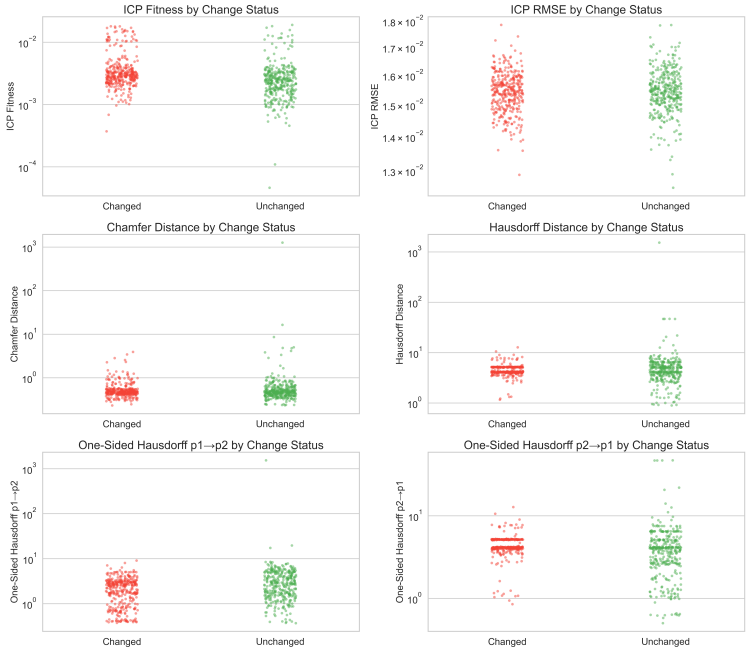


Figure F.7.: Scatterplots showing the results of six distance metrics. Red dots are changed houses; green dots are unchanged. Each plot represents one metric.

to difficulties in concave hull generation on synthetic data with linear point patterns. The alpha parameter used for the concave hull significantly affects the result, especially near corners or gaps.

- **Vertical Changes (Height Difference)** — This requirement worked well when the highest point of a house was changed (e.g., roof raised). However, it was less effective for localized height changes, such as small rooftop extensions. Partial modifications that do not affect the highest point were often missed.
- **Roof Modifications (Histogram Correlation)** — This method showed promising results. In general, changed and unchanged houses had different correlation values. However, some changed houses had correlations close to 1.0, making it difficult to define a strict threshold. Several factors could influence this:
  - Removing wall bins may also remove roof data, especially with complex roof shapes like gables.
  - Uneven point density due to flight lines passing directly over part of a house.
  - For large houses, small changes like a single dormer may not significantly impact the height histogram.

Although the rule-based approach is limited, it offers useful insights. Some suggestions for improvement include:

- **Raster-Based Comparison** — Convert house-level point clouds into raster grids. Label cells as wall or roof, and compare height histograms only for roof cells.

- **Graph-Based Similarity** — Model each house as a graph (e.g., nodes = segments, edges = connections), and compute similarity between graphs across epochs.

G. Pseudocodes

G.1. Labelling Pseudocodes

Algorithm G.1: Assign LiDAR points to house polygons using spatial querying

Input: `Houses` – List of house polygons;
`Raster` – 2D grid where each cell contains zero or more LiDAR points
Output: List of houses with LiDAR points spatially assigned to the corresponding list of polygons (house)

```
1  Function MatchLiDARPointsWithHouses(Houses, Raster):
2      Build spatial index from house geometries;
3      foreach raster cell in Raster do
4          if cell contains building points then
5              Get bounding box and geometry of the cell;
6              Query index for candidate houses;
7              if exactly one candidate then
8                  Assign all points to that house;
9              else if multiple candidates then
10                 Filter intersecting houses;
11                 if none intersect then
12                     Find closest house by distance;
13                     Assign all points to that house;
14                 else
15                     foreach point in cell do
16                         Assign to closest intersecting house;
17             else
18                 Expand buffer until candidates found;
19                 Assign points to closest house;
20     return Updated list of Houses;
```

