EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mechanical Engineering

Synthetic Data-Driven Mismatch Detection for Updating BIM-Based Digital Twins from a Single Viewpoint

by

P.C.J. Hundepool

MSc Thesis

Thesis committee		**Graduation**	
Chair:	Elena Torta, dr.	Program:	Artificial Intelligence and Engineering Systems
Member 1:	Pieter Pauwels, dr. ir.	Research group:	Control Systems Technology
Member 2:	Jos Elfring, dr. ir.	Thesis supervisor:	Elena Torta
Advisory member:	Koen de Vos	Date of defense:	26/06/2025
		Student ID:	1427520
		Study load (ECTS):	45
		Track:	High-tech systems and robotics
		RBT number:	RBTMA25023

This thesis is public and Open Access.

This thesis has been realized in accordance with the regulations as stated in the TU/e Code of Scientific Conduct.

Disclaimer: the Department of Mechanical Engineering of the Eindhoven University of Technology accepts no responsibility for the contents of MSc theses or practical training reports.

***Abstract*—This thesis presents a simulation-based pipeline for detecting geometric mismatches between a prior Building Information Model (BIM) and acquired 3D scan data, with a focus on structural elements in indoor environments. A fully controllable synthetic data generation pipeline is developed using Blender and the BLAINDER range scanner tool to simulate realistic 3D sensor scans of both clean and modified environments, simulating mobile robot perception. Structural mismatches such as added or removed walls and pillars are introduced and labels are generated through geometric comparison and cluster-based filtering, supporting supervised learning. Four scenario-specific mismatch detection models are trained, each targeting a distinct object-mismatch combination (e.g., pillar-added or wall-removed), using the same lightweight 1D convolutional neural network architecture that operates on preprocessed point clouds and outputs per-point mismatch probabilities. After detection, bounding boxes are fitted to clusters of mismatched points to localize affected objects and to act as a bridge between per-point predictions and structured BIM updates by converting detection outputs into interpretable object-level changes. This work contributes with a robust baseline for object-centric mismatch detection, provides a foundation for updating digital twins in real-time using mobile robots, and supports future extensions to real-world deployment and BIM-integrated scene editing.**

I. Introduction

The Architecture, Engineering, and Construction (AEC) industry is increasingly driven by digitalization, with Building Information Models (BIMs) getting more attention as foundational tools for representing, analyzing, and managing buildings and built infrastructure. A Building Information Model (BIM) is a structured digital representation of a building's geometric, semantic, and functional properties, commonly used during the design and construction phases [1]. A BIM-based digital twin refers to a dynamic, often server-based counterpart that reflects the state of a physical building in real-time and serves as a continuously updated data source throughout the building's operational lifecycle.

As BIM software becomes more often used, keeping the digital model synchronized with the actual state of the physical building becomes an important challenge. Buildings are dynamic, often undergoing changes such as construction, renovations or daily operations. This misalignment between the digital twin and the physical world can lead to poor decisions, safety assessments, and facility operations.

One possible solution to reduce inconsistencies between the digital model and the real world involves integrating robotics into building monitoring workflows. Mobile robots equipped with 3D sensors can autonomously scan environments, detect discrepancies, and assist in keeping digital twins up-to-date. This could help update the models more accurately and efficiently, with potential uses in construction, building inspections, and facility management [2].

A. Related Work

This section reviews prior work relevant to the integration of 3D point cloud data for digital twin updates, with a focus on indoor environments. Three main topics are covered. First, digital twin synchronization is examined, focusing on the challenge of keeping digital twins up to date using sensed data. Next, the use of synthetic data for 3D deep learning is discussed, motivating the choice to rely on simulated 3D sensor scans and labels due to the limited availability of high-quality real-world datasets. Finally, deep learning methods for mismatch detection and point cloud segmentation are reviewed. Together, these topics provide the foundation for the approach developed in this project.

1) Synchronizing Digital Twins with Real-World Data: BIM modelling tools do not only serve as design and planning tools but also as operational references for maintenance, renovation, and inspection workflows [1]. In recent years, the idea of a digital twin has expanded on this by linking BIMs to real-time or regularly updated sensor data, aiming to create a dynamic representation of the current state of a building [3].

However, synchronizing a BIM model with the real world remains a significant challenge. Changes to building geometry (e.g., moved walls, added elements) during construction or daily operations can quickly cause the model to be outdated [4]. This misalignment limits the reliability of digital twins for supporting accurate decisions, especially in settings where safety or maintenance are critical.

Existing approaches to updating digital twins can in general be classified into three categories. First there is manual updating, which is labor-intensive and error-prone [5]. Secondly we have rule-based or geometric matching methods, which compare as-built scans to digital twins using handcrafted thresholds or CAD-model alignment strategies [6]. Thirdly there are SLAM-integrated methods, which use simultaneous localization and mapping to continuously scan indoor environments and detect changes [7].

Despite progress in these areas, many digital twin update processes still rely heavily on manual input or make assumptions about ideal scanning conditions. Real-world issues like occlusions, incomplete sensor coverage, and a lack of semantic labeling in raw point

clouds remain key obstacles. As a result, fully automating these processes remains a long-term challenge, with user interpretation still playing a crucial role [4].

This motivates the development of more robust, object-aware methods for mismatch detection and digital twin synchronization, especially those that can be trained on synthetic data and operate without manual segmentation or registration.

2) Synthetic Data for 3D Deep Learning: Training deep learning models on 3D data typically requires large, well-annotated datasets. However, acquiring such datasets from real-world scans is labor-intensive and time-consuming. To overcome this, the usage of synthetic data generated in simulation environments like Blender is used, often with the help of tools such as BLAINDER [8]. These platforms enable the creation of diverse and customizable 3D environments and the generation of synthetic point clouds with precise annotations. For example, Fedorova et al. [9] developed a synthetic 3D data generation pipeline for geometric deep learning in architecture, enabling the production of varied datasets suitable for multiple deep learning tasks. Furthermore, Van Den Akker [10] created a parametric dataset generator in Blender that is able to create random three-dimensional rooms with a small set of different object types whose dimensions can also be randomized.

While synthetic data provides clear advantages, like perfect ground-truth labels and flexibility in design, it also introduces the challenge of the domain gap, which refers to the distribution mismatch between synthetic and real-world data [11]. This gap occurs due to differences in sensor noise, lighting conditions, and object textures, which can affect the generalization of models that are only trained on synthetic data. Zhao et al. [12] highlight that synthetic-to-real domain generalization remains a challenging problem in 3D indoor point cloud segmentation, mostly due to differing layouts and point patterns between domains.

To reduce these mismatches between synthetic and real-world data, techniques like domain adaptation and the inclusion of realistic noise models during data generation are used. Tremblay et al. [13] propose domain randomization as a way to improve generalization by varying simulation parameters such as object types, scene layout, and camera viewpoint.

Controlled synthetic data offers several benefits in supervised learning tasks. It ensures high-quality annotations, repeatability, and the ability to generate balanced datasets across various classes, something that is difficult to achieve in real-world settings due to class imbalance, as discussed by Buda et al. [14]. This control is particularly useful in domains like mismatch detection, where annotated real-world data is scarce.

3) Deep Learning for Mismatch Detection and Segmentation: Detecting changes or mismatches in point clouds is a critical task in applications such as digital twin updates, robotic navigation, and scene reconstruction. Traditional methods such as Iterative Closest Point (ICP) [15], point cloud differencing [16], and geometry alignment [17] provide foundational approaches but often fail in real-world scenarios. Zhang et al. report that ICP and other local registration methods assume significant overlap and accurate initialization, and they become unreliable, easily trapped in local minima, when overlap is limited or noise is present [18]. Moreover, in multi-temporal scans, real-world point clouds suffer from noise and occlusion, which complicates both alignment and change detection [19].

Deep learning has opened new possibilities by offering more robust and scalable alternatives. PointNet [20] introduced a fresh approach to processing point clouds by directly consuming unordered point sets, addressing the challenge of permutation invariance. The architecture applies a series of Multi-Layer Perceptrons (MLPs) to each point independently, followed by a symmetric function, typically max pooling, to aggregate global features. This design ensures that the network's output is invariant to the order of input points.

However, while PointNet effectively captures global features, it lacks the ability to model local structures. To address this, PointNet++ [21] was introduced, incorporating hierarchical feature learning by grouping points into local regions and applying PointNet recursively. This enhancement enables the network to capture both local and global features, improving performance on tasks requiring fine-grained understanding.

Further advancements have explored alternative aggregation functions and attention mechanisms to enhance feature representation [22]. For example, learnable pooling operations and kernel-based methods have been proposed to better capture the complex structures within point clouds [23].

With Siamese Networks [24], two point clouds are processed through shared-weight networks. Siamese architectures can learn to identify differences between them. This approach is particularly effective for change

detection, as it focuses on learning discriminative features between pairs.

Kernel Point Convolution (KPConv) [25] is a convolutional approach designed for point clouds, KPConv allows for flexible and deformable kernel applications, capturing local geometric structures effectively. KPConv has also been adapted into Siamese-style models for improved change detection [26].

These deep learning methods offer improved robustness to noise and partial data, learning complex features that traditional methods might miss. Mismatch detection can be approached at different levels. One approach is Per-Point Classification [27], which assigns a change/no-change label to each point, enabling detailed localization of changes. This approach is beneficial for applications that require precise change maps, which are spatial representations that highlight which parts of the environment have changed over time, like scene reconstruction and map updating [28]. In dynamic environments, point-wise segmentation aids in identifying and updating changes, ensuring that maps remain current and accurate [29]. Another application is autonomous navigation. For autonomous vehicles and robots, understanding the environment is crucial for obstacle avoidance and path planning. Per-Point segmentation provides detailed semantic information, enabling precise identification of object boundaries and fine structures. However, it can be computationally intensive and may struggle with consistency in regions that are geometrically or visually similar. In such regions, small local differences may not provide enough distinctive features for the network to confidently separate adjacent objects or structures, leading to misclassifications along boundaries and noisy segmentation outputs [23].

An alternative method is object-level segmentation, which groups points into objects before labeling them. This is efficient in processing and provides a higher level of understanding of objects, which is helpful in object tracking or recognition, but it may overlook fine details and struggle with objects that have complex geometries or are partially occluded.

Another approach is Scene-Level Comparison [30], which evaluates the overall similarity between point clouds. While less granular, it is useful for applications where the presence of change is more critical than its exact location.

The choice between these approaches depends on the specific requirements of the application, balancing the need for detail, robustness, and computational efficiency. In this work, point-level classification is used to enable fine-grained localization of structural mismatches and to support downstream clustering and bounding box fitting. This approach avoids the need for shape completion techniques, often required in object-level methods and especially in the presence of occlusion and partial visibility [31], and allows for more lightweight and flexible processing of large-scale indoor scans.

B. Research question

This research investigates whether structural mismatches in indoor environments can be detected from a single simulated 3D viewpoint using only synthetic training data. The aim is to develop a method for identifying changes, such as added or removed walls and pillars, that is lightweight, requiring minimal computational resources and model complexity, and efficient in the sense that it enables fast, per-scan inference without relying on heavy pre-processing or shape completion. The changes are identified between a digital twin scan (DT scan) that represents the BIM-state, and a simulated real-world scan (robot scan) that represents the real-world state, which can ultimately support real-time updates of BIM-based digital twins.

The main research question therefore is:

How can structural mismatches in indoor environments be detected using per-point predictions from 3D scan pairs generated from a single simulated viewpoint?

Although the end goal is object-level mismatch detection, per-point predictions serve as the foundation of the approach, enabling downstream clustering and object reconstruction.

To answer this, the following sub-questions are explored:

- How can a fully synthetic data pipeline be designed to generate high-quality, labeled 3D scan pairs for training mismatch detection models?
- Can simple, lightweight neural network models trained on synthetic data generalize to unseen scenes and identify different types of structural changes?
- How can mismatch predictions be validated and model outputs selected when mismatch types are unknown?
- How can we create a bridge from mismatch detection to the updating of a BIM-based digital twin?

These questions are addressed by developing a pipeline with scenario-specific models, training them on generated synthetic scan pairs, and applying them to a wide range of test scenes, both synthetic and semi-real, to explore the limits of single-viewpoint, scan-vs-scan mismatch detection.

C. Outline

The remainder of this thesis is organized as follows. Section II presents the full methodology used in this project, covering the synthetic data generation pipeline, preprocessing steps, model architecture, and evaluation procedure. This section also introduces the four specialized neural networks trained to detect specific mismatch types: pillar-added, pillar-removed, wall-added, and wall-removed. Section III reports the experimental results, including individual model performance, ablation studies, full pipeline evaluation, generalization to real digital twin scenes and multi-mismatch settings. Section IV discusses the main findings, limitations of the current approach, and opportunities for future work. Finally, Section V concludes the report by summarizing the key contributions and outcomes. The appendix provides supporting visual and quantitative material. Appendix A illustrates the different mismatch scenarios used in this study. Appendix B contains the training curves for all four specialized models. Appendix C provides the test visualizations and evaluation metrics. Appendix D contains bounding box fitting results used in model selection. Appendix E details the semi-real-world validation using an IFC model. Appendix F provides additional examples from multi-mismatch scenario evaluations.

II. Methods

This chapter describes the full methodology used to design, train, and evaluate the proposed mismatch detection pipeline. The approach is entirely data-driven and begins with the generation of synthetic indoor environments, followed by 3D scan simulation, data preprocessing, neural network training, and scenario-specific evaluation. All components are designed to be modular and repeatable, allowing the pipeline to be extended to new mismatch types or sensor conditions.

A. Data Generation

This project relies entirely on synthetic data generated in simulation to support the training and evaluation of a mismatch detection neural network. The use of simulation allows for precise manipulation of indoor environments, reproducible sensor conditions, and the introduction of ground-truth mismatches. This section describes the tools, simulation flow, and data formats used in the dataset generation process.

1) Room Generation in Blender: Randomized indoor scenes are generated in Blender 3.6 using a parametric dataset generator developed in prior work by Van Den Akker [10]. Each scene includes structural elements, such as walls, pillars, doors, ceilings, and floors, as well as optional non-structural objects like chairs and tables. The generator uses geometry nodes and Python scripting to vary room layouts, object placements, and dimensions across samples. These synthetic scenes serve as digital twins, providing clean reference environments with known geometry and semantic labels for training and evaluation. Examples of generated rooms can be seen in Appendix A. In every room, the 3D sensor is randomly placed within the room to simulate a variety of viewpoints. This ensures that the model is able to learn object features that are robust to changes in sensor position, rather than overfitting to specific spatial configurations.

2) 3D Sensor Simulation using BLAINDER: To simulate realistic 3D scans, the generated scenes are imported into Blender 4.2 and scanned using the BLAINDER add-on [8]. A virtual RGB-D camera, modelled after the OAK-D-PRO from the TurtleBot 4, is positioned at a height of 1 meter, similar to mobile manipulation platforms such as in the work of Wettach et al, [32] or Pepper [33]. This height improves structural coverage given the limited vertical field of view (VFOV) of such sensors.

BLAINDER produces both clean scans and occluded scans (scans with non-structural clutter like chairs or tables). It outputs point clouds with geometry and semantic labels in the form of a categoryID, which is a float value that identifies object instances. This enables instance-level grouping without requiring a separate segmentation model, though it does limit real-world applicability, where such labels are typically not directly available. An example of a labeled scan is shown in Figure 1.

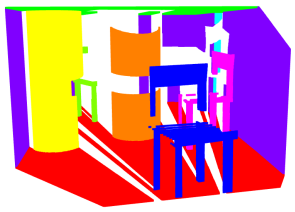


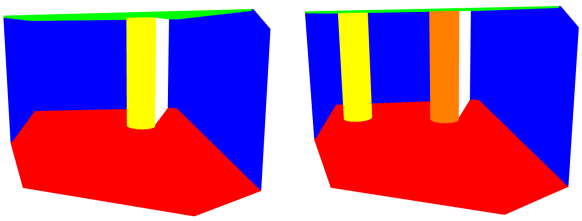
Figure 1: Example of instance segmented point cloud.

3) Mismatches and Scenario Control: Each scene is first saved in its original form as a DT scan, representing a digital twin-aligned point cloud with only structural elements. Mismatches are then introduced, such as the addition or removal of walls or pillars, and optional disturbances (e.g., chairs, tables) are added to create the robot scan, simulating a real-world perception.

In real-world indoor settings, movable objects like chairs and tables frequently obstruct the sensor's line of sight. These are therefore used as occlusion sources in the robot scans to reflect realistic visual obstacles, as commonly encountered in indoor environments, see Appendix E for an example.

This project uses a registration-free setup by leveraging pre-aligned synthetic scenes, which eliminates the need for scan alignment and allows precise control over mismatch scenarios. However, the assumption of perfect alignment between the DT scan and robot scan is a limitation when moving to real-world applications. In practice, a robot's current location may not be precisely known, making exact pose replication difficult or impossible. This introduces inevitable misalignment between scans, which must be addressed through robust registration techniques or models that can tolerate small spatial inconsistencies.

During training, each scene contains only a single type of mismatch (e.g., pillar removed, wall added) to isolate feature learning and simplify interpretation. In later experiments, multiple mismatches are introduced to evaluate a model's generalization. An example raw scan pair for the pillar added scenario is shown in Figure 2 and an overview of scenario examples is given in Appendix A.



(a) DT scan (b) robot scan

Figure 2: Scan pair example for the *pillar added* scenario.

4) Export Format and Data Structure: BLAINDER allows for exporting point clouds as .csv files, alongside other formats, where each row corresponds to a point and includes spatial coordinates (X, Y, Z) and a categoryID for semantic labeling. Other metadata fields (e.g., partID, distance to sensor, intensity, RGB values) are exported but not used in this project.

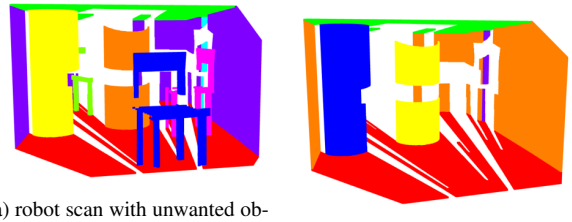
Sensor noise simulation was intentionally disabled to maintain full control over scene geometry and labels during development. This allowed for easier debugging and consistent evaluation. Once the core pipeline is validated, noise can be added to assess the model's robustness under more realistic sensor conditions.

In a later step, each point is annotated with a binary mismatch label, and a nearest-neighbour distance feature is computed by measuring the distance from each robot scan point to the closest point in the DT scan. This feature captures local geometric deviations between the scans and provides the model with a signal for identifying structural mismatches.

B. Data Preprocessing

Before training, each synthetic scene is passed through a structured preprocessing pipeline to improve data consistency and prepare the point clouds for feature extraction and model input. Raw scans may contain non-structural clutter and variable point densities, which can interfere with mismatch detection. The preprocessing includes five main steps: object filtering, downsampling, mismatch label generation, data augmentation, and formatting. These steps ensure that the network receives clean, well-annotated input and learns to detect structural mismatches reliably. While the overall pipeline is consistent across scenarios, some logic (e.g., label generation) is tailored to specific mismatch types.

1) Object Filtering: In indoor scans, movable objects, such as chairs and tables, may appear alongside structural elements like walls and pillars. These non-structural points can introduce noise and lead to false positives during training. To address this, a filtering step removes irrelevant points from the robot scan based on two criteria: semantic class (categoryID) and height. Structural elements typically span a significant height, while the considered clutter in this work remains near the floor. Objects are retained only if their height exceeds a threshold (e.g. 2 meters), with exceptions for floor and ceiling regions. This approach assumes access to semantic labels, which in this project are provided by BLAINDER but may not be directly available in real-world scans, thus requiring an extra prior segmentation step. Filtered scans are used for subsequent preprocessing steps. An example is shown in Figure 3.

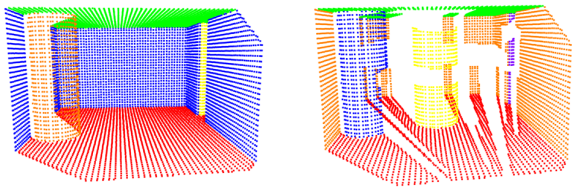


(a) robot scan with unwanted objects

(b) Object filtered robot scan

Figure 3: Example of object filtering. All non-structural objects are filtered out.

2) Downsampling: To ensure uniform point density and reduce memory usage during training, all DT scan and robot scan point clouds are voxel-downsampled using a resolution of 0.1 meters. This step also reduces noise and redundancy, helping the model generalize better. An example of a downsampled scan pair is shown in Figure 4. Statistical outlier removal is supported but disabled by default, as sensor noise is not included in this stage and structural detail is prioritized.



(a) Downsampled DT scan (no occlusions or mismatched objects)

(b) Downsampled robot scan (filtered out occlusions and added pillar)

Figure 4: Example of downsampled pair.

3) Label Generation & Filtering: Accurate per-point labels are essential for supervised mismatch detection. Each point in a DT scan or robot scan is labeled as a mismatch (1) or not (0) by comparing it to the opposite scan using nearest-neighbour distance. A KD-tree [34] is used to compute distances, and points exceeding a defined threshold are labeled as mismatched. In a later stage, during training, a model learns a smooth, probabilistic mapping from distance to mismatch likelihood, based on patterns seen across many training scenes. This soft interpretation helps reduce sensitivity to noise and occlusion artefacts, and enables more robust downstream processing, such as clustering and bounding box fitting, compared to a hard decision rule.

The labeling logic depends on the considered scenario. In object removal cases, mismatches are labeled in the DT scan since these indicate objects present in the DT scan but absent from the robot scan (object was removed from the real world). In object addition cases, mismatches are labeled in the robot scan since these points indicate objects present in the robot scan but absent from the DT scan (object was added to the scene). In unchanged cases, all points are labeled 0. An example of label generation for the pillar-added scenario is shown in Figure 5.

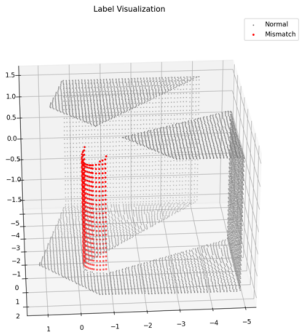
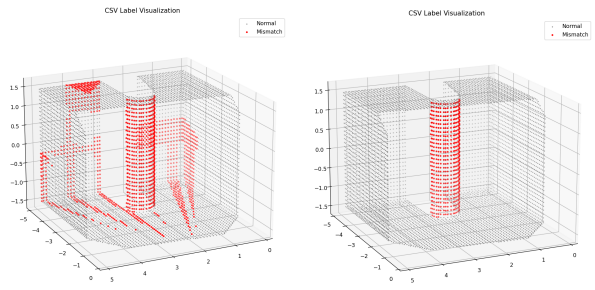


Figure 5: Example of label generation for the *pillar added* scenario. The red points are marked as mismatched in the robot scan.

The label generation process for removed objects can be affected by occlusion shadows, especially when objects in the robot scan block the sensor's view. This often results in false mismatch labels appearing on nearby surfaces such as ceilings, walls, or floors. This occurs because removed-object labeling is performed by comparing each point in the DT scan to its nearest neighbour in the robot scan. If the distance exceeds a certain threshold, the DT scan point is labeled as a mismatch. However, the DT scan contains no movable objects like chairs or tables, while the robot scan may include them. These objects can cast occlusion shadows, leaving empty regions in the robot scan. As a result, DT scan points that fall within these shadowed regions find no close neighbour and are mistakenly labeled as mismatches. This issue does not occur in added-object scenarios, where mismatch labels are assigned to robot scan points instead.

To reduce false positives caused by occlusion shadows, a clustering-based label filtering step is applied. This removes small, misaligned clusters that may result from missing points behind occluding furniture. Clusters are retained only if they stay within a local spatial radius, belong to the same categoryID and show vertical alignment (as all structural objects are vertical). It must be noted that the categoryID value is used here for label filtering and is not used during training or testing. The filtered label files serve as the final ground

truth for training. These high-quality labels greatly reduce false positives during inference and allow the model to learn robust mismatch detection patterns. See Figure 6 for a visual comparison.



(a) Original labels (b) Filtered labels

Figure 6: Example of label filtering. Note that the shadows are gone in this example and we are left with only the mismatched object.

To facilitate focused learning and evaluation, four specialized models are trained independently: one for each combination of object type and mismatch type. These include the pillar-added, wall-added, pillar-removed, and wall-removed models. Each model is trained on a dataset containing only that specific mismatch type, enabling the network to learn highly scenario-specific geometric patterns while avoiding confusion from mixed-label training.

4) Data Augmentation: To improve generalization and prevent overfitting to absolute coordinates, spatial augmentations are applied to each DT scan and robot scan pair. Since the model focuses on relative geometry rather than global position, it must learn to be invariant to transformations.

For each scene, three augmented versions are created by applying a rotation around a random axis, translation using a random 3D offset and combined rotation and translation. Transformations are applied consistently to both scans in a pair to maintain alignment of mismatch labels.

5) Data Formatting: To enable fast loading during training, all labeled .csv files are converted into NumPy binary (.npy) format. Each file is split into a point cloud file with coordinates (X, Y, Z) and a label file with binary mismatch values. This structure allows point clouds and labels to be loaded efficiently in parallel during training. This step finalizes the preprocessing pipeline, producing clean, structured data ready for use.

C. Mismatch Detection Neural Network Architecture

The proposed architecture draws inspiration from PointNet [20] in its use of shared 1D convolutions, global max pooling, and concatenation of global and local features. However, it omits components like spatial transformer networks and uses a simpler input feature set tailored to geometric mismatch detection. It performs per-point classification on either the DT scan or robot scan, with the opposing scan used only to compute an additional nearest-neighbour distance feature. An overview of the architecture is shown in Figure 7.

This architecture was selected over alternatives like Siamese networks or KPConv for reasons of simplicity, interpretability, and computational efficiency. Siamese models are typically designed for symmetric input pairs where both inputs are processed equally through shared weights to assess similarity or difference. While our task also involves comparing a DT scan to a robot scan, the comparison is encoded more indirectly: only one scan together with computed labels is used as input to the model, while the other is used solely to compute a nearest-neighbour distance feature. This design simplifies the architecture while still capturing geometric differences between the two scans. KPConv is known to require more training data and computational resources due to its kernel point convolution operations [25]. It has shown strong performance in high-resolution semantic segmentation tasks, where precise point-level classification is needed across dense 3D scenes. Given the simpler binary classification task and limited training data in this project, a PointNet-style 1D-CNN was considered a more efficient and appropriate choice than more complex architectures.

1) Input Representation: Each point in the preprocessed cloud is represented as a feature vector:

$$\mathbf{X}_i = (x_i, y_i, z_i, f_i) \tag{1}$$

where:

- x_i, y_i, z_i denote the global coordinates of the point.
- f_i represents a nearest neighbour distance feature.

Each point cloud is paired with a label file containing binary mismatch annotations. To improve robustness and false positives, the training set also includes unchanged scenes, rooms where the DT scan and robot scan are identical. These scenes help the network learn to recognize normal geometry and avoid overfitting

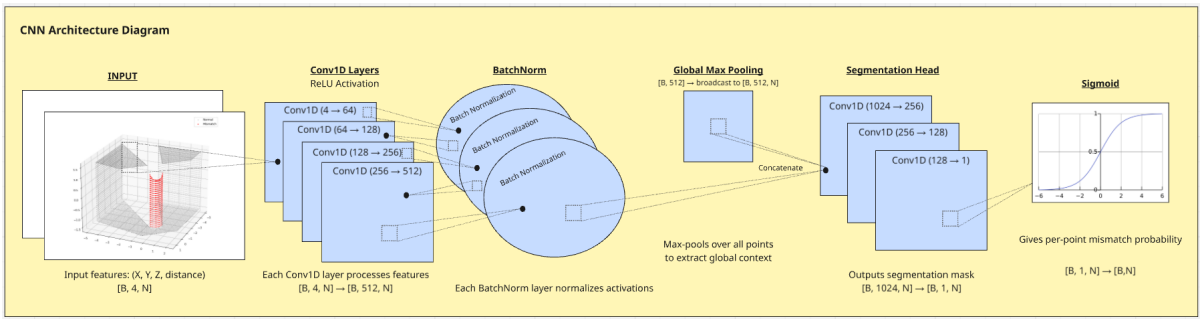


Figure 7: 1D-CNN architecture used for per-point mismatch detection between a digital twin scan and a robot LiDAR scan. The network applies stacked Conv1D layers and batch normalization, aggregates global features via max pooling, and outputs per-point mismatch probabilities through a segmentation head and sigmoid activation.

to the presence of structural changes. Including such negative examples is crucial for realistic deployment, where many scans may contain no mismatches at all.

The input shape for a batch of point clouds is:

$$(\text{batch\_size}, \text{num\_points}, \text{num\_features})$$

2) 1D Convolutional Feature Extraction: The network applies four sequential 1D convolutional layers with increasing channel dimensions, each followed by ReLU activation and batch normalization. After feature extraction, a global max pooling operation aggregates scene-level information. The global feature is replicated across all points in the scan, allowing it to be concatenated with the local features at each point, resulting in a combined feature tensor of shape $(B, 1024, N)$.

3) Global Feature Pooling: A global max pooling layer is applied to summarize the most dominant features across all points:

$$\mathbf{g} = \max_{i=1}^{N} \mathbf{f}_i \tag{2}$$

where $\mathbf{g}$ is the global feature vector, and N is the number of points in the input scan.

4) Segmentation Head: Following the pooled features, a segmentation head reconstructs per-point classifications through three additional Conv1D layers. A Sigmoid activation function is applied to the final output to produce per-point mismatch probabilities between 0 and 1.

5) Loss Function: The model is trained using a binary Focal Loss function [35] to address class imbalance, defined as:

$$L = -\alpha(1-\hat{y}_i)^{\gamma} \cdot y_i \log(\hat{y}_i) - (1-\alpha)\hat{y}_i^{\gamma} \cdot (1-y_i) \log(1-\hat{y}_i) \tag{3}$$

where $\hat{y}_i$ is the predicted probability, y_i is the ground truth label, and the hyperparameters are set to $\alpha = 10.0$, $\gamma = 2.0$. The weighting factor α increases the penalty for false negatives, helping the model focus on the minority class that is mismatched points. The focusing parameter γ reduces the influence of well-classified examples, allowing the model to better learn from harder or ambiguous cases. This is especially important for detecting subtle geometric changes in noisy or cluttered point clouds.

6) Specialized Models per Scenario: To address the distinct characteristics of different mismatch types, four specialized versions of the above network architecture are trained independently. Each model focuses on a single scenario: pillar-added, wall-added, pillar-removed, or wall-removed. By isolating mismatch types during training, each model can learn specific geometric patterns without interference from unrelated scenarios. This division also simplifies training labels and improves interpretability during evaluation.

D. Testing

This section outlines the five-stage evaluation process used to validate the mismatch detection pipeline. The experiments are as follows:

1) Testing each specialized model on controlled single-mismatch scenarios.
2) A set of ablation studies to evaluate model robustness to occlusion, feature loss, and augmentation.
3) Selecting the best specialized mismatch detection model for test cases where mismatch type is unknown beforehand.

4) Validation on scans derived from a real BIM (IFC) model.

5) Testing each specialized model on controlled multi-mismatch scenarios.

1) Scenario-Specific Evaluation Setup: Each specialized model (e.g., pillar-removed, wall-added) is trained and evaluated independently using preprocessed 3D scan pairs. During training, the model input consists of a point cloud from either the DT scan or robot scan, depending on the mismatch type. During inference, the same scan type used for training (DT scan or robot scan) is preprocessed and fed into the trained model. The opposing scan is only used to recompute the nearest-neighbour distance feature. For addition scenarios, the robot scan is evaluated and for removal scenarios, the DT scan is used. The model outputs per-point logits, which are passed through a sigmoid activation to yield predicted mismatch probabilities $\hat{y}_i \in [0, 1]$. Model outputs are thresholded to produce binary predictions so that points can be categorized into True Positives (TP), False Positives (FP), True Negatives (TN) and False Negatives (FN). Rather than fixing this threshold, we identify the operating point where recall reaches 95%, and report the corresponding False Positive Rate ($\frac{FP}{FP+TN}$), precision ($\frac{TP}{TP+FP}$) and IoU ($\frac{TP}{TP+FP+FN}$) values, alongside the ROC AUC score. This ensures fair comparison across models under a controlled minimum-recall condition.

2) Ablation Studies: To investigate the model's robustness, several ablation tests are performed on the single-scenario models. These isolate the contribution of specific training components. The tests that are done are:

- Removing the nearest-neighbour distance as an input feature, so training the models only on coordinates.
- Removing the X,Y,Z as input features, so training the models only on nearest-neighbour distance.
- Removing occluded samples from the training set, so training the model on clean samples only.
- Test the models on a scan pair that is misaligned.
- Removing augmented samples from the training set.
- Inclusion of sensor noise.

3) Full Pipeline Model Selection: In a realistic test case, it is unknown beforehand whether a mismatch has occurred and if so, what the mismatch is. All four trained models are thus applied in parallel. For each model, per-point predictions are generated and thresholded, using a fixed threshold (e.g., 0.5). A model is selected if it produces at least one valid bounding box after mismatch detection. In test cases where a scene may contain more than one mismatch and mismatches of different types, multiple models may be selected. This ensures that all potential mismatch types, across both pillars and walls, can be detected independently, even when they occur in the same scene. PCA-based [36] bounding boxes are fitted to clusters of predicted mismatched points that result from using DBSCAN clustering [37]. Bounding box fitting is guided by the object type known from the selected model (e.g., 'pillar-removed'), which helps filter out irrelevant shapes and improves robustness to occlusions and sensor noise. Dimension bounds for boxes can be set if needed be.

Axis-Aligned Bounding Boxes (AABBs) are used for both pillar and wall bounding boxes, leveraging the Manhattan world assumption [38] that most objects in indoor environments, in our case the structural elements, align with the global axes. Bounding boxes can be configured to be vertically stretched to match the full Z-span of the point cloud, see Figure 64 and Figure 65 for example, since clusters of mismatched points may not always span the entire vertical height of an actual mismatched object. Since walls are generally thin in one dimension and the depth dimension of walls can never be seen from a single viewpoint, we fix the smallest non-vertical dimension to a modifiable constant 0.15 meters for all predicted wall bounding boxes. Duplicate boxes across models can be suppressed using center-distance filtering, ensuring that each detected mismatch is represented once. Overlapping wall bounding boxes that align can be merged so that one complete wall bounding box is obtained and not several, see Figure 66. To improve robustness against unwanted shadow artefacts, referring to the falsely labeled mismatched points due to occlusion shadows, it is also possible to trim all mismatched points near the floor and ceiling of the point cloud before fitting a bounding box or to discard clusters that have a too large percentage of points on the floor or ceiling. These post-processing steps are all configurable settings within the current implementation.

The final output of the pipeline is a structured sentence for human readability, or collection of sentences if there are multiple mismatched objects, that summarizes a detected mismatch. Internally, this output is represented as a dictionary containing four key components that are taken from the fitted bounding box metadata: the

object type (pillar or wall), the **mismatch type** (added or removed), the **bounding box center** (Blender's global coordinates in meters), and the **bounding box dimensions** (in meters). An example of such a sentence is:

> *"A pillar has been added at [-0.3 -0.4 0.] with dimensions [0.34 0.34 3.]."*

The actual updating of a BIM-based digital twin lies outside of the scope of this work. In future work, the resulting bounding boxes can be overlaid onto the prior digital twin to first check whether the bounding box is valid and not falsely produced by for example a shadow artefact. If an "object-added" box overlaps with an already existing object at that location, or if an "object-removed" box is fit around an empty space in the prior digital twin, the produced box is likely false. If a bounding box is incomplete because of the limited visibility challenge in this work, the prior building model could be leveraged to complete shapes where needed. The digital twin can also be used to identify the shape of an object. If an object is removed, the shape is already known and if an object is added, the assumption could be made that the shape is similar to surrounding objects of the same type. It must be noted here that the digital twin is thus not used as a ground truth, but rather as a prior map. The BIM-based digital twin can then be updated through the remaining fitted bounding boxes.

4) Semi-Real-World Validation: To simulate a deployment setting, a semi-real-world test is performed on a room scanned from an actual IFC model. An example of the 8th floor of the Atlas building of Eindhoven University of Technology and the IFC version of it can be seen in Appendix E. This test provides insight into the pipeline's readiness for real-world use, where the mismatch type is unknown and all mismatch detection models must be considered.

5) Multi-Mismatch Scenario Evaluation: In this last testing stage, the framework is tested on scenes that contain multiple mismatches. The four different scenarios are now transformed into ingredients that can be used to create different multi-mismatch cases. These more complex test cases include:

- Multiple objects of the same type and mismatch (e.g., several removed pillars)
- Multiple objects of different types but the same mismatch (e.g., walls and pillars added)
- Multiple mismatch types across the same object class (e.g., one pillar removed, another added)
- Fully mixed scenes with several types and mismatch categories

These cases test the pipeline's ability to scale, avoid redundant detections, and correctly localize all mismatches.

III. Results

This chapter presents the results of the trained mismatch detection models on different test cases spread across five stages. First the evaluation results of the four specialized mismatch detection models: pillar-added, wall-added, pillar-removed, and wall-removed are presented. Hereafter follow some ablation studies, a section for the model selection logic, semi-real-world validation and lastly the multi-mismatch handling. Key results are summarized quantitatively in and visually in Appendix C. Detailed interpretation is discussed in the Discussion chapter.

A. Pillar-Added Model

This model was trained to detect newly added pillars in robot scans. The training curves, shown in Appendix B1, indicate smooth convergence and the learning of a generalizable pattern, since the training and validation loss steadily decrease and reach a plateau at a low value without divergence, while the validation precision and recall stabilize at high levels early in training. Training loss reflects how well the model fits the training data, while validation loss measures performance on unseen data. Validation precision indicates the proportion of predicted mismatches that are correct (low false positives), and validation recall reflects how many actual mismatches are detected (low false negatives). The lack of overfitting suggests that the model generalizes well to unseen data. Score metrics and visualizations in Appendix C2 confirm that the model detects added pillars with high precision and spatial accuracy, even under occlusion. The heatmaps show strong activation only around the added pillar, while the classification maps indicate minimal false positives and false negatives. Predicted probability histograms are clearly bimodal, reflecting confident decision boundaries. Both clean and occluded test cases show consistent performance, demonstrating the model's robustness to viewpoint constraints and occluding structures.