Algorithm G.2: Assign LiDAR points to individual polygonal faces within each house

Input: `Houses` – List of houses with points assigned (output of Algorithm G.1)
Output: Each face of every house contains a list of 0 or more spatially matched LiDAR points

```
Function MatchPointsToPolygonFaces(Houses):
    foreach house in Houses do
        if house is unchanged then
            Assign all points to the first face;
            continue;
        foreach point in house do
            threshold ← 0.2;
            while threshold ≤ 8 do
                Find nearby candidate faces;
                foreach face in candidates do
                    if point is within threshold then
                        Compute distance to face;
                        if distance is smallest so far then
                            Store as closest face;
                if a face was matched then
                    Assign point to face;
                    break;
                else
                    threshold ← threshold +0.1;
    return Updated Houses with face-level point mapping;
```

Algorithm G.3: Match corresponding houses between two epochs and identify additions/removals

Input: `HousesEpoch1`, `HousesEpoch2` – List of houses with per-face points for epoch 1 and 2 (from Algorithm G.2)
Output: Matched house pairs; lists of unmatched (added/removed) houses

```
Function MatchHousesOnHouseLevel(HousesEpoch1, HousesEpoch2):
    Initialize matchedHouses, notMatchedEpoch1, notMatchedEpoch2;
    foreach house1 in HousesEpoch1 do
        Compute convex hull and area;
        bestMatch ← null, bestScore ← 0;
        foreach house2 in HousesEpoch2 do
            Compute intersection area;
            if intersection ratio > 0.6 then
                Compute symmetric intersection score;
                if score > bestScore then
                    bestMatch ← house2;
        if bestMatch exists then
            Add to matchedHouses;
            Remove bestMatch from HousesEpoch2;
        else
            Label house1 as removed;
            Add to notMatchedEpoch1;
    foreach remaining house in HousesEpoch2 do
        Label as added;
        Add to notMatchedEpoch2;
    return matchedHouses, notMatchedEpoch1, notMatchedEpoch2;
```

Algorithm G.4: Complete building change labelling pipeline using LiDAR data from two epochs

Input: `CityEpoch1, CityEpoch2` – List of houses for each epoch;
`Raster1, Raster2` – Corresponding 2D LiDAR rasters
Output: Labelled house objects, optionally saved per house or per change class

```
Function GeneralLabelling(CityEpoch1, CityEpoch2, Raster1, Raster2):
    Points1 ← GeneralLabelling(MatchLiDARPointsWithHouses(CityEpoch1, Raster1));
    Points2 ← GeneralLabelling(MatchLiDARPointsWithHouses(CityEpoch2, Raster2));
    Faces1 ← GeneralLabelling(MatchPointsToPolygonFaces(Points1));
    Faces2 ← GeneralLabelling(MatchPointsToPolygonFaces(Points2));
    matched, unmatched1, unmatched2 ← GeneralLabelling(MatchHousesOnHouseLevel(Faces1, Faces2));
    if SaveHouseLevelResults then
        foreach matched house pair do
            Save points and face labels to LAZ;
        foreach unmatched house do
            Save full building as added or removed;
    if SavePointLevelResults then
        Group points by face labels;
        Export grouped points;
```

G.2. Occlusion Detection

Algorithm G.5: positionsAircraft: Initialization and flightline filtering

Input: `epochFlightlines` – List of flightlines, including flightstrip information;
`Scanner` – Scanner object with scanner details;
`Synthetic` – Boolean flag indicating synthetic (true) or real data (false);
`Reference` – Boolean flag for first occlusion boolean calculation (see Section 5.6).