B. Wall-Added Model

This model was trained to detect newly added walls in robot scans. The training curves in Appendix B2 show steady convergence and good generalization, with both training and validation loss stabilizing at low values.

Validation precision and recall remain high throughout, indicating strong model confidence. Visualizations in Appendix C3 confirm that the model localizes added walls with high spatial precision and very few false positives. The predicted probabilities show a clearly bimodal distribution, suggesting confident separation between changed and unchanged regions. Performance remains consistent even in cluttered scenes with occlusion, demonstrating that the model has learned to robustly identify added structural elements under realistic conditions.

C. Pillar-Removed Model

This model was trained to detect removed pillars in DT scans. The training curves in Appendix B3 show stable learning and strong generalization. Visualizations in Appendix C4 confirm that the model consistently localizes the removed pillar, even under occlusion. In the clean test case, predictions are highly accurate with minimal errors. In the occluded case, while the removed object is still correctly identified, the model produces a number of false positives, particularly on surfaces affected by occlusion shadows. This leads to a notable drop in precision and IoU. These errors do not stem from the labeling process before training, which has been refined to filter out shadow artefacts, but rather from the nearest-neighbour distance feature used during training. This is the first covered test case where such shadow-induced false positives visibly degrade performance.

D. Wall-Removed Model

This model was trained to detect removed walls using DT scans. The training curves in Appendix B4 show steady convergence and strong learning, with high validation precision and recall across most epochs. Visualizations in Appendix C5 confirm accurate detection of the removed wall in clean scenes, with minimal classification error. In the occluded test case, however, the model shows a marked drop in performance: the ROC AUC drops significantly compared to other scenarios, and both precision and IoU decline similarly to the occluded pillar-removed case. This is again caused by false positives arising from shadowed regions, where the nearest-neighbour distance feature misleads the model into labeling background points as mismatches. It must be noted that the removed wall is successfully detected completely.

E. Ablation Study

1) Removal of NN-Distance Feature: This ablation retrains the pillar-removed model using only (X, Y, Z) coordinates to assess the importance of the nearest-neighbour distance feature and its role in shadow-related errors. Although training loss converged smoothly, validation precision and recall remained near zero, resulting in the model failing to detect the removed pillar. These results confirm that NN-distance is essential for mismatch detection, as it provides the only signal encoding the geometric discrepancy between the scans. Without it, the model is unable to distinguish between unchanged and mismatched areas. These results reinforce that coordinates alone are insufficient for mismatch detection.

2) Removal of XYZ Features: This ablation removes the (X, Y, Z) coordinates, training the pillar-removed model using only the nearest-neighbour distance feature. While clean case performance matched the original model, visualizations showed reduced confidence near the top and bottom of the removed pillar, since these are closer to unchanged points between the scans. In the occluded case, performance dropped notably, with lower predicted probabilities and incomplete, but sufficient detection. These results show that NN-distance alone captures some local mismatches but lacks the spatial context needed to handle occlusions robustly or maintain prediction consistency. Despite augmentation, the global coordinates remain useful for understanding object shape and scene layout.

3) Training on Clean Samples Only: This ablation study retrains the pillar-removed and wall-removed models using only clean scenes, to test whether occluded training samples, even though their labels are cleaned, contribute to shadow artefacts. Clean test performance remained unchanged, confirming that occluded samples do not harm generalization. On occluded test cases, however, the shadow problem persisted. The pillar-removed model becomes more confident but less accurate, predicting stronger outputs in both the right and wrong places with increased false negatives on the pillar. The wall-removed model showed slight improvement but struggled to localize edges under occlusion. These results suggest that training on clean data alone does not resolve shadow artefacts and reduces robustness. A mixed training set remains preferable.

4) Misaligned Scans: To simulate real-world imperfections, the DT scan was slightly translated and rotated during inference. On clean test cases, the model still correctly identifies the removed pillar, though a thin band of false positives appears near edges and shadowed regions. In occluded scenes, misalignment amplifies the shadow artefact issue, spreading mis-

match predictions across wider areas. These results show that while the model handles minor misalignment under clean conditions, occluded scenes remain highly sensitive. Robust alignment between the DT scan and robot scan is thus crucial for reliable performance in real-world use.

5) Removal of augmented samples: This ablation tests whether spatial data augmentation is needed. The pillar-removed and wall-removed models were retrained without rotated or translated samples. On clean test cases, performance matched the original models, confirming augmentation is unnecessary for ideal scenes. Surprisingly, on occluded test cases, performance improved: false positives from occlusion shadows were reduced, and mismatch predictions became more focused. This suggests that randomized scenes and sensor positions already offer sufficient variability. These new models are adopted instead of the original ones.

6) Inclusion of Sensor Noise: The inclusion of sensor noise in the form of Gaussian noise with a standard deviation of 0.1 meters proved no challenge. The only downgrade compred to noiseless sensor data was the addition of a few false positives near the edges of the mismatched object. The reason why performance does not degrade significantly is because of two things. First, one of the first things we do in our preprocessing pipeline is voxel downsampling, where out of every collection of points, one point is outputted per 10 x 10 x 10 cm block that is in the voxel grid. This already has a smoothing effect. Secondly, when we compute the nearest-neighbour between a point in the DT scan and robot scan, the nearest neighbour distance still falls below the threshold for a point to be marked as mismatched.

F. Evaluation of Full Pipeline with Model Selection

The full mismatch detection pipeline was evaluated using a model selection strategy that relies on bounding box validity. In each test case, all four specialized models, trained for one specific object-mismatch combination, were executed in parallel. Each model's per-point predictions were thresholded to extract high-probability mismatch points. A model was only selected if these points provided at least one valid bounding box, whose shape and orientation matched the expected geometry of the object type (pillar or wall).

While per-point predictions are accurate, bounding box fitting performance is naturally limited by the visibility constraints of the scene. The produced bounding boxes were geometrically reasonable in the majority of cases, see Appendix D for some examples. In all successful cases, the bounding box encloses the mismatched object from the observed viewpoint. In pillar cases, the center of the predicted bounding box is offset toward the sensor due to only the front-facing surface being visible and the cluster of mismatched points having a center closer to the sensor than the actual pillar center is. The orientation of the bounding box for square pillars is imperfect if the pillars themselves are not axis-aligned due to the Manhattan World assumption and PCA being applied only to the visible points. For walls, the depth is always assumed to be 0.15 m, but occlusions can still produce partial bounding boxes (e.g., only one side of a wall that is visible).

Some edge cases were observed in the test set of synthetic rooms that contain only one mismatched object. First is an unwanted shadow artefact getting its own bounding box. Another edge case is a square pillar that is not axis-aligned where only one of four faces is visible because the pillar is directly facing the sensor, making the pillar look like a wall. This is solved to some extent by the assumption that walls are axis-aligned, resulting in a pillar bounding box that is badly centered and oriented. If the pillar was actually axis-aligned, then this cluster of mismatched points would have gotten a wall bounding box. There is also an edge case where a pillar is obstructing a middle portion of a wall, resulting in a split wall where only one portion gets a bounding box because the other portion is too small in length to be allowed to get a bounding box. This minimum required length can be configured however.

G. Semi-Real-World Validation

During testing on the clean real-world pillar-removed scenario using IFC-based scan data, the model trained with all four features [X, Y, Z, NN-distance] failed to detect any mismatch, resulting in the mismatched pillar consisting of 100% false negatives. This occurred despite perfect labeling, correct DT scan selection, and previously high performance on synthetic data. To isolate the cause, the pillar-removed-no-xyz model was tested, which had been trained using only the nearest-neighbour distance as input. This model successfully detected the removed pillar in the real-world scan. This suggests that the inclusion of absolute spatial coordinates, whose ranges are different between synthetic and IFC scenes and always will be different between different BIM-based digital twins, harmed generalization. This behaviour is consistent with the findings from the ablation study in Section III-E2,

where the NN-distance feature was identified as the most critical input.

As a result of this finding, all other specialized models were retrained using only the NN-distance feature. These updated models are used for all remaining evaluations, including model selection and multi-mismatch scenario testing. This adjustment improves generalization to real IFC-based input without affecting clean synthetic performance. When using only the nearest-neighbour distance as input, standard augmentations like translation and rotation have no effect on the feature vector. As a result, including augmented versions of a sample leads to redundant training data, which can reduce training efficiency. Therefore, augmentation was disabled for all models trained on NN-distance-only input. Results can be seen in Appendix E.

While the results of the pillar-added and wall-added models are as promising as expected for both clean and occluded test cases, and the clean test cases for the pillar-removed and wall-removed performed well, testing on the occluded real-world pillar-removed and wall-removed scenarios revealed a generalization bottleneck. Predicted mismatched points were drawn heavily toward floor and wall regions that lie behind an occluding object in the robot scan. This is the shadow artefact problem, first mentioned in II-D3, returning. This caused many false bounding boxes to be fitted. The occluded pillar-removed test case did manage to also fit a bounding box around the actual mismatched pillar while the wall-removed model missed the actual mismatched wall because of having points with a too low predicted mismatch probability. The issue arises from the reliance on the NN-distance feature, which remains sensitive to gaps in observed geometry.

H. Multi-Mismatch Scenario Evaluation

1) Same Object and Mismatch Type: Test cases, both clean and occluded, where multiple pillars are added work as intended, even though the pillar-added model has not been trained on samples where multiple pillars were added. Walls-added cases where the added walls do not touch each other result in proper bounding boxes. Cases where the two added walls are connected result in no bounding boxes being fitted because during clustering of mismatched points the two walls become one gigantic cluster that is too wide to get even one bounding box. For clean test cases, similar behaviours as described above occurred for the pillar-removed and wall-removed models. The occluded multi-mismatch cases for these models however brought the infamous shadow artefact problem along. In the test case where multiple pillars were removed, the pillars correctly get their bounding box, but one shadow artefact in the back corner of the room also got a bounding box. In the walls removed test case, only one wall got a bounding box because the other cluster of wall points is affected by a shadow artefact that bleeds into it, resulting in a wall bounding box that is too thick to be valid for fitting. This problem and the two-connected-mismatched-walls problem can actually both be fixed by performing cluster separation based on direction before bounding box fitting, which has been implemented as an experimental configurable setting and has not been perfected. Visualizations for above cases can be seen in F1.

2) Different Object Type and Same Mismatch Type: The clean and occluded test cases where both a pillar and wall are added work precisely as intended. The clean test case where a pillar and wall are removed also works perfectly. In our occluded pillar and wall removal test case however, the wall is split into two by the occluding removed pillar that correctly gets a bounding box, but only one of two left wall portions gets a bounding box because the other is affected by a shadow artefact that bleeds into the cluster of mismatched points, rendering the cluster invalid for a wall bounding box.

Enabling the selection of multiple models brought about a new unwanted behaviour. In the "objects-removed" test case, aside from correct bounding boxes being fit inside the DT scan, the wall-added model produced high-scoring bounding boxes due to the robot scan containing unseen regions behind the removed wall. This is the infamous shadow artefact problem returning again, but now on a bigger scale. However, these predictions can be safely suppressed by verifying that these regions already exist in the digital twin. Figures for the above cases can be seen in Appendix F2.

3) Same Object and Different Mismatch Type: In both clean and occluded test cases where a pillar has been removed and another pillar has been added, the mismatched pillars correctly get a bounding box, but bounding boxes are also fit around shadow artefacts. These shadow artefacts do not only result from chairs and tables as occluding objects, like in the single-mismatch occluded cases, but also the presence and absence of one of the two pillars in one of the two scans. Put differently, the mismatched pillar itself forms a shadow artefact behind it in the other scan. In the wall-added-and-removed test case, the wall-added model succeeded in detecting and fitting a bounding

box to the added wall in the robot scan in both a clean and occluded setting. In the clean test case, the removed wall also gets a correct bounding box alongside an incorrect bounding box fitted around a shadow artefact that is formed by the added wall which is not present in the DT scan. In the occluded case, the wall-removed model fails to detect the removed wall since the predicted mismatch probabilities of corresponding points lie below the fixed threshold, resulting in no fitted bounding box. Visualizations are shown in Appendix F3.

In this type of multi-mismatch case, geometrically speaking, the removed and added object of the same object type are two separate objects, but it could be interesting to find out whether these two objects might actually be one same object that has been moved. A potential strategy for this could involve comparing predicted bounding boxes from different mismatch types. For example, if a predicted removed pillar bounding box significantly overlaps with a predicted added one elsewhere in the room, this could indicate that the same object has moved. In the context of updating a digital twin, this is desirable since we can then just move an object and keep all of its other properties instead of having to remove an object and adding in a new one. It must be noted however that this logic has not been implemented in this work.

4) Different Object Type and Mismatch Type: In this last testing stage all four scenarios are activated simultaneously. In the clean test case, the added pillar got a correct bounding box, the added wall did not because it got infected with shadow artefacts, the removed pillar and wall did get a correct bounding box and a shadow artefact got an incorrect wall-removed bounding box. In the occluded test case, the added wall and pillar got a correct box plus a shadow artefact wall-added box, while the removed wall and pillar did not get a box, since the removed pillar cluster got infected by shadow artefacts and the removed wall did not get detected by the wall-removed mismatch detection model. It was decided at this very last testing stage to trim all floor and ceiling points not after clustering of found mismatched points, but before. This solved the problem of mismatched objects being infected by shadow artefacts that were present of the floor or ceiling, like the removed pillar here. Visualizations can be seen in Appendix F4.

IV. Discussion

This work proposes a modular pipeline for mismatch detection in 3D indoor environments using simulated scans. By designing specialized neural networks for different mismatch types, it avoids the complexity of multi-label learning and focuses on binary mismatch classification at per-point level. The final pipeline integrates model selection, bounding box fitting, and mismatch reporting. While the current implementation does not perform automatic digital twin updates, it produces structured bounding box outputs that can serve as actionable inputs for updating an IFC-based building model. This updating step is considered out of scope for this project but provides a clear direction for future work.

Each of the four core models demonstrated high performance in clean test cases, with scores and bounding box fits showing that the networks reliably isolate and classify mismatches. In occluded scenarios however, added-object models (especially for pillars) are more robust to occlusions, while removed-object models suffer more from shadow artefacts, often misclassifying occluded parts of unchanged objects as removed.

To determine the potential applicability of the system beyond just synthetic environments, a semi-real-world validation stage was introduced using a real BIM-based digital twin. A point cloud was generated from the building model using simulated 3D sensor and a corresponding robot scan with mismatches was created. Results show that the mismatch detection models maintained high accuracy in most scenes, though false positives from shadow artefacts remain a challenge. This stage highlights the potential of the method to operate on digital twins of real buildings, while also revealing the limitations of the current bounding box fitting in complex structural layouts.

A limitation observed during testing is that models trained with global XYZ coordinates tended to overfit to the spatial bounds of the synthetic rooms. As a result, these models perform poorly when applied to rooms at different coordinate ranges. To address this, an alternative could be to use relative coordinates, such as positions centered around the object cluster, room centroid or the 3D sensor.

In addition to single-mismatch cases, the pipeline is also evaluated on scenes containing multiple mismatches. These include combinations of different object types and different mismatch types. The model selection logic and bounding box filtering prove effective in most cases, successfully isolating distinct mismatch regions. However, certain scenes reveal false bounding boxes around shadow artefacts or the absence of a fitted bounding box around a mismatched object due to the

object being clustered together with another object or shadow artefact. These limitations are partly a result of the per-point classification design, where context is limited to point features. As a result, bounding boxes must be reconstructed from sparse mismatch predictions without object-level reasoning. While this approach keeps the model lightweight and flexible, it also introduces challenges in scenes with clutter, occlusions, or subtle geometric variations. These findings suggest that although the pipeline scales to more complex scenes, future work could benefit from object-centric models that learn to localize and describe entire mismatched objects more directly, even when only a small portion of the object is visible, reducing reliance on bounding box fitting.

Another important limitation of the current method is that bounding boxes are only fitted to the visible portion of a mismatched object. When only the front face is captured, this can result in underestimation of the object's depth. One possible future extension is to update bounding boxes multiple times using multiple scans from different viewpoints, allowing a more complete reconstruction of the object's geometry over time.

While the mismatch detection works reliably in almost all cases, the current full pipeline chooses a model only if a bounding box can be fitted. This approach does not score or measure uncertainty between models like the mismatch detection part does. As a result, scenes with weak or ambiguous predictions may lead to no model being selected at all, or worse: a false model being selected based on bounding box fitting around a cluster of mismatched points corresponding to a shadow artefact. A more principled model selection strategy such as bounding box confidence or adding a learned selector model could improve pipeline robustness.

Several types of errors were observed. First there are the shadow effects, particularly in the "removed" scenarios. Strong occlusions cause false positives by leaving empty regions behind static objects, eventually resulting in shadow artefacts: regions of false mismatched points that take the shape of the former shadows. This problem is caused by the challenge of using only a single viewpoint, which makes it hard to detect geometric mismatches when parts of the scene are not visible. Second is bounding box overgrowth, where clusters that span multiple nearby structures can result in boxes that are too large to be plausible. One possible future extension to reduce these shadow artefacts is to apply interpolation techniques during preprocessing. By filling in small holes or sparse regions in the robot scan that result from occlusion, the mismatch labeling process could avoid falsely flagging unaffected DT scan regions as removed. While not implemented in this work, such stitching methods could provide a lightweight way to mitigate the most persistent source of false positives in this work.

The pipeline is fully synthetic. Although occlusions were simulated, the experiments did not account for real-world noise, such as surface reflectivity or specific lighting conditions. Future work must address this generalization issue through the inclusion of domain adaptation techniques such as fine-tuning on real-world scans and robust features less sensitive to density and viewpoint.

Only walls and pillars are considered as mismatched object types in this work, and only the addition or removal of such objects is implemented and tested. While this restriction simplifies the setup, it does not imply that future work must expand by adding one object type at a time. In fact, experiments show that the trained models for structurally different objects, but same mismatch type (e.g., wall-added model vs. pillar-added model) show near-identical mismatch detection behaviour when trained on the same mismatch type. This is likely due to the fact that the model ultimately relies on the nearest-neighbour distance feature during training, making the object's shape irrelevant for the mismatch detection stage. This insight suggests that the framework may generalize to additional object types, if those objects do not introduce properties that interfere with the computation or interpretation of nearest-neighbour distances. For example, reflective surfaces, or light-absorbing materials could introduce complexities not captured in this framework and would require additional model adjustments or features. However, even if mismatch detection generalizes across object types, the bounding box fitting stage remains object-type specific, since different shapes (e.g., pillars vs. walls) can possibly require different geometric assumptions.

Future work could also explore the integration of additional information that is present inside a BIM model, such as material properties and surface colour information, into the mismatch detection pipeline. These attributes are often available and could help with distinguishing between structurally similar elements. Aside from mismatched object types, future work could also deal with a wider variety of occlusion types beyond static furniture like chairs and tables, for example, dynamic objects such as humans may pose different challenges for mismatch detection.

Not all limitations identified in this work are equally impactful. Some, such as imperfect bounding box fitting or false positives caused by shadows, can likely be resolved with improved clustering or more informative features beyond the nearest-neighbour distance. Others, like relying on just one viewpoint, are more fundamental issues. However, this constraint was not accidental, since it was an intentional challenge in this project to investigate how far mismatch detection can be pushed using only a single scan, as would be the case in many real-world robotic deployments. The results show that even under this constraint, robust detection is feasible for a range of scenarios. Nevertheless, future work could explore multi-view fusion strategies that aggregate scans from different poses to improve coverage and geometric completeness. Methods such as those proposed in [39] demonstrate how registered partial scans can be integrated into a more complete scene representation. Incorporating these techniques may improve robustness and enable more reliable mismatch detection in cluttered or partially visible environments.

While the system successfully detects and describes mismatches, integration with BIM updating remains conceptual. The bounding box outputs (shape, object type, mismatch type, position, size) can serve as inputs to an updating module. Whether this update should be automatic depends on the application: in some cases, manual review may still be preferred. Automatic updates are technically feasible. Tools like IfcOpenShell allow for direct mesh and metadata editing in IFC files, while RDF-based digital twin systems enable semantic updates. However, this work focuses solely on mismatch detection, and assumes that parameters such as scan settings are configured manually. Future work could link detected bounding boxes to specific BIM elements, enabling not only geometry updates but also metadata assignment and connectivity logic—similar to the IFC-based graph integration demonstrated by Van der Meer [40]. To evaluate the impact of such updates, the updated BIM could be used in downstream applications like robot path planning, following approaches such as BIM-derived map generation for navigation demonstrated in [41] and [42] . Performing inference and fitting potential bounding boxes takes at most 0.3 seconds per model, depending on the scan size, hardware and whether a mismatch is present for that model or not. Since the full pipeline applies up to four specialized models in parallel, the total runtime is at most 1.2 seconds per scan pair if all four models are selected. This confirms that the method is computationally efficient and could be integrated into practical scan-review or BIM model maintenance workflows in future work.

V. Conclusion

This thesis explores how structural mismatches in indoor environments can be detected using per-point predictions from 3D scan pairs generated from a single simulated viewpoint. A fully synthetic data pipeline is developed to generate labeled scan pairs, allowing lightweight neural networks to be trained for specific mismatch scenarios.

The trained models perform well in clean and partially occluded scenes, successfully identifying added or removed structural elements. A semi-real-world validation using a real BIM-based digital twin confirms the pipeline's potential beyond synthetic settings. In cases with multiple mismatches, the system generally produces separate bounding boxes and interpretable descriptions of each change. Model selection is handled by checking which model produced a valid bounding box, which works reliably in most scenes. The final output of the pipeline consists of simple descriptive sentences that summarize the detected mismatch and could be used to update a BIM.

While the presented pipeline demonstrates robust performance in detecting and localizing geometric mismatches under various synthetic and semi-real-world conditions, it also has several limitations. First, the method relies on only a single viewpoint, which restricts visibility and often causes shadow artifacts, especially in object removal scenarios. Second, per-point classification without explicit object-level context can lead to fragmented or incomplete mismatch predictions. Third, the system was evaluated only on clean, synthetic structural elements (walls and pillars), excluding real-world sensor noise and more complex objects such as glass or reflective surfaces. Lastly, the mismatch detection models operate independently, and the final model selection logic may fail in ambiguous cases.

Future research could address these limitations by incorporating multi-view fusion strategies, refining bounding box logic, and expanding to more diverse object types and materials. Improvements to model selection, such as a learned selector module or uncertainty scoring, could increase robustness in complex scenes. While this work focuses solely on mismatch detection, a natural next step is the integration of bounding box outputs into the updating of an actual BIM-based digital twin.

In summary, this thesis demonstrates that reliable mismatch detection is possible using per-point predictions from simulated 3D scans that are taken from a single viewpoint. The proposed mismatch detection pipeline is particularly useful in scenarios where a BIM-based digital twin must be kept up to date with real-world structural changes, like facility management, construction progress monitoring or renovation planning. By automating the detection of additions and removals of structural elements, the devised framework reduces the need for manual inspection and supports more frequent, low-cost updates to the digital twin. While there are still challenges in generalization, shadow artefacts, and full BIM model integration, the method offers a strong foundation for future work in the updating of BIM-based digital twins.

REFERENCES

[1] R. Sacks, G. Lee, L. Burdi and M. Bolpagni, *BIM Handbook: A Guide to Building Information Modeling for Owners, Designers, Engineers, Contractors, and Facility Managers, Fourth Edition*, 4th. Wiley, Jan. 2025, pp. 1–525, ISBN: 9781394222223. DOI: 10.1002/9781394222254.

[2] O. Odugu, F. Ghafari, E. Shourangiz, M. T. Khan and C. Wang, 'Building Information Model (BIM) and Robotic Systems Integration for Construction: A Comprehensive Workflow Analysis and Future Perspectives,' *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 14736 LNAI, pp. 272–282, 2024, ISSN: 16113349. DOI: 10.1007/978-3-031-60615-1{_}18/TABLES/2.

[3] E. Noroozinejad Farsangi, A. O. Shehata, M. Rashidi, N. Ghassempour and S. Mirjalili, *Transitioning from BIM to Digital Twin to Metaverse*, 2024. DOI: 10.3389/fbuil.2024.1486423.

[4] R. Volk, J. Stengel and F. Schultmann, 'Building Information Modeling (BIM) for existing buildings — Literature review and future needs,' *Automation in Construction*, vol. 38, pp. 109–127, Mar. 2014, ISSN: 0926-5805. DOI: 10.1016/J.AUTCON.2013.10.023.

[5] P. Tang, D. Huber, B. Akinci, R. Lipman and A. Lytle, 'Automatic reconstruction of as-built building information models from laser-scanned point clouds: A review of related techniques,' *Automation in Construction*, vol. 19, no. 7, pp. 829–843, Nov. 2010, ISSN: 0926-5805. DOI: 10.1016/J.AUTCON.2010.06.007.

[6] T. Kang, S. Patil, K. Kang, D. Koo and J. Kim, 'Rule-Based Scan-to-BIM Mapping Pipeline in the Plumbing System,' *Applied Sciences 2020, Vol. 10, Page 7422*, vol. 10, no. 21, p. 7422, Oct. 2020, ISSN: 2076-3417. DOI: 10.3390/APP10217422.

[7] G. P. Vassena, L. Perfetti, S. Comai, S. Mastrolembo Ventura and A. L. Ciribini, 'Construction Progress Monitoring through the Integration of 4D BIM and SLAM-Based Mapping Devices,' *Buildings 2023, Vol. 13, Page 2488*, vol. 13, no. 10, p. 2488, Sep. 2023, ISSN: 2075-5309. DOI: 10.3390/BUILDINGS13102488.

[8] S. Reitmann, L. Neumann and B. Jung, 'BLAINDER—A Blender AI Add-On for Generation of Semantically Labeled Depth-Sensing Data,' *Sensors 2021, Vol. 21, Page 2144*, vol. 21,

no. 6, p. 2144, Mar. 2021, ISSN: 1424-8220. DOI: 10.3390/S21062144.

[9] S. Fedorova, A. Tono, M. S. Nigam *et al.*, 'Synthetic 3D Data Generation Pipeline for Geometric Deep Learning in Architecture,' *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, vol. 43, no. B2-2021, pp. 337–344, Apr. 2021. DOI: 10.5194/isprs-archives-XLIII-B2-2021-337-2021.

[10] P. J. Van Den Akker, *A Parametric Dataset Generator For Updating Digital Twins Through 2D LiDAR Perceptual Data. Master Thesis. Eindhoven University of Technology.* 2024.

[11] N. Duc, Y.-L. Lai, P. Madlindl *et al.*, 'Mind the Domain Gap: Measuring the Domain Gap Between Real-World and Synthetic Point Clouds for Automated Driving Development,' May 2025. DOI: 10.48550/arXiv.2505.17959.

[12] Y. Zhao, N. Zhao and G. H. Lee, 'Synthetic-to-Real Domain Generalized Semantic Segmentation for 3D Indoor Point Clouds,' Dec. 2022. DOI: 10.48550/arXiv.2212.04668.

[13] J. Tremblay, A. Prakash, D. Acuna *et al.*, 'Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization,' *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, vol. 2018-June, pp. 1082–1090, Apr. 2018, ISSN: 21607516. DOI: 10.1109/CVPRW.2018.00143.

[14] M. Buda, A. Maki and M. A. Mazurowski, 'A systematic study of the class imbalance problem in convolutional neural networks,' *Neural Networks*, vol. 106, pp. 249–259, Oct. 2018, ISSN: 0893-6080. DOI: 10.1016/J.NEUNET.2018.07.011.

[15] I. Salhi, M. Poreba, E. Piriou, Gouet-BrunetValerie and M. Ojail, 'Multimodal localization for embedded systems: A survey,' *Multimodal Scene Understanding: Algorithms, Applications and Deep Learning*, pp. 199–278, Jan. 2019. DOI: 10.1016/B978-0-12-817358-9.00014-7.

[16] J. Mallalieu, J. L. Carrivick, D. J. Quincey, M. W. Smith and W. H. James, 'An integrated Structure-from-Motion and time-lapse technique for quantifying ice-margin dynamics,' *Journal of Glaciology*, vol. 63, no. 242, pp. 937–949, Dec. 2017, ISSN: 00221430. DOI: 10.1017/JOG.2017.48.

[17] Z. J. Yew and G. H. Lee, 'City-scale Scene Change Detection using Point Clouds,' *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2021-May, pp. 9579–9585, Mar. 2021, ISSN: 10504729. DOI: 10.1109/ICRA48506.2021.9561855.

[18] W. Zhang, Y. Zhang and J. Li, 'A Two-Stage Correspondence-Free Algorithm for Partially Overlapping Point Cloud Registration,' *Sensors (Basel, Switzerland)*, vol. 22, no. 13, p. 5023, Jul. 2022, ISSN: 14248220. DOI: 10.3390/S22135023.

[19] A. Kharroubi, F. Poux, Z. Ballouch, R. Hajji and R. Billen, 'Three Dimensional Change Detection Using Point Clouds: A Review,' *Geomatics 2022, Vol. 2, Pages 457-485*, vol. 2, no. 4, pp. 457–485, Oct. 2022, ISSN: 2673-7418. DOI: 10.3390/GEOMATICS2040025.

[20] C. R. Qi, H. Su, K. Mo and L. J. Guibas, 'PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation,' *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, pp. 77–85, Dec. 2016. DOI: 10.1109/CVPR.2017.16.

[21] C. R. Q. Li, Y. Hao, S. Leonidas and J. Guibas, 'PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space,' in *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017.

[22] M. Jonek, *Enhancing PointNet: New Aggregation Functions and Contextual Normalization. Thesis.* 2024. [Online]. Available: https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-349362.

[23] K. T. Wijaya, D. H. Paek and S. H. Kong, 'Advanced Feature Learning on Point Clouds Using Multi-Resolution Features and Learnable Pooling,' *Remote Sensing 2024, Vol. 16, Page 1835*, vol. 16, no. 11, p. 1835, May 2024, ISSN: 2072-4292. DOI: 10.3390/RS16111835. [Online]. Available: https://www.mdpi.com/2072-4292/16/11/1835/htm%20https://www.mdpi.com/2072-4292/16/11/1835.

[24] G. Koch, R. Zemel and R. Salakhutdinov, 'Siamese Neural Networks for One-shot Image Recognition,' in *ICML deep learning workshop*, vol. 2, 2015, pp. 1–30.

[25] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette and L. J. Guibas, 'KPConv: Flexible and Deformable Convolution for Point Clouds,' in *Proceedings of the IEEE/CVF*

International Conference on Computer Vision (ICCV), Oct. 2019, pp. 6411–6420.

[26] I. De Gelis, T. Corpetti and S. Lefevre, 'Change Detection Needs Change Information: Improving Deep 3-D Point Cloud Change Detection,' *IEEE Transactions on Geoscience and Remote Sensing*, vol. 62, pp. 1–10, 2024, ISSN: 15580644. DOI: 10.1109/TGRS.2024.3359484.

[27] Y. Liu, B. Fan, G. Meng, J. Lu, S. Xiang and C. Pan, 'DensePoint: Learning densely contextual representation for efficient point cloud processing,' *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2019-October, pp. 5238–5247, Oct. 2019, ISSN: 15505499. DOI: 10.1109/ICCV.2019.00534.

[28] Y. Sun, X. Zhang and Y. Miao, 'A review of point cloud segmentation for understanding 3D indoor scenes,' *Visual Intelligence*, vol. 2, no. 1, pp. 1–13, Jun. 2024, ISSN: 27319008. DOI: 10.1007/S44267-024-00046-X/TABLES/4.

[29] Z. Zheng, L. Mentzer, B. Iskender, M. Price, C. Prendergast and A. Cloitre, 'Semantic Segmentation and Scene Reconstruction of RGB-D Image Frames: An End-to-End Modular Pipeline for Robotic Applications,' Oct. 2024. [Online]. Available: https://arxiv.org/pdf/2410.17988v2.

[30] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys and A. Geiger, 'Convolutional Occupancy Networks,' Mar. 2020. DOI: 10.48550/arXiv.2003.04618.

[31] C. J. Si, Z. B. Yin, Z. Q. Fan *et al.*, 'Point cloud completion network for 3D shapes with morphologically diverse structures,' *Complex and Intelligent Systems*, vol. 10, no. 3, pp. 3389–3409, Jun. 2024, ISSN: 21986053. DOI: 10.1007/S40747-023-01325-8/FIGURES/12.

[32] J. Wettach and K. Berns, 'Experimental evaluation of some indoor exploration strategies,' in *ICINCO 2016 - Proceedings of the 13th International Conference on Informatics in Control, Automation and Robotics*, vol. 2, SciTePress, 2016, pp. 280–287, ISBN: 9789897581984. DOI: 10.5220/0005977502800287.

[33] T. Alhmiedat, A. M. Marei, W. Messoudi *et al.*, 'A SLAM-Based Localization and Navigation System for Social Robots: The Pepper Robot Case,' *Machines 2023, Vol. 11, Page 158*, vol. 11, no. 2, p. 158, Jan. 2023, ISSN: 2075-1702. DOI: 10.3390/MACHINES11020158. [Online]. Available: https://www.mdpi.com/2075-1702/11/2/158/htm%20https://www.mdpi.com/2075-1702/11/2/158.

[34] J. L. Bentley, 'Multidimensional binary search trees used for associative searching,' *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975, ISSN: 15577317. DOI: 10.1145/361002.361007.

[35] T. Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollar, 'Focal Loss for Dense Object Detection,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 318–327, Aug. 2017, ISSN: 19393539. DOI: 10.1109/TPAMI.2018.2858826.

[36] I. T. Jollife and J. Cadima, 'Principal component analysis: a review and recent developments,' *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, no. 2065, Apr. 2016, ISSN: 1364503X. DOI: 10.1098/RSTA.2015.0202.

[37] Ester M, Kriegel H.P, Sander J and Xu X, 'A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise,' *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, 1996.

[38] H. Guo, S. Peng, H. Lin *et al.*, *Neural 3D Scene Reconstruction With the Manhattan-World Assumption*, 2022. [Online]. Available: https://zju3dv.github.io/manhattan_sdf..

[39] A. Dai, D. Ritchie, M. Bokeloh, S. Reed, J. Sturm and M. Nießner, 'ScanComplete: Large-Scale Scene Completion and Semantic Segmentation for 3D Scans,' in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4578–4587, ISBN: 1480×1230×64. DOI: 10.1109/CVPR.2018.00481.

[40] J. Van Der Meer, *The History of The walls - Update of indoor IFC models using point cloud geometry. Master Thesis. Eindhoven University of Technology.* 2024.

[41] S. Karimi, R. G. Braga, I. Iordanova and D. St-Onge, 'Semantic Navigation Using Building Information on Construction Sites,' Apr. 2021. DOI: 10.48550/arXiv.2104.10296.

[42] Z. Chen, K. Chen and J. C. Cheng, 'A BIM-integrated Indoor Path Planning Framework for Unmanned Ground Robot,' *Proceedings of the International Symposium on Automation and Robotics in Construction*, vol. 2021-November, pp. 776–783, 2021, ISSN: 24135844. DOI: 10.22260/ISARC2021/0105.

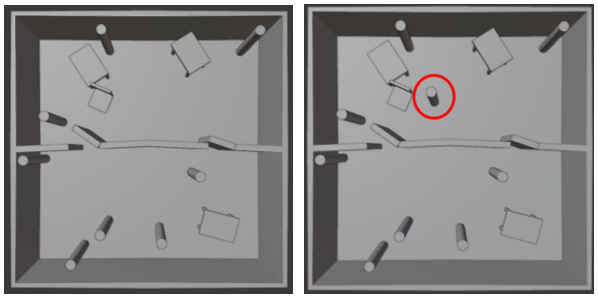
APPENDIX

A. Scenarios

← *Click here to go back to Data Generation*

← *Click here to go back to Mismatches and Scenario Control*

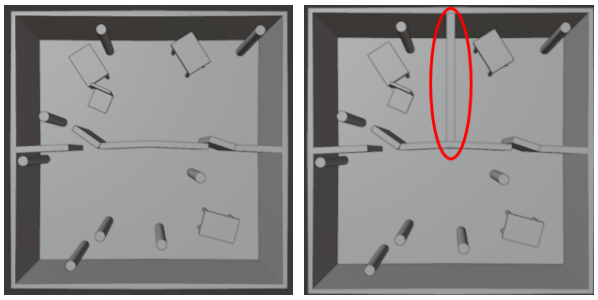
1) Pillar Added: See example of a generated room with an added pillar below.



(a) Original room (b) Modified room

Figure 8: Example of pillar-added scenario

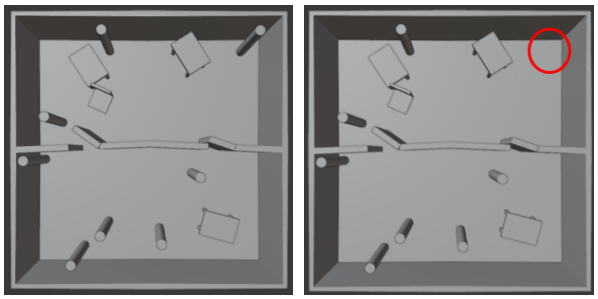
2) Wall Added: See example of a generated room with an added wall below.



(a) Original room (b) Modified room

Figure 9: Example of wall-added scenario

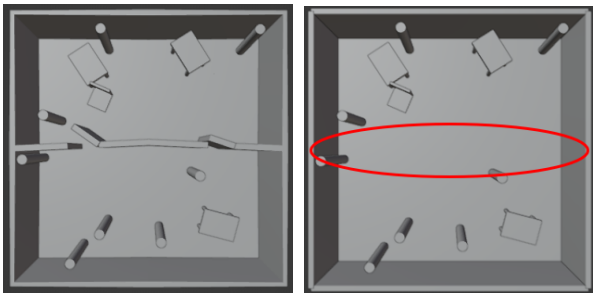
3) Pillar Removed: See example of a generated room with a removed pillar below.



(a) Original room (b) Modified room

Figure 10: Example of pillar-removed scenario

4) Wall Removed: See example of a generated room with a removed wall below.