Output: `closestFlightlines` – Candidate flightlines possibly seeing the point;
`XYZPoints, XYPoints, kdtree` – Points used to check occlusion between aircraft and target point.

```
1  if Reference then
2      XYZPoints ← XYZReferenceNeighbours
3      XYPoints ← XYReferenceNeighbours
4      kdtree ← ReferencePointCloudKDtree
5  else
6      if Synthetic then
7          foreach flightLineEntry in epoch_flightLines do
8              Determine FinalPosition by interpolating GPSTime on flightLineEntry
                 coordinates
9              if FinalPosition is not null then
10                 Add FinalPosition to positionAircrafts
11         return positionAircrafts
12     XYZPoints ← XYZTargetNeighbours
13     XYPoints ← XYTargetNeighbours
14     kdtree ← TargetPointCloudKDtree
15 Initialize empty dictionary closestFlightLines
16 foreach line in epoch_flightLines do
17     if line.GeometryFlightLine ≠ null and line.StripEnvelope contains PointOfInterest then
18         if line.GeometryStrip contains PointOfInterest then
19             closestFlightLines[line] ← distance from line.GeometryFlightLine to
                 PointOfInterest
```

Algorithm G.6: positionsAircraft: Synthetic final position estimation per flightline, continuation of Algorithm G.5.

Input: See Algorithm G.5.
Output: `positionAircrafts` – Estimated aircraft positions (x, y, z) that can see the target point.

```
if Synthetic then
    if XYZPoints contains points with classification 6 then
        Neighbours ← XYZPoints
    else if XYPoints contains points with classification 6 then
        Neighbours ← XYPoints
    else
        Initialize ClosestNeighbours as empty list
        newThreshold ← ThresholdNeighbours + 0.5
        while no point in ClosestNeighbours has classification 6 do
            Define EnvelopePoint with newThreshold around PointOfInterest
            ClosestNeighbours ← kdtree.Query(EnvelopePoint)
            newThreshold ← newThreshold + 1
        Neighbours ← ClosestNeighbours
    foreach (flightLine, distance) in closestFlightLines do
        Get coords from flightLine.GeometryFlightLine
        Initialize PointsCurrentFlightline as empty list
        foreach nb in Neighbours do
            if nb.GpsTime in [tStart, tEnd] then
                Add nb to PointsCurrentFlightline
        if PointsCurrentFlightline is empty then
            continue
        FinalPosition ← null
        ClosestDistance ← ∞
        foreach nb in PointsCurrentFlightline do
            distance ← Distance3D(PointOfInterest, nb)
            if distance < ClosestDistance then
                Interpolate position on coords at nb.GpsTime
                ClosestDistance ← distance
                Update FinalPosition
        if FinalPosition ≠ null then
            Add FinalPosition to positionAircrafts
```

Algorithm G.7: positionsAircraft: Handling scanner types for real data (non-synthetic), continuation of Algorithm G.5.

Input: See Algorithm G.5.
Output: `positionAircrafts` – Estimated aircraft positions (x, y, z) that can see the target point.

```
if Scanner.ScannerName = "Leica CityMapper - 2" then
    foreach line in closestFlightLines do
        Compute radius_circle based on FOV and average height
        Create buffer circle around PointOfInterest with radius_circle
        Find intersection points between circle boundary and flightline
        Add intersection points to positionAircrafts
else if Scanner.ScannerName = "Riegl VQ-780i" then
    foreach line in closestFlightLines do
        Find closest point on flightline segment to PointOfInterest
        Compute height of closest point by linear interpolation
        Add closest point with computed height to positionAircrafts
else if Scanner.ScannerName = "Riegl VQ-1560i" then
    foreach flightLine in closestFlightLines do
        Compute angles in radians
        Determine if PointOfInterest is left, right or collinear to flightline segment
        if collinear then
            Compute forward/backward offset and intersection points
            Add intersection points to positionAircrafts
            continue
        Compute distances and tilted distances
        Compute two candidate tilted points on line segment
        Extend lines from PointOfInterest through tilted points
        Compute intersection points of extended lines with flightline
        Calculate aircraft altitudes and forward/backward offsets
        Compute buffered zones and intersection scans
        Determine forwardLeft and backwardRight aircraft positions based on geometry
        Add these positions to positionAircrafts
```

Algorithm G.8: isOccluded: Determine if target point is occluded from all given aircraft positions