(a) Original room (b) Modified room

Figure 11: Example of wall-removed scenario

B. Training

← Click here to go back to Pillar-Added Model

1) Pillar-Added Model: See figures below for loss curves and validation precision and recall curves for the pillar-added model.

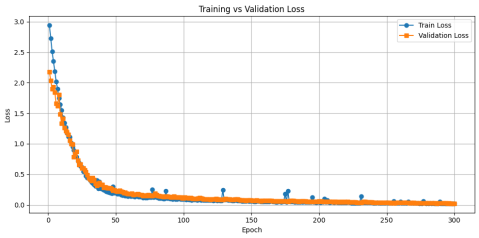


Figure 12: Training loss and validation loss for the pillar-added model.

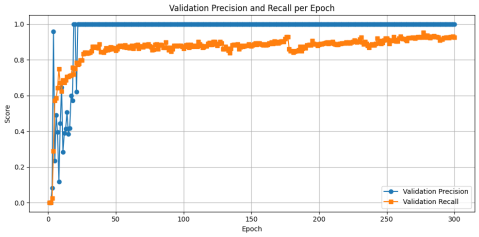


Figure 13: Validation precision and recall for the pillar-added model.

← Click here to go back to Wall-Added Model

2) Wall-Added Model: See figures below for loss curves and validation precision and recall curves for the wall-added model.

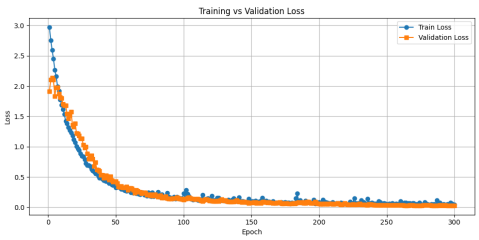


Figure 14: Training loss and validation loss for the wall-added model.

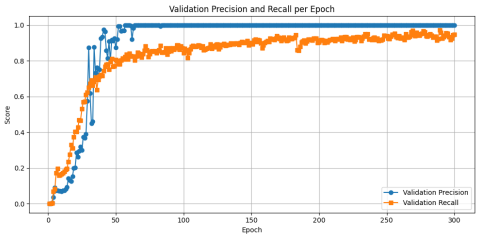


Figure 15: Validation precision and recall for the wall-added model.

← Click here to go back to Pillar-Removed Model

3) Pillar-Removed Model: See figures below for loss curves and validation precision and recall curves for the pillar-removed model.

Figure 16: Training loss and validation loss for the pillar-removed model.

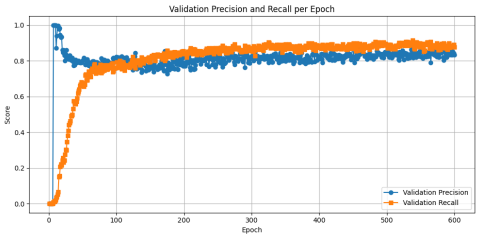


Figure 17: Validation precision and recall for the pillar-removed model.

← Click here to go back to Wall-Removed Model

4) Wall-Removed Model: See figures below for loss curves and validation precision and recall curves for the wall-removed model.

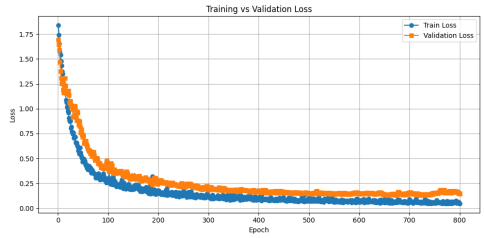


Figure 18: Training loss and validation loss for the wall-removed model.

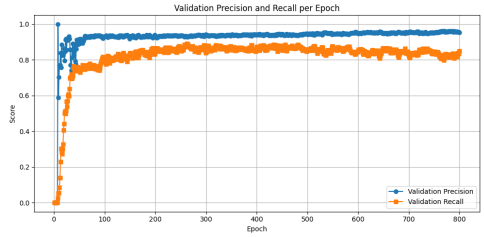


Figure 19: Validation precision and recall for the wall-removed model.

C. Testing

← Click here to go back to Results

1) Quantitative Results: See below table, Table I, for all quantitative results. Note that for multi-mismatch cases, the average scores of multiple selected models have been taken.

Table I: Summary of Evaluation Metrics for All Test Cases (Clean and Occluded)

Test Case	ROC AUC	FPR@95%Rec.	Prec.@95%Rec.	IoU@95%Rec.
Individual Models				
Pillar Added (Clean)	**1.0000**	**0.0000**	**1.0000**	**0.9494**
Pillar Added (Occluded)	**1.0000**	**0.0000**	**1.0000**	**0.9480**
Wall Added (Clean)	**0.9995**	**0.0000**	**1.0000**	**0.9495**
Wall Added (Occluded)	**0.9998**	**0.0005**	**0.9981**	**0.9479**
Pillar Removed (Clean)	**0.9999**	**0.0000**	**1.0000**	**0.9493**
Pillar Removed (Occluded)	**0.9923**	**0.0605**	0.5484	0.5326
Wall Removed (Clean)	**1.0000**	**0.0000**	**1.0000**	**0.9495**
Wall Removed (Occluded)	**0.8464**	**0.2119**	0.7238	0.6972
Ablation Studies				
Removed NN-Distance (Clean)	0.7718	0.2538	0.1395	0.1384
Removed NN-Distance (Occluded)	**0.9411**	**0.2075**	0.2300	0.2271
Removed X,Y,Z (Pillar Removed Clean)	**1.0000**	**0.0000**	**1.0000**	**0.9493**
Removed X,Y,Z (Pillar Removed Occluded)	**0.9852**	**0.1043**	0.4018	0.3932
Removed X,Y,Z & No Augmented (PR-Clean)	**1.0000**	**0.0000**	**1.0000**	**0.9493**
Removed X,Y,Z & No Augmented (PR-Occluded)	**0.9852**	**0.1043**	0.4018	0.3932
Removed X,Y,Z & No Augmented (WR-Clean)	**1.0000**	**0.0000**	**1.0000**	**0.9493**
Removed X,Y,Z & No Augmented (WR-Occluded)	**0.9859**	**0.1254**	**0.8580**	**0.8206**
Removed X,Y,Z & No Augmented (PA-Clean)	**1.0000**	**0.0000**	**1.0000**	**0.9494**
Removed X,Y,Z & No Augmented (PA-Occluded)	**1.0000**	**0.0000**	**1.0000**	**0.9480**
Removed X,Y,Z & No Augmented (WA-Clean)	**1.0000**	**0.0000**	**1.0000**	**0.9495**
Removed X,Y,Z & No Augmented (WA-Occluded)	**1.0000**	**0.0000**	**1.0000**	**0.9496**
Clean Samples Only (Pillar Removed Clean)	**0.9970**	**0.0000**	**1.0000**	**0.9493**
Clean Samples Only (Pillar Removed Occluded)	**0.9851**	**0.1026**	0.4060	0.3973
Clean Samples Only (Wall Removed Clean)	**1.0000**	**0.0000**	**1.0000**	**0.9495**
Clean Samples Only (Wall Removed Occluded)	**0.9491**	**0.1978**	0.7408	0.7129
Misaligned Scans (Clean)	**0.9872**	**0.0884**	0.3624	0.3555
Misaligned Scans (Occluded)	**0.9709**	**0.1706**	0.2754	0.2713
No Augmented Samples (Pillar Removed Clean)	**0.9982**	**0.0000**	**1.0000**	**0.9493**
No Augmented Samples (Pillar Removed Occluded)	**0.9941**	**0.0384**	0.6618	0.6389
Semi-Real-World Validation				
Pillar Added (Clean)	**1.0000**	**0.0000**	**1.0000**	**0.9491**
Pillar Added (Occluded)	**1.0000**	**0.0000**	**1.0000**	**0.9470**
Wall Added (Clean)	**1.0000**	**0.0000**	**1.0000**	**0.9498**
Wall Added (Occluded)	**1.0000**	**0.0000**	**1.0000**	**0.9500**
Pillar Removed (Clean)	**1.0000**	**0.0000**	**1.0000**	**0.9484**
Pillar Removed (Occluded)	**0.8205**	**0.2042**	0.0611	0.0609
Wall Removed (Clean)	**1.0000**	**0.0000**	**1.0000**	**0.9499**
Wall Removed (Occluded)	0.4274	0.4807	0.0921	0.0916
Multi-Mismatch Testing				
Same Object Type and Mismatch Type				
Pillars Added (Clean)	**1.0000**	**0.0000**	**1.0000**	**0.9491**
Pillars Added (Occluded)	**1.0000**	**0.0000**	**1.0000**	**0.9499**
Walls Added (Clean)	**1.0000**	**0.0000**	**1.0000**	**0.9500**
Walls Added (Occluded)	**1.0000**	**0.0000**	**1.0000**	**0.9500**
Pillars Removed (Clean)	**0.9353**	0.4840	0.2090	0.2067
Pillars Removed (Occluded)	**0.9271**	0.4678	0.1136	0.1129
Walls Removed (Clean)	**0.9784**	**0.0647**	**0.9154**	**0.8732**
Walls Removed (Occluded)	**0.9994**	**0.0052**	**0.9940**	**0.9445**
Different Object Type and Same Mismatch Type				
Objects Added (Clean)	**1.0000**	**0.0000**	**1.0000**	**0.9496**
Objects Added (Occluded)	**1.0000**	**0.0000**	**1.0000**	**0.9498**
Objects Removed (Clean)	**0.9999**	**0.0014**	**0.9975**	**0.9476**
Objects Removed (Occluded)	**0.9893**	**0.0911**	**0.7960**	0.7636
Same Object Type and Different Mismatch Type				
Pillars Added & Removed (Clean)	**0.9449**	0.2947	0.1794	0.1767
Pillars Added & Removed (Occluded)	**0.9046**	0.2797	0.1436	0.1422
Walls Added & Removed (Clean)	0.6928	0.3804	0.2254	0.2228
Walls Added & Removed (Occluded)	0.4995	0.4143	0.2160	0.2155
Different Object Type and Different Mismatch Type				
Objects Added & Removed (Clean)	0.5626	0.3960	0.2820	0.2775
Objects Added & Removed (Occluded)	0.5043	0.6400	0.2823	0.2781

2) Pillar-Added Model: *Click **here** to go back to Pillar-Added Model*

Clean Test Case

Below are the visualizations for clean test cases that have been used alongside the quantitative results to analyze the Pillar-Added model.

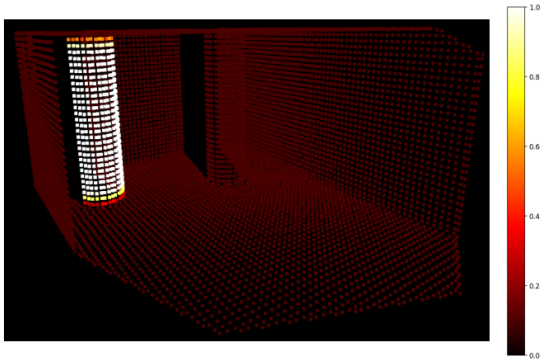


Figure 20: Heatmap of robot scan for a clean pillar-added test case. The points corresponding to the added pillar have a high predicted probability.

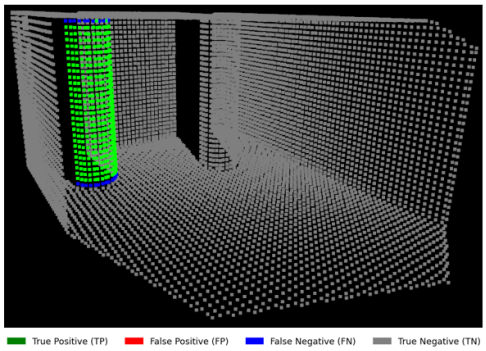


Figure 21: Points categorized in TP/FP/TN/FN in combination with a defined threshold (0.6) for the clean pillar-added test case. Note that there are no FP and only a few FN.

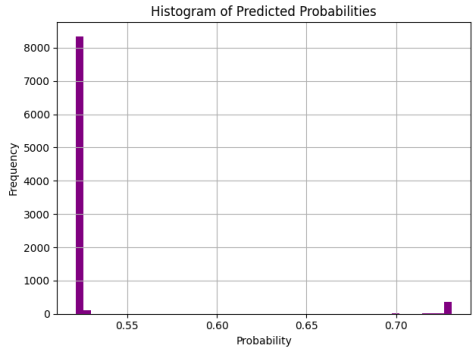
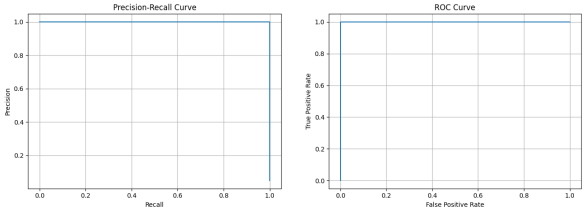


Figure 22: Histogram of predicted probabilities for a clean pillar-added test case. Note how there is a clear distinction between points with a low and high probability.



(a) PR-curve for a clean pillar-added test case.

(b) ROC-curve for a clean pillar-added test case.

Figure 23

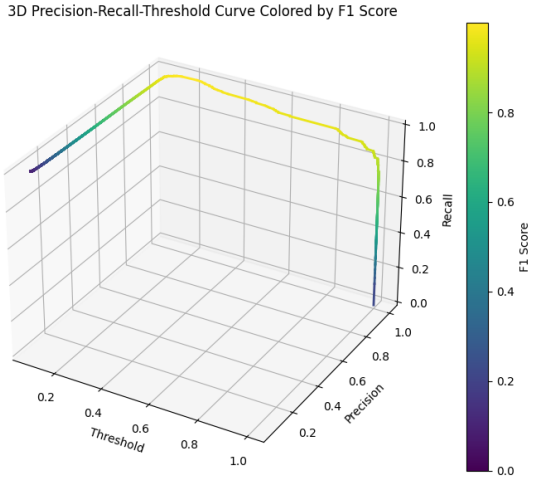


Figure 24: 3D precision-recall-threshold curve colored by F1 score for a clean pillar-added test case.

*Click **here** to go back to Pillar-Added Model*

Occluded Test Case

*← Click **here** to go back to Pillar-Added Model*

Below are the visualizations of occluded test cases that have been used alongside the quantitative results to analyze the Pillar-Added model.

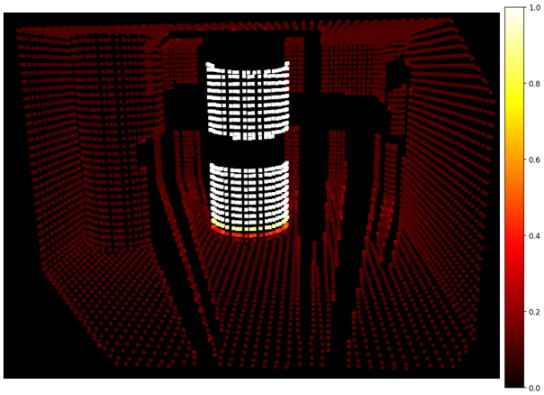


Figure 25: Heatmap of robot scan for an occluded pillar-added test case. The points corresponding to the added pillar have a high predicted probability.

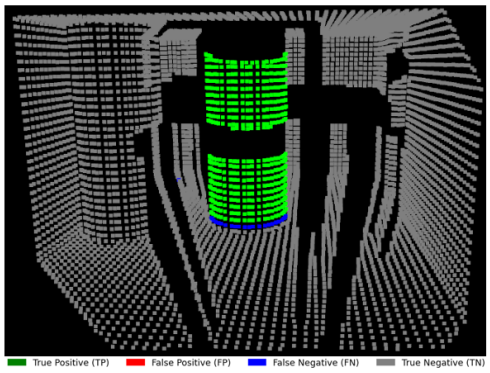


Figure 26: Points categorized in TP/FP/TN/FN in combination with a defined threshold (0.6) for the occluded pillar-added test case. Note that there are no FP and only a few FN.

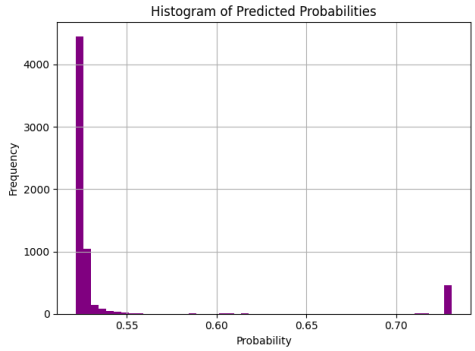
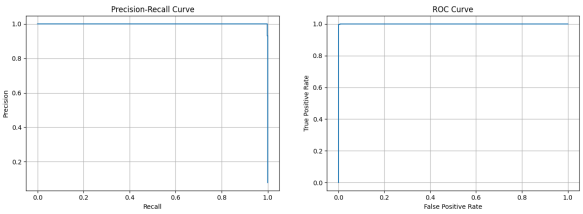


Figure 27: Histogram of predicted probabilities for an occluded pillar-added test case. Note how there is a clear distinction between points with a low and high probability.



(a) PR-curve for an occluded pillar-added test case.

(b) ROC-curve for an occluded pillar-added test case.

Figure 28

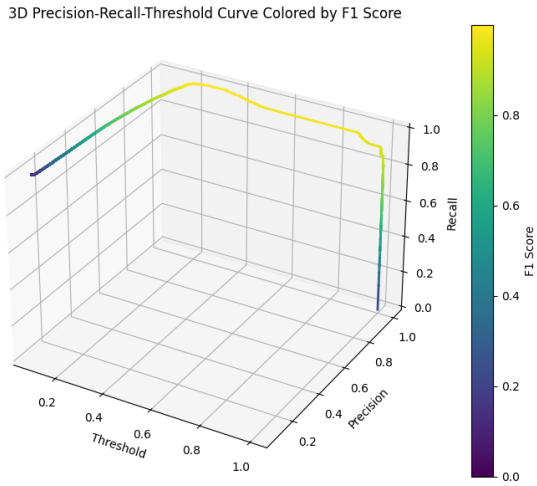


Figure 29: 3D precision-recall-threshold curve colored by F1 score for an occluded pillar-added test case.

*3) Wall-Added Model: Click **here** to go back to Wall-Added Model*

Clean Test Case

Below are the visualizations of clean test cases that have been used alongside the quantitative results to analyze the Wall-Added model.

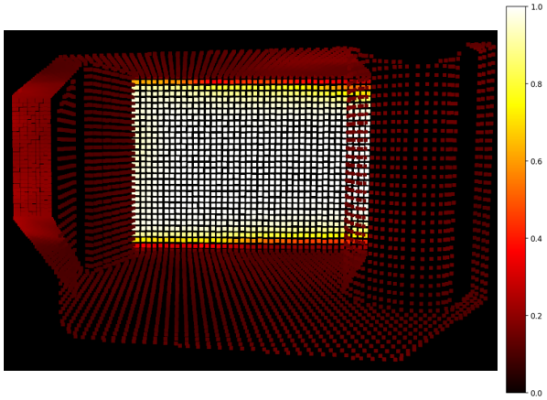


Figure 30: Heatmap of robot scan for a clean wall-added test case. The points corresponding to the added wall have a high predicted probability.