Input: `positionsAircrafts` – List of aircraft positions (x, y, z);
`PointTree` – Octree of points in the scene;
`Scanner` – Scanner parameters including accuracy and laser divergence.
Output: `occluded` – Boolean indicating if target point is occluded (true) or visible (false).

```
laserDivergence ← Scanner.LaserDivergence_mrad / 1000
targetPoint ← vector(PointOfInterest.X, PointOfInterest.Y, PointOfInterest.Z)
foreach positionAircraft in positionsAircrafts do
    aircraftPos ← vector(positionAircraft.X, positionAircraft.Y, positionAircraft.Z)
    direction ← normalize(aircraftPos - targetPoint)
    rayLength ← distance(aircraftPos, targetPoint)
    ray ← Ray(origin=targetPoint, direction=direction)
    thresholdError ← 2 × tan(laserDivergence/2) × positionAircraft.Z + 2 ×
      (Scanner.HorizontalAccuracy + Scanner.VerticalAccuracy)
    ptsAlongRay ← points in PointTree near ray within thresholdError filtered by projection
      distance along ray between thresholdError and rayLength
    if count(ptsAlongRay) ¡ 3 then
        return false
    end
end
return true
```

Algorithm G.9: CalculateOcclusionPoint: Main occlusion boolean calculation for target point

Input: `flightLinesEpoch` – List of flightlines for the epoch;
`PointTree` – Octree of points;
`ScannerReference` – Scanner parameters;
`Synthetic` – Boolean flag if data is synthetic;
`OccludedReference` – Boolean flag for first occlusion boolean calculation.
Output: `occluded` – Boolean indicating if target point is occluded (true) or visible (false).

```
positionsAircrafts ← positionsAircraft(flightLinesEpoch, ScannerReference, Synthetic,
  OccludedReference)
occluded ← isOccluded(positionsAircrafts, PointTree, ScannerReference)
return occluded
```

Bibliography

Bacher, U. (2022). HYBRID AERIAL SENSOR DATA AS BASIS FOR A GEOSPATIAL DIGITAL TWIN. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLIII-B4-2022:653–659. Conference Name: XXIV ISPRS Congress "Imaging today, foreseeing tomorrow", Commission IV - 2022 edition, 6–11 June 2022, Nice, France Publisher: Copernicus GmbH.

Belgiu, M. and Drăguţ, L. (2016). Random forest in remote sensing: A review of applications and future directions. *ISPRS Journal of Photogrammetry and Remote Sensing*, 114:24–31.

Biau, G. and Scornet, E. (2016). A random forest guided tour. *Test*, 25(2):197–227. Num Pages: 197-227 Place: Heidelberg, Netherlands Publisher: Springer Nature B.V.

Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S., and Çöltekin, A. (2015). Applications of 3D City Models: State of the Art Review. *ISPRS International Journal of Geo-Information*, 4(4):2842–2889. Number: 4 Publisher: Multidisciplinary Digital Publishing Institute.

Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1):5–32.

de Gélis, I., Lefèvre, S., and Corpetti, T. (2021a). 3D URBAN CHANGE DETECTION WITH POINT CLOUD SIAMESE NETWORKS. *ISPRS International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLIII-B3-2021:879–886. Publisher: Copernicus GmbH (Copernicus Publications).

de Gélis, I., Lefèvre, S., and Corpetti, T. (2021b). Change Detection in Urban Point Clouds: An Experimental Comparison with Simulated 3D Datasets. *Remote Sensing*, 13(13):2629. Number: 13 Publisher: Multidisciplinary Digital Publishing Institute.

De Gélis, I., Lefèvre, S., Corpetti, T., Ristorcelli, T., Thénoz, C., and Lassalle, P. (2021). Benchmarking Change Detection in Urban 3D Point Clouds. In *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*, pages 3352–3355. ISSN: 2153-7003.

Diaz, V., van Oosterom, P., Meijers, M., Verbree, E., Ahmed, N., and van Lankveld, T. (2024a). Comparison of Cloud-to-Cloud Distance Calculation Methods - Is the Most Complex Always the Most Suitable? In Kolbe, T. H., Donaubauer, A., and Beil, C., editors, *Recent Advances in 3D Geoinformation Science*, pages 329–334, Cham. Springer Nature Switzerland.

Diaz, V., van Oosterom, P., Meijers, M., Verbree, E., van Lankveld, T., and Ahmed, N. (2024b). Fast SFC-based point cloud change detection.

Eberly, D. (2024). Least Squares Fitting of Data by Linear or Quadratic Structures.

Evers, D., van Bemmel, B., and Spoon, M. (2023). Quickscan toename van het ruimtebeslag in Nederland. Technical Report 5152, Planbureau voor de Leefomgeving, Den Haag.

Google (2021). Google Maps.

Hebel, M., Arens, M., and Stilla, U. (2013). Change detection in urban areas by object-based analysis and on-the-fly comparison of multi-view ALS data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 86:52–64.

Kashani, A. G., Olsen, M. J., Parrish, C. E., and Wilson, N. (2015). A Review of LIDAR Radiometric Processing: From Ad Hoc Intensity Correction to Rigorous Radiometric Calibration. *Sensors*, 15(11):28099–28128. Number: 11 Publisher: Multidisciplinary Digital Publishing Institute.

Kharroubi, A., Poux, F., Ballouch, Z., Hajji, R., and Billen, R. (2022). Three Dimensional Change Detection Using Point Clouds: A Review. *Geomatics*, 2(4):457–485. Number: 4 Publisher: Multidisciplinary Digital Publishing Institute.

Kharroubi, A., Remondino, F., Ballouch, Z., Hajji, R., and Billen, R. (2025). Semantic and Geometric Fusion for Object-Based 3D Change Detection in LiDAR Point Clouds. *Remote Sensing*, 17(7):1311. Number: 7 Publisher: Multidisciplinary Digital Publishing Institute.

Lague, D., Brodu, N., and Leroux, J. (2013). Accurate 3D comparison of complex topography with terrestrial laser scanner: Application to the Rangitikei canyon (N-Z). *ISPRS Journal of Photogrammetry and Remote Sensing*, 82:10–26.

Lemmens, M. (2018). Point Clouds and Smart Cities: The Need for 3D Geodata and Geomatics Specialists. *GIM International: The Worldwide Magazine for Geomatics*, 32(5):24–30.

Liu, H. (2022). A+BE | Architecture and the Built Environment, No. 12 (2022): nD-PointCloud Data Management.

Liu, Z., van Oosterom, P., Balado, J., Swart, A., and Beers, B. (2022). Detection and reconstruction of static vehicle-related ground occlusions in point clouds from mobile laser scanning. *Automation in Construction*, 141:104461.

Manders, N. (2024). Comparing AHN point clouds for their performance in representing 3D buildings in Zuid-Holland; A quantitative and qualitative performance review between AHN3 and AHN4. Master's thesis. Accepted: 2024-01-09T00:00:55Z.

Mayr, A., Bremer, M., and Rutzinger, M. (2020). 3D POINT ERRORS AND CHANGE DETECTION ACCURACY OF UNMANNED AERIAL VEHICLE LASER SCANNING DATA. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, V-2-2020:765–772. Conference Name: XXIV ISPRS Congress, Commission II (Volume V-2-2020) - 2020 edition Publisher: Copernicus GmbH.

Niculescu-Mizil, A. and Caruana, R. (2005). Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning - ICML '05*, pages 625–632, Bonn, Germany. ACM Press.

Nofulla, J. (2023). Deep Learning-based Change Detection and Classification for Airborne Laser Scanning Data.

Politz, F. and Sester, M. (2022). Building Change Detection in Airborne Laser Scanning and Dense Image Matching Point Clouds Using a Residual Neural Network. In *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume XLIII-B2-2022, pages 625–632, Gottingen, Germany. Copernicus GmbH. ISSN: 16821750 Num Pages: 625-632.