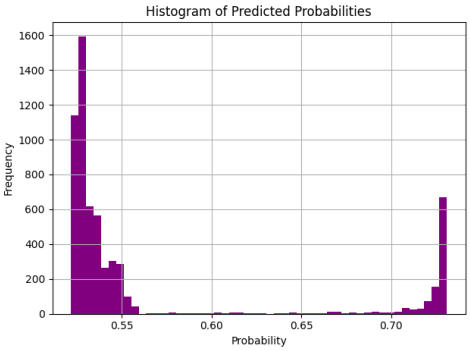
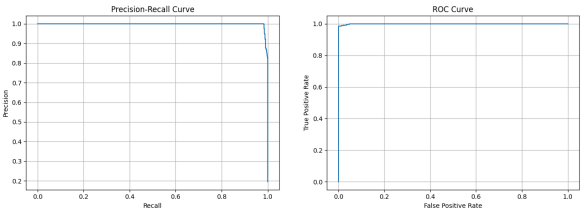


Figure 32: Histogram of predicted probabilities for a clean wall-added test case. Note how there is a clear distinction between points with a low and high probability.



(a) PR-curve for a clean wall-added test case.

(b) ROC-curve for a clean wall-added test case.

Figure 33

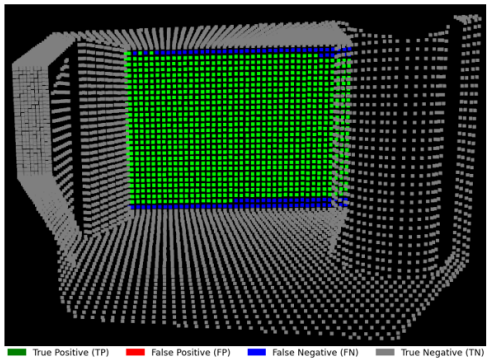


Figure 31: Points categorized in TP/FP/TN/FN in combination with a defined threshold (0.6) for the clean wall-added test case. Note that there are no FP and only a few FN.

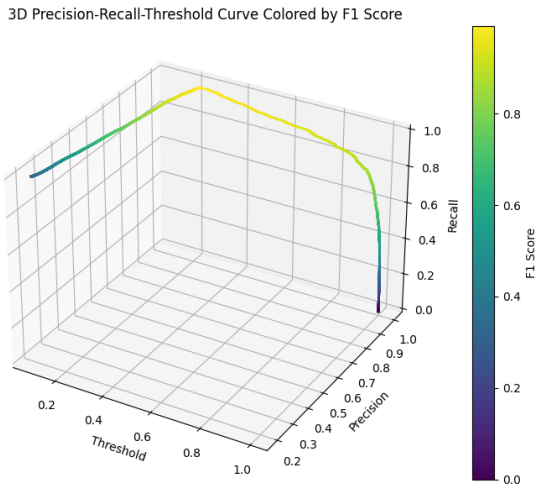


Figure 34: 3D precision-recall-threshold curve colored by F1 score for a clean wall-added test case.

← *Click **here** to go back to Wall Added Model*

Occluded Test Case

*← Click **here** to go back to Wall Added Model*

Below are the visualizations of occluded test cases that have been used alongside the quantitative results to analyze the wall-added model.

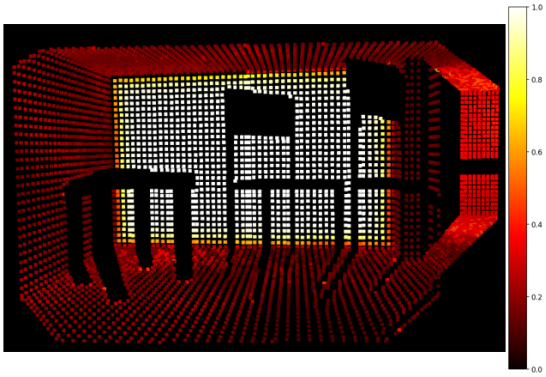


Figure 35: Heatmap of robot scan for an occluded wall-added test case. The points corresponding to the added wall have a high predicted probability.

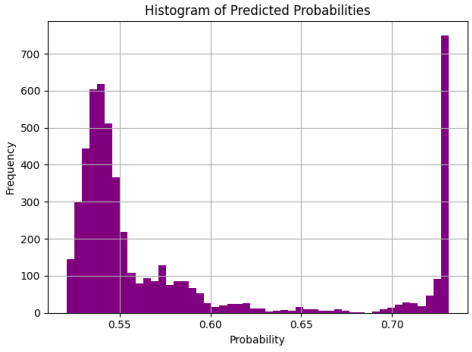
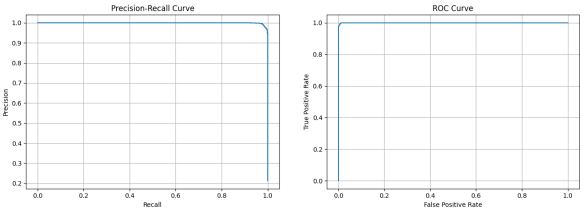


Figure 36: Histogram of predicted probabilities for an occluded wall-added test case. Note how there is a clear distinction between points with a low and high probability.



(a) PR-curve for an occluded wall-added test case.

(b) ROC-curve for an occluded wall-added test case.

Figure 37

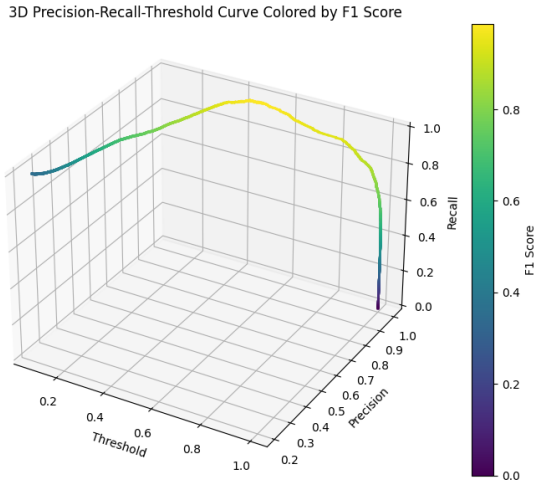


Figure 38: 3D precision-recall-threshold curve colored by F1 score for an occluded wall-added test case.

4) Pillar-Removed Model: *Click **here** to go back to Pillar-Removed Model*

Clean Test Case

Below are the visualizations of clean test cases that have been used alongside the quantitative results to analyze the pillar-removed model.

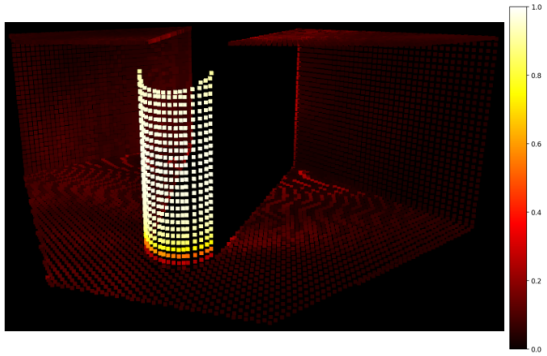


Figure 39: Heatmap of DT scan for a clean pillar-removed test case. The points corresponding to the removed pillar have a high predicted probability.

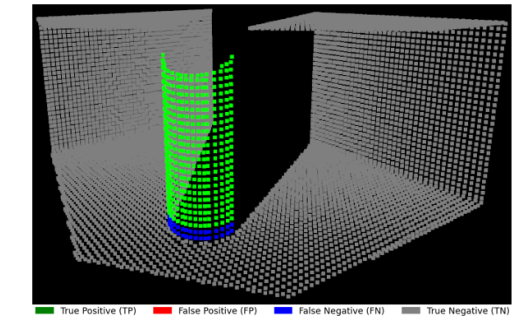


Figure 40: Points categorized in TP/FP/TN/FN in combination with a defined threshold (0.6) for the clean pillar-removed test case. Note that there are no FP and only a few FN.

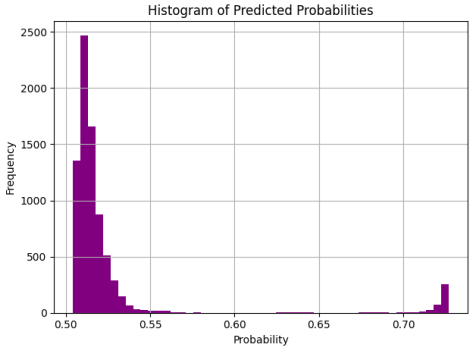
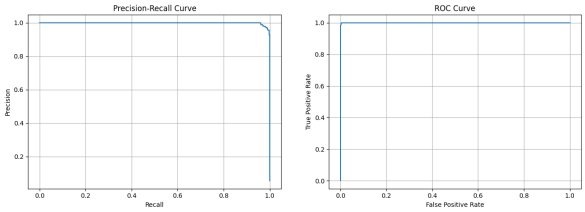


Figure 41: Histogram of predicted probabilities for a clean pillar-removed test case. Note how there is a clear distinction between points with a low and high probability.



(a) PR-curve for a clean pillar-removed test case.

(b) ROC-curve for a clean pillar-removed test case.

Figure 42

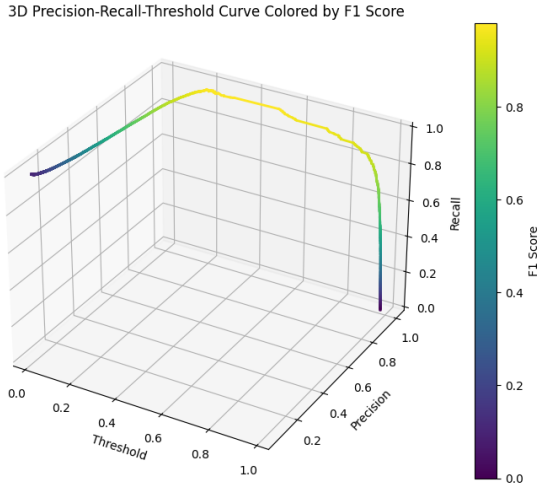


Figure 43: 3D precision-recall-threshold curve colored by F1 score for a clean pillar-removed test case.

Occluded Test Case

*← Click **here** to go back to Pillar-Removed Model*

Below are the visualizations of occluded test cases that have been used alongside the quantitative results to analyze the pillar-removed model.

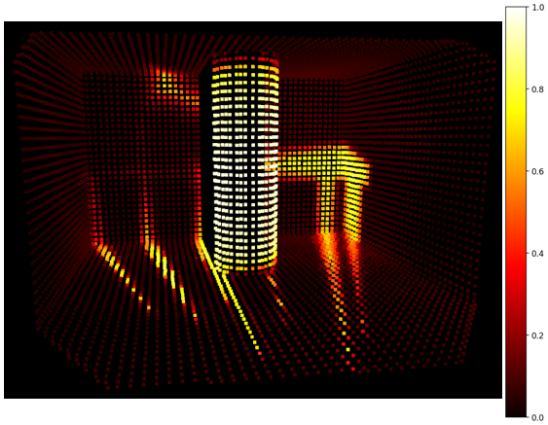


Figure 44: Heatmap of DT scan for an occluded pillar-removed test case. The points corresponding to the removed pillar have a high predicted probability, but there are also unwanted shadows with high predicted mismatch probability.

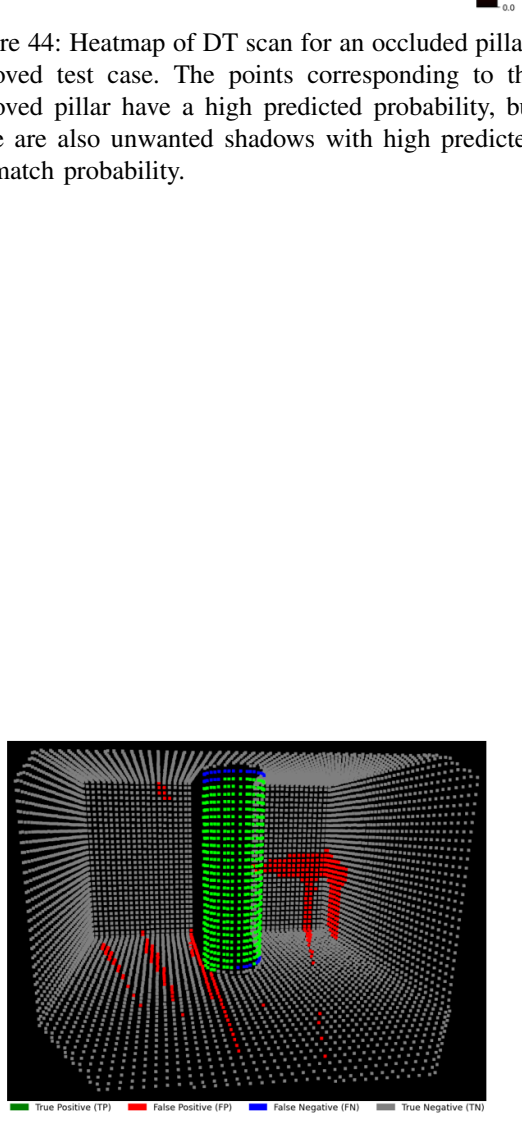


Figure 45: Points categorized in TP/FP/TN/FN in combination with a defined threshold (0.6) for the occluded pillar-removed test case. Note that the FP correspond to the shadows.

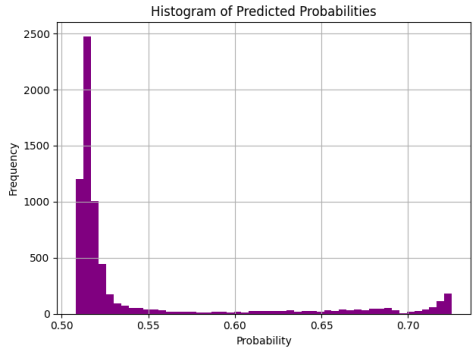
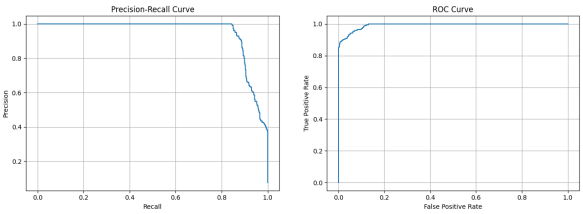


Figure 46: Histogram of predicted probabilities for an occluded pillar-removed test case. Note how there is a clear distinction between points with a low and high probability.



(a) PR-curve for an occluded pillar-removed test case.

(b) ROC-curve for an occluded pillar-removed test case.

Figure 47

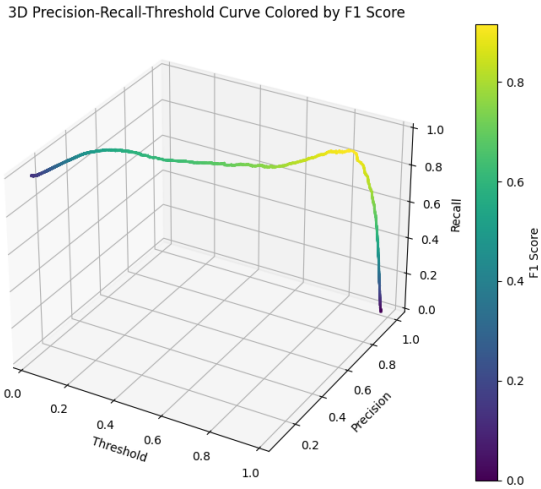


Figure 48: 3D precision-recall-threshold curve colored by F1 score for an occluded pillar-removed test case.

5) Wall Removed Model: *Click **here** to go back to Wall-Removed Model*

Clean Test Case

Below are the visualizations of clean test cases that have been used alongside the quantitative results to analyze the wall-removed model.

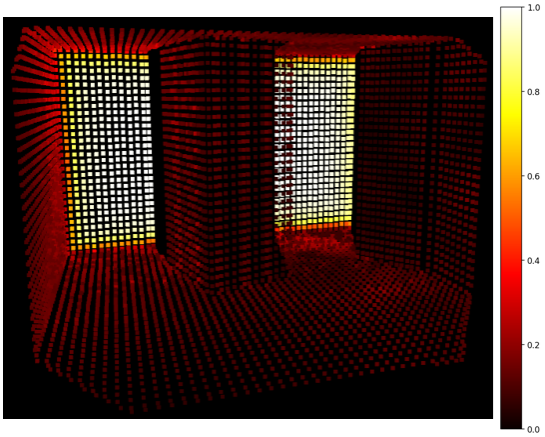


Figure 49: Heatmap of DT scan for a clean wall-removed test case. The points corresponding to the removed wall have a high predicted probability.

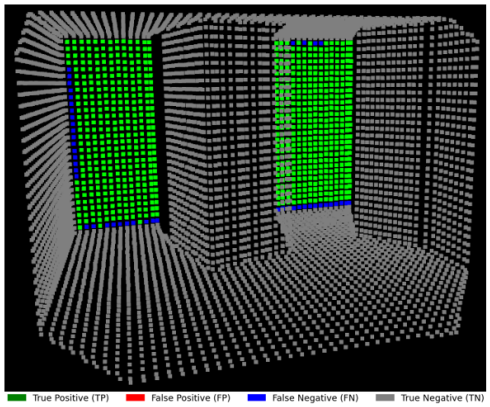


Figure 50: Points categorized in TP/FP/TN/FN in combination with a defined threshold (0.6) for the clean wall-removed test case. Note that there are no FP and only a few FN.

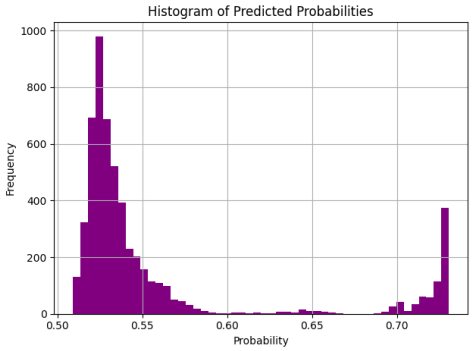
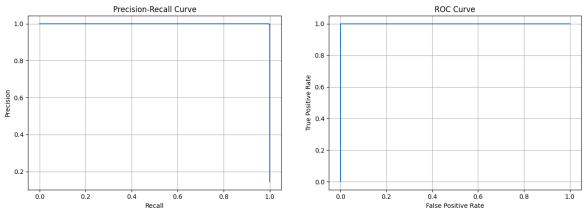


Figure 51: Histogram of predicted probabilities for a clean wall-removed test case. Note how there is a clear distinction between points with a low and high probability.



(a) PR-curve for a clean wall-removed test case.

(b) ROC-curve for a clean wall-removed test case.

Figure 52

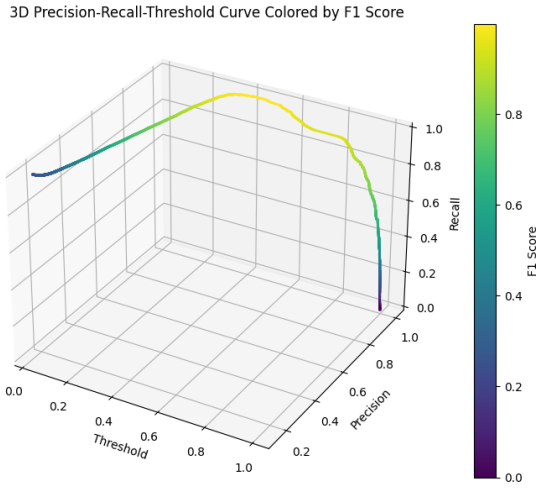


Figure 53: 3D precision-recall-threshold curve colored by F1 score for a clean wall-removed test case.

Occluded Test Case

*Click **here** to go back to Wall-Removed Model*

Below are the visualizations of occluded test cases that have been used alongside the quantitative results to analyze the wall-removed model.

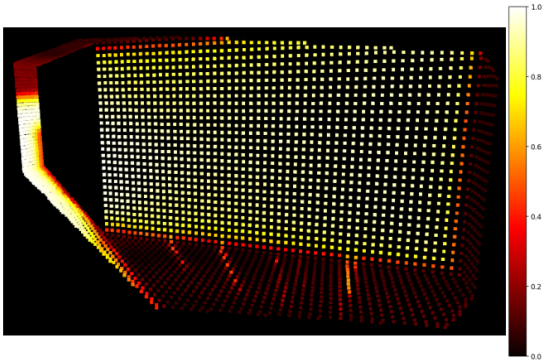


Figure 54: Heatmap of DT scan for an occluded wall-removed test case. The points corresponding to the removed wall have a high predicted probability, but there is also a large unwanted shadow to the left.

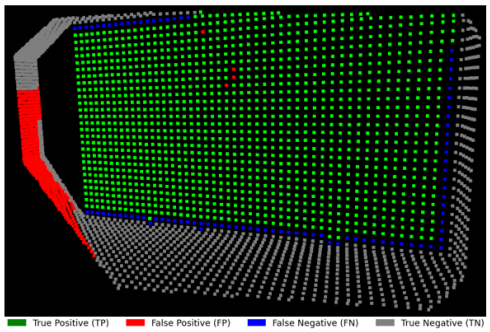


Figure 55: Points categorized in TP/FP/TN/FN in combination with a defined threshold (0.6) for the occluded wall-removed test case. Note the large FP shadow on the left.

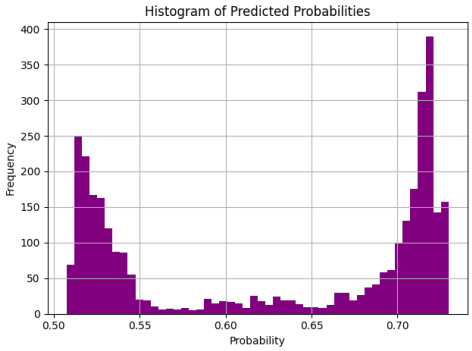
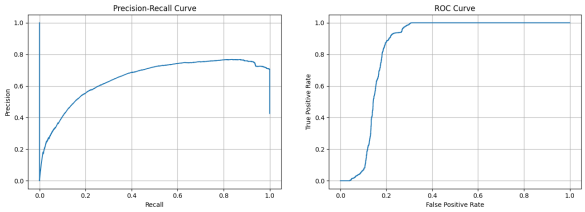


Figure 56: Histogram of predicted probabilities for an occluded wall-removed test case. Note how there is a clear distinction between points with a low and high probability.



(a) PR-curve for an occluded wall-removed test case.

(b) ROC-curve for an occluded wall-removed test case.

Figure 57

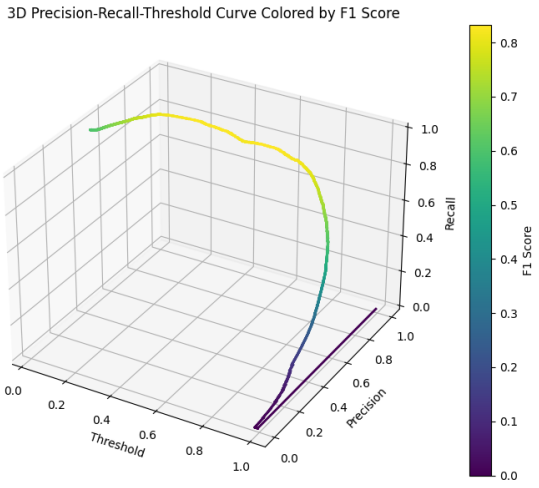


Figure 58: 3D precision-recall-threshold curve colored by F1 score for an occluded wall-removed test case.

← Click here to go back to Evaluation Model Selection

D. Model Selection

Below are some examples on bounding box fitting on synthetic single-mismatch test cases.

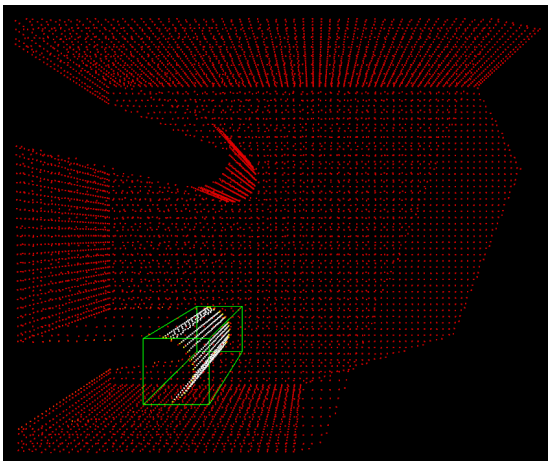


Figure 59: Example of a bounding box (green for added object) fitted around a clearly visible circular mismatched pillar from a top-down view. Note how the box center is slightly off since the center of the cluster of mismatched points is closer to the sensor that the actual pillar center.

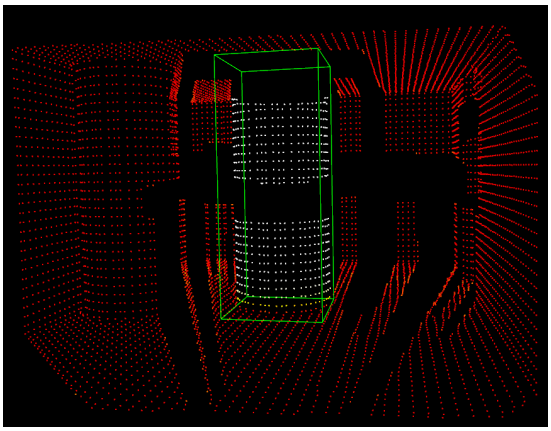


Figure 60: Example of a bounding box fitted around a partially visible circular pillar.

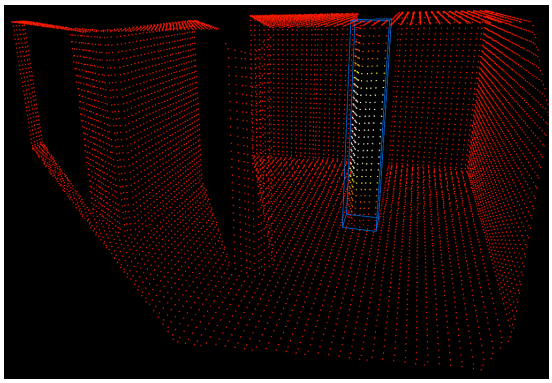


Figure 61: Example of a bounding box (blue for removed object) fitted around a clearly visible square pillar.

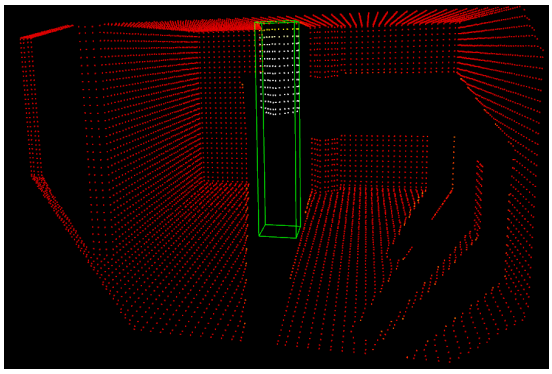


Figure 62: Example of a bounding box fitted around a partially visible circular pillar.

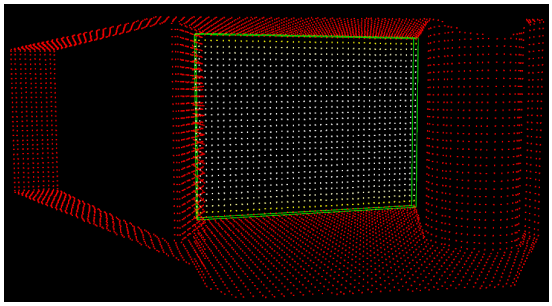


Figure 63: Example of a bounding box fitted around a clearly visible wall.

← Click here to go back to Evaluation Model Selection

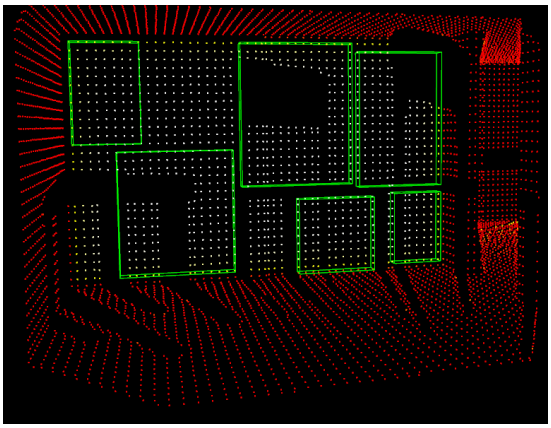


Figure 64: Example of several bounding boxes fitted around a partially visible wall.

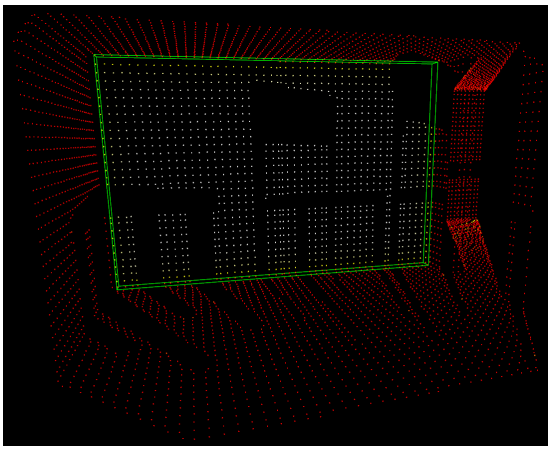


Figure 66: Example of a bounding box fitted around a partially visible wall. Also note how it is now one large bounding box instead of separate ones.

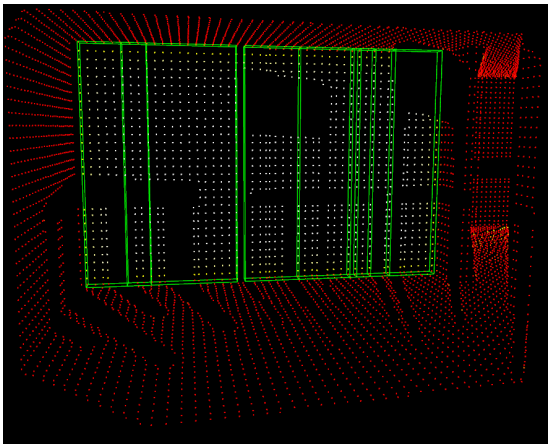


Figure 65: Example of the above bounding boxes stretched along the vertical span of the point cloud / room.

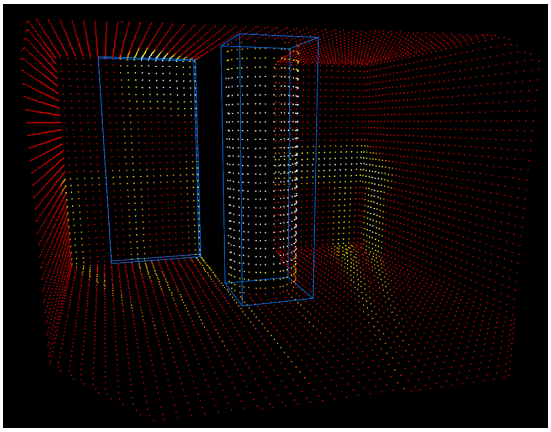


Figure 67: Example of an incorrect bounding box (left) fitted around a shadow artefact of a removed chair.

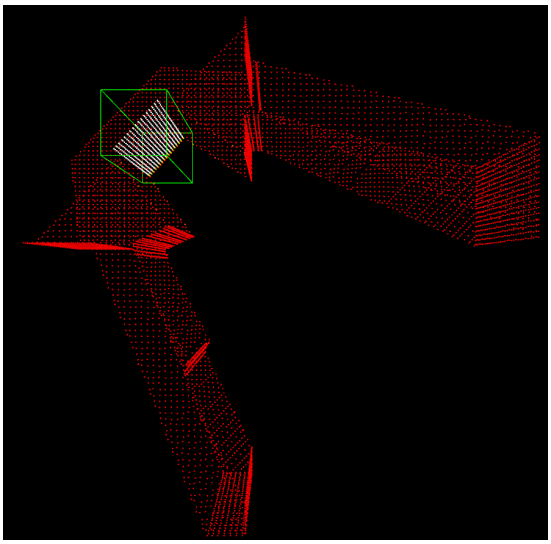


Figure 68: Example of a bounding box fitted around a clearly visible rectangular mismatched pillar that is directly facing the sensor with only one of its sides. Note how the fitted pillar box is falsely oriented because of the Manhattan World assumption, but how there still is a distinction being made between what looks like a wall, but actually is a pillar.

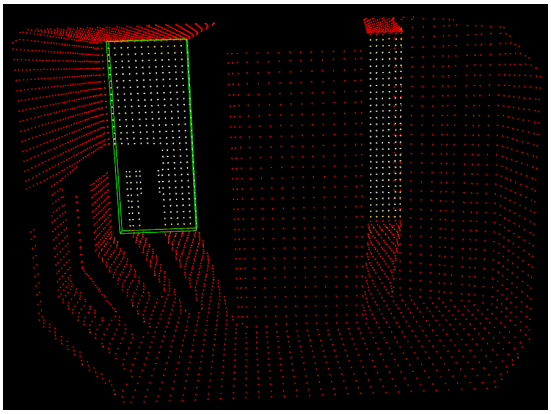


Figure 69: Example of a split wall case where only one portion gets a bounding box because it is of sufficient length.

← Click here to go back to Mismatches and Scenario Control

← Click here to go back to Semi-Real-World Validation

E. Semi-Real-World Validation

Below some examples on semi-real world validation for mismatch detection and bounding box fitting are given.

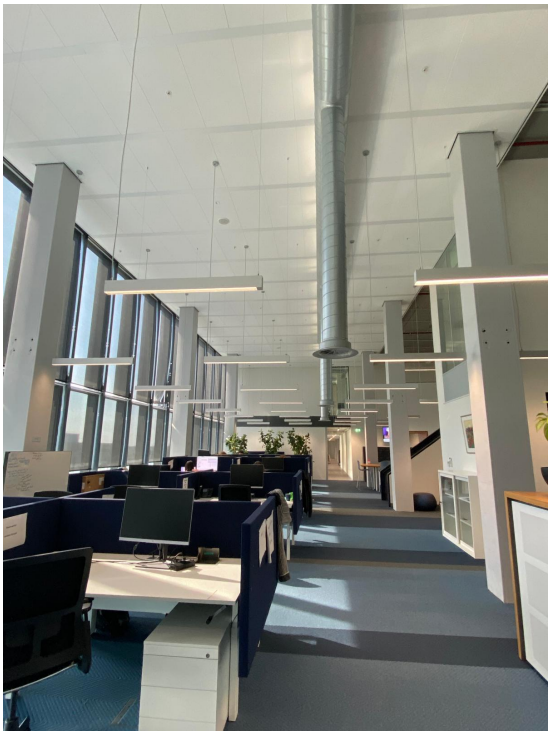


Figure 70: Atlas-8 floor inside the Atlas building on the TU/e campus.

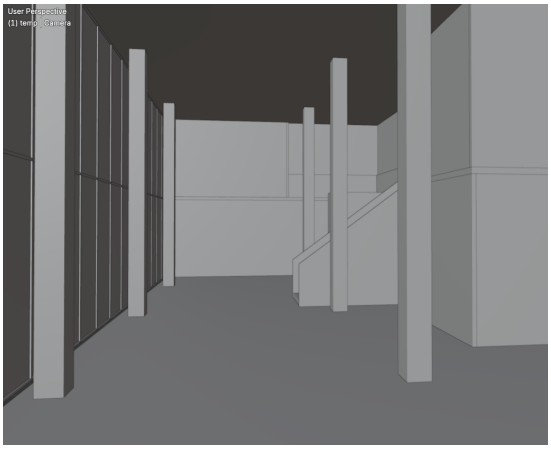


Figure 71: IFC file of Atlas-8 floor from the same viewpoint

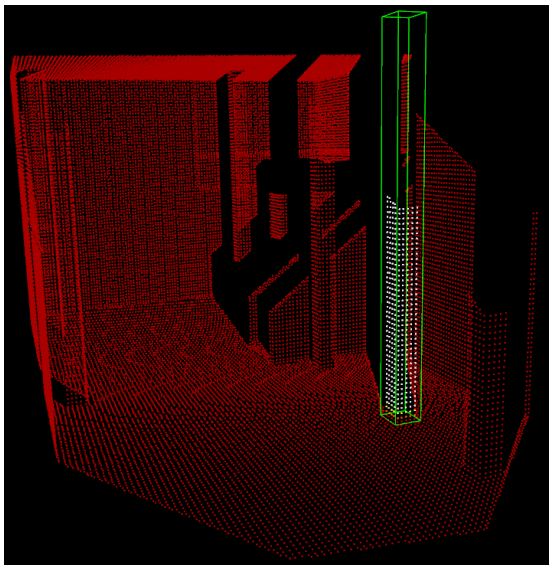


Figure 72: Result of the pillar-added model for a clean test case inside the Atlas building. The pillar is correctly predicted as mismatched and a bounding box is fitted around it.

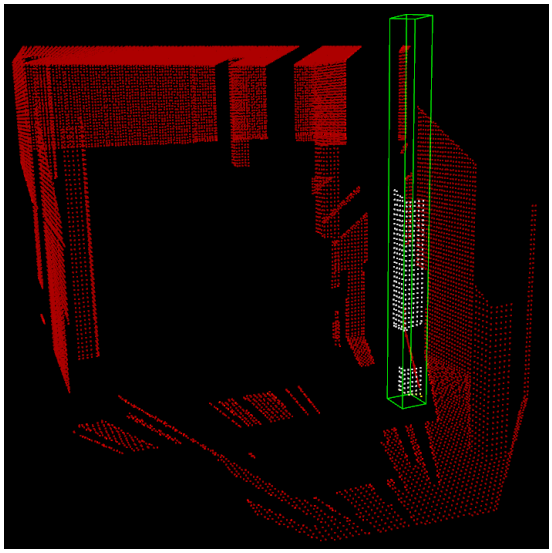


Figure 73: Result of the pillar-added model for an occluded test case inside the Atlas building. The pillar is correctly predicted as mismatched and a bounding box is fitted around it.