Psomadaki, S. (2016). Using a space filling curve for the management of dynamic point cloud data in a Relational DBMS. Master's thesis, Delft University of Technology, Delft.

Qin, R., Tian, J., and Reinartz, P. (2016). 3D change detection – Approaches and applications. *ISPRS Journal of Photogrammetry and Remote Sensing*, 122:41–56.

RIEGL Laser Measurement Systems GmbH (2017). RIEGL VQ-1560i at a Glance.

Scaioni, M., Höfle, B., Baungarten Kersting, A. P., Barazzetti, L., Previtali, M., and Wujanz, D. (2018). METHODS FROM INFORMATION EXTRACTION FROM LIDAR INTENSITY DATA AND MULTISPECTRAL LIDAR TECHNOLOGY. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-3:1503–1510.

Schütz, M., Ohrhallingerm, S., and Wimmer, M. (2020). Fast Out-of-Core Octree Generation for Massive Point Clouds. *John Wiley & Sons, Inc.*, 39:13.

Stilla, U. and Xu, Y. (2023). Change detection of urban objects using 3D point clouds: A review. *ISPRS Journal of Photogrammetry and Remote Sensing*, 197:228–255.

Tran, T. H. G., Ressl, C., and Pfeifer, N. (2018). Integrated Change Detection and Classification in Urban Areas Based on Airborne Laser Scanning Point Clouds. *Sensors*, 18(2):448. Number: 2 Publisher: Multidisciplinary Digital Publishing Institute.

van Bochove, D. (2019). COMBINING MLS & ALS POINT CLOUD DATA.

van Natijne, A. and Optical and Laser Remote Sensing group TU Delft (2025). GeoTiles: readymade geodata with a focus on the Netherlands.

van Oosterom, P. (1999). Spatial Access Methods. In Paul A. Longley, Michael F. Goodchild, David J. Maguire, and David W. Rhind, editors, *Geographical Information Systems: Principles, Technical Issues, Management Issues, and Applications*, volume 1, pages 385–400. John Wiley & Sons, Chichester UK.

van Oosterom, P. (2016). nD-PointClouds a model for deeply integrating space, time and scale.

van Oosterom, P., van Oosterom, S., Liu, H., Thompson, R., Meijers, M., and Verbree, E. (2022). Organizing and visualizing point clouds with continuous levels of detail. *ISPRS Journal of Photogrammetry and Remote Sensing*, 194:119–131.

Vosselman, G. (2010). *Airborne and Terrestrial Laser Scanning*. Whittles Publishing Ltd, Dunbeath, UNITED KINGDOM.

Winiwarter, L., Anders, K., and Höfle, B. (2021). M3C2-EP: Pushing the limits of 3D topographic point cloud change detection by error propagation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 178:240–258.

Winiwarter, L., Esmorís Pena, A. M., Weiser, H., Anders, K., Martínez Sánchez, J., Searle, M., and Höfle, B. (2022). Virtual laser scanning with HELIOS++: A novel take on ray tracing-based simulation of topographic full-waveform 3D laser scanning. *Remote Sensing of Environment*, 269:112772.

Xiao, W., Cao, H., Tang, M., Zhang, Z., and Chen, N. (2023). 3D urban object change detection from aerial and terrestrial point clouds: A review. *International Journal of Applied Earth Observation and Geoinformation*, 118:103258.

Xiao, W., Vallet, B., and Paparoditis, N. (2013). Change Detection in 3D Point Clouds Acquired by a Mobile Mapping System. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, II-5-W2:331–336.

Xu, S., Vosselman, G., and Oude Elberink, S. (2015). Detection and Classification of Changes in Buildings from Airborne Laser Scanning Data. *Remote Sensing*, 7(12):17051–17076. Number: 12 Publisher: Multidisciplinary Digital Publishing Institute.

Yıldırım, S. (2020). Decision Tree and Random Forest - Explained.

Colophon

This document was typeset using LaTeX, using the KOMA-Script class `scrbook`. The main font is Palatino.