← Click here to go back to Semi-Real-World Validation

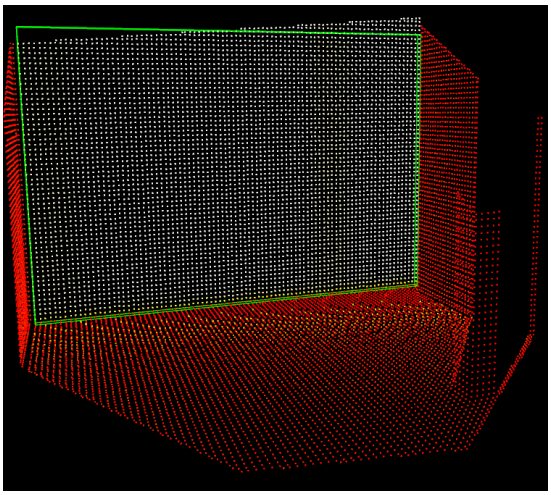


Figure 74: Result of the wall-added model for a clean test case inside the Atlas building. The wall is correctly predicted as mismatched and a bounding box is fitted around it.

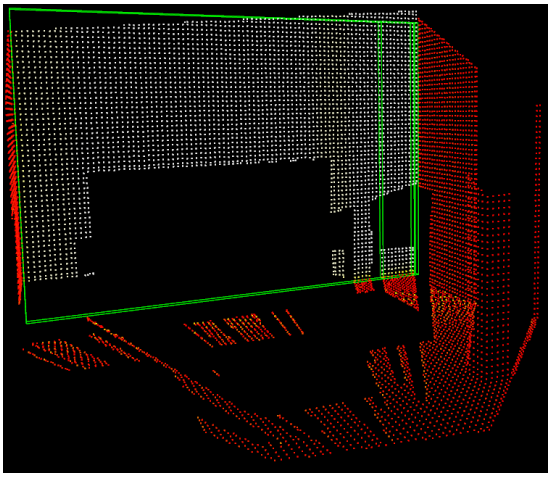


Figure 75: Result of the wall-added model for an occluded test case inside the Atlas building. The wall is correctly predicted as mismatched and a bounding box is fitted around it. There is an overlapping box on the right as well.

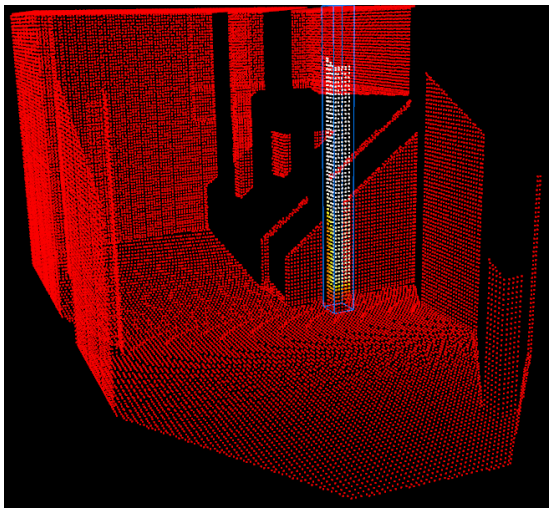


Figure 76: Result of the pillar-removed model for a clean test case inside the Atlas building. The pillar is correctly predicted as mismatched and a bounding box is fitted around it.

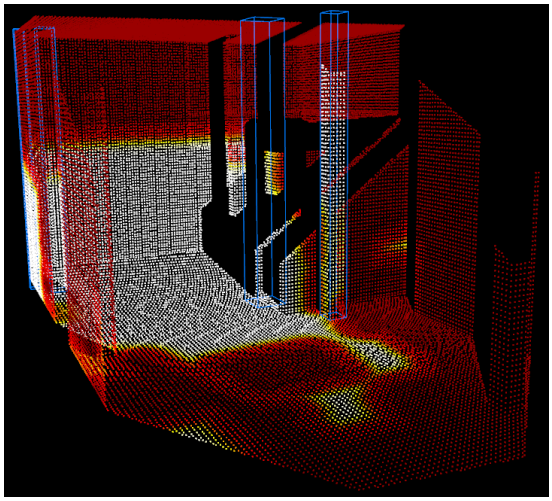


Figure 77: Result of the pillar-removed model for an occluded test case inside the Atlas building. The pillar (right) is correctly predicted as mismatched and a bounding box is fitted around it. There are some false bounding boxes present as well however due to shadow artefacts.

← Click here to go back to Semi-Real-World Validation

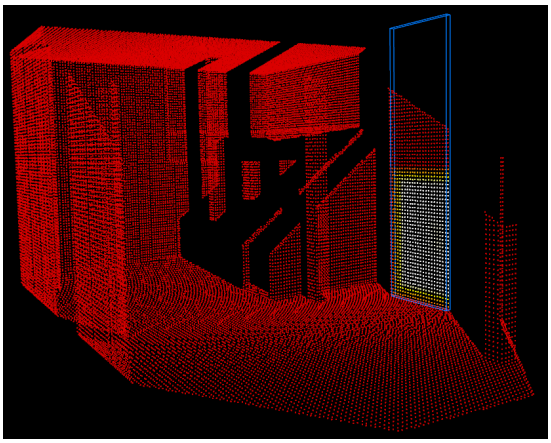


Figure 78: Result of the wall-removed model for a clean test case inside the Atlas building. The wall is correctly predicted as mismatched and a bounding box is fitted around it. Note that due to vertical stretching the bounding box incorrectly spans the entire vertical of the point cloud. This can be disabled to get the figure below.

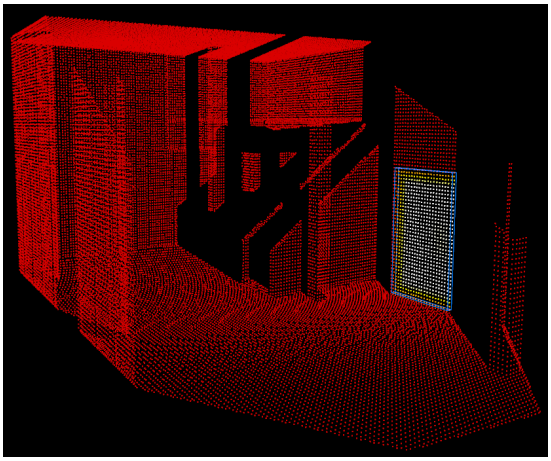


Figure 79: Result of the wall-removed model for a clean test case inside the Atlas building. The wall is correctly predicted as mismatched and a bounding box is fitted around it. Note that the bounding box is now correctly fit around only the actual mismatched wall.

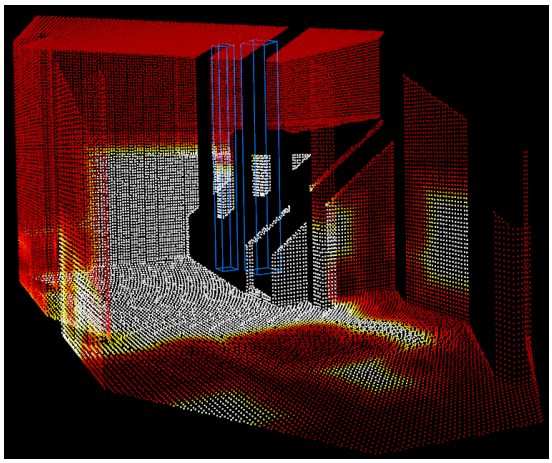


Figure 80: Result of the wall-removed model for an occluded test case inside the Atlas building. The mismatched wall (right) does not get a bounding box since the points have a too low predicted probability of being mismatched (< 0.5) while there are some pillar points that falsely get a bounding box because of shadow artefacts.

← Click here to go back to Semi-Real-World Validation

← Click here to go back to Multi-Mismatch Scenario Evaluation

F. Multi-Mismatch Scenario Evaluation

1) Same Object and Mismatch Type: *Click **here** to go back to Same Object and Mismatch Type*

Below some examples of multi-mismatch test cases are given where the object type and mismatch type are the same between objects.

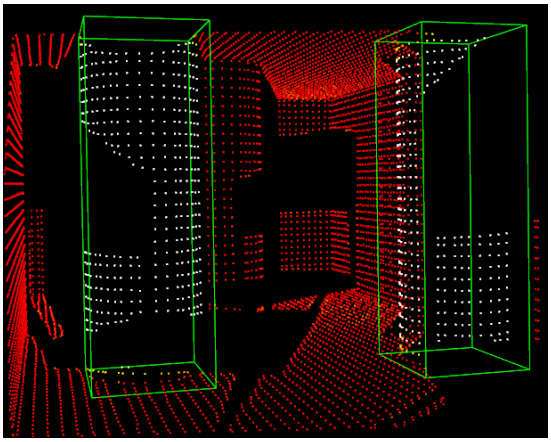


Figure 81: Example of working pillars-added test case. Multiple bounding boxes are fitted correctly around the correctly predicted mismatched points.

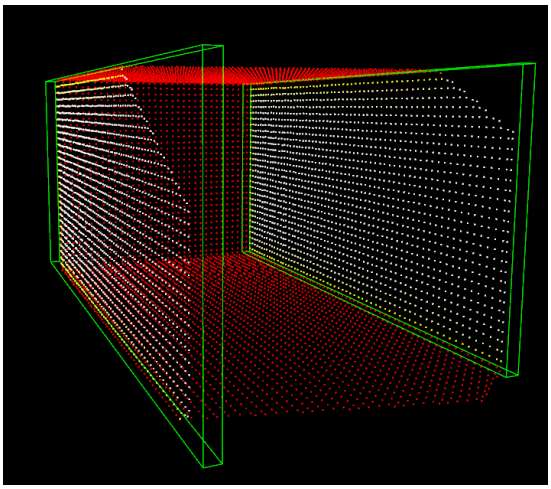


Figure 82: Example of working walls-added test case. Two bounding boxes are fitted correctly around the correctly predicted mismatched points.

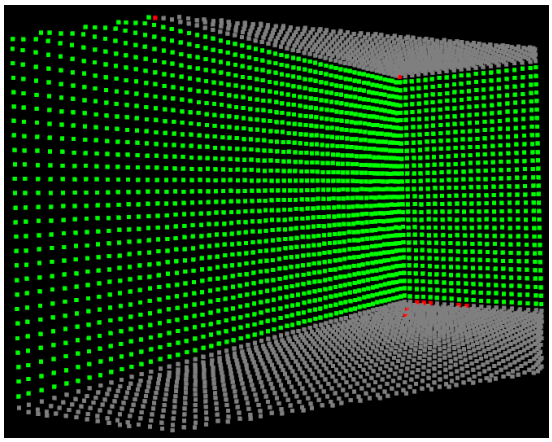


Figure 83: Example of two added walls that are connected but do not get a bounding box because the mismatched points form one gigantic cluster instead of two separate clusters.

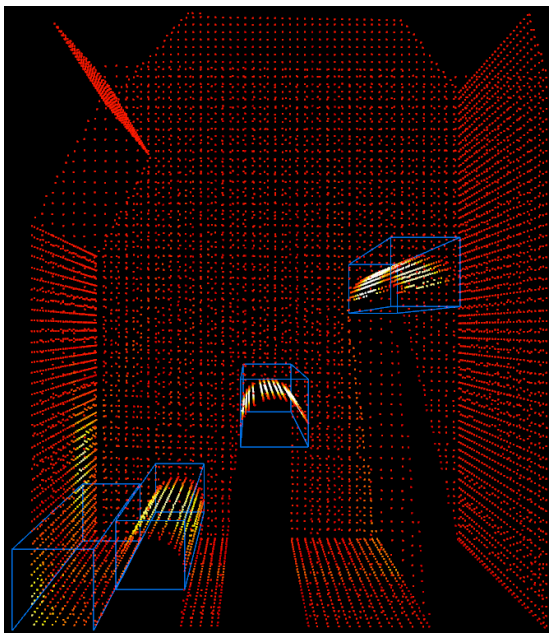


Figure 84: Example of three removed pillars that get a correct bounding box, but also a shadow artefact incorrectly getting a bounding box (right).

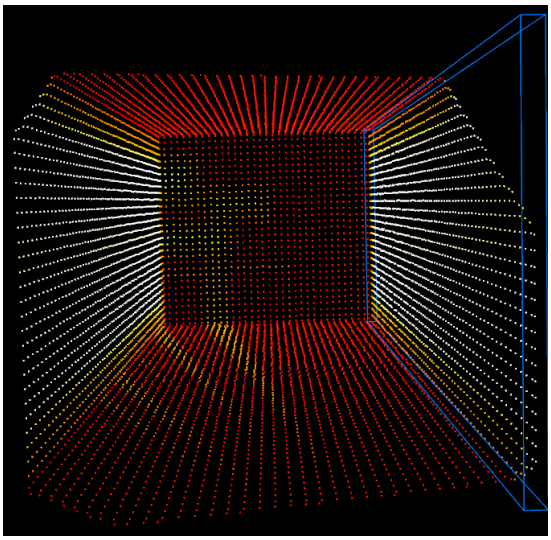


Figure 85: Example of two removed walls of which only one gets a bounding box because the other cluster of mismatched points is affected by the shadow artefact on the back wall which results in a too wide and invalid bounding box.

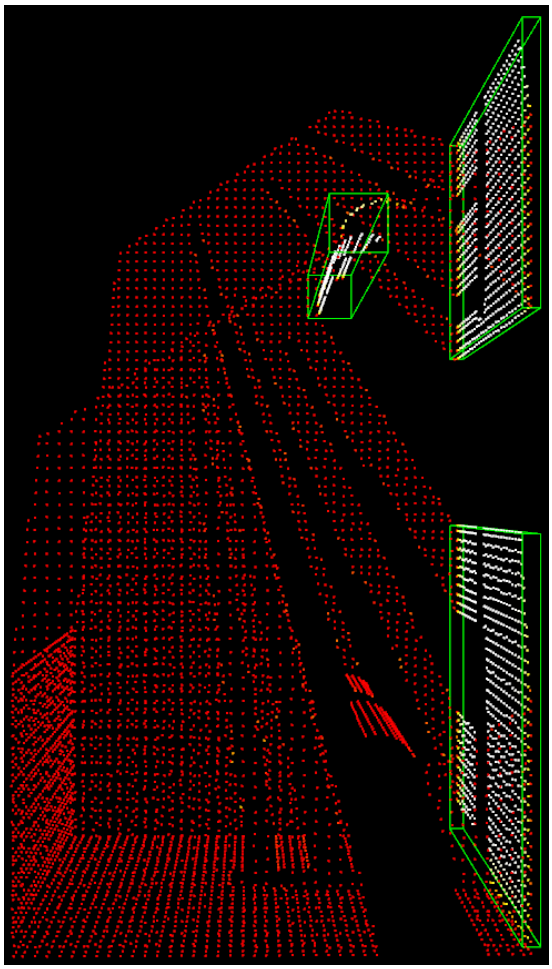


Figure 86: Example of an occluded multi-mismatch test case where a pillar and wall have been added. Note how the wall is split into two separate bounding boxes because of the occluding added pillar.

2) Different Object Type and Same Mismatch Type:
*Click **here** to go back to Different Object and Same Mismatch Type*

Below some examples of multi-mismatch test cases are given where the object type is different and mismatch type is the same between objects.

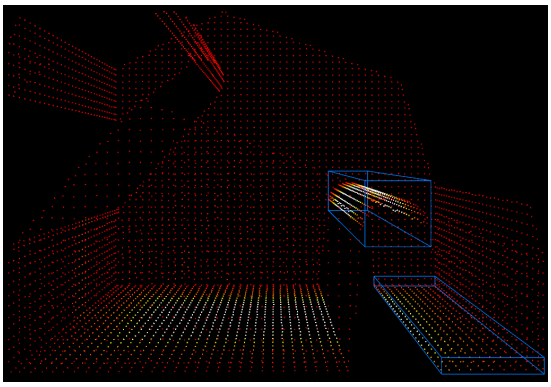


Figure 87: Example of an occluded multi-mismatch test case where a pillar and wall have been removed. Note how the wall is split into two but only one portion gets a box because the other one is affected by a shadow artefact.

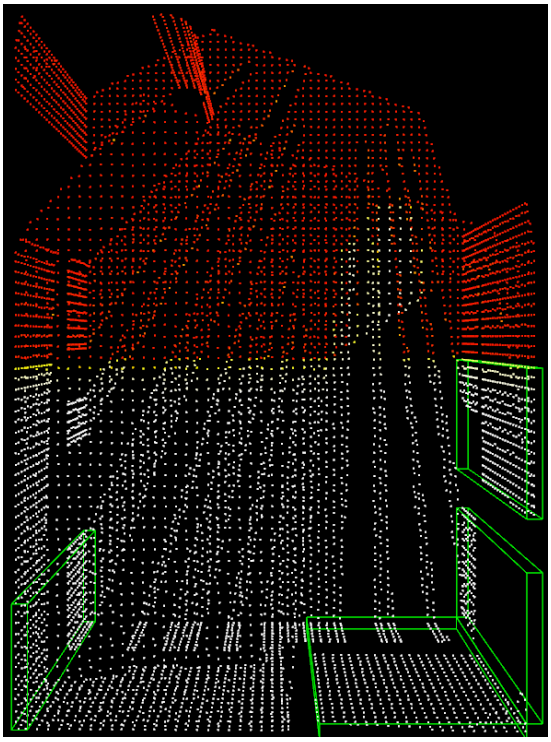


Figure 88: This figure is the robot scan that belongs to the same scan pair as the DT scan in the above figure. Since the selection of multiple models is enabled here, all points corresponding to the space behind the removed wall are marked as mismatched even though they are actually not, resulting in some false bounding boxes.

3) Same Object Type and Different Mismatch Type: *Click **here** to go back to Same Object Type and Different Mismatch Type*

Below an example of multi-mismatch test cases is given where the object type is the same and mismatch type is different between objects.

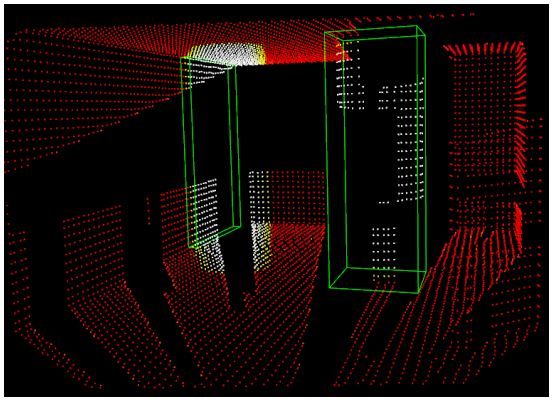


Figure 89: Example of a robot scan from an occluded test case where a pillar has been added and removed. Note how the barely visible pillar correctly gets a bounding box, but how a small piece of background wall incorrectly gets a bounding box due to a shadow artefact that is caused by the removed pillar.

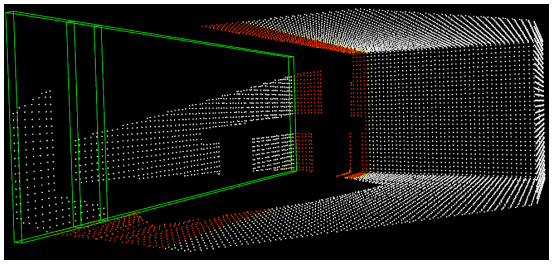


Figure 90: Example of a robot scan from an occluded test case where a wall has been added and removed. Note how the barely visible wall correctly gets a bounding box.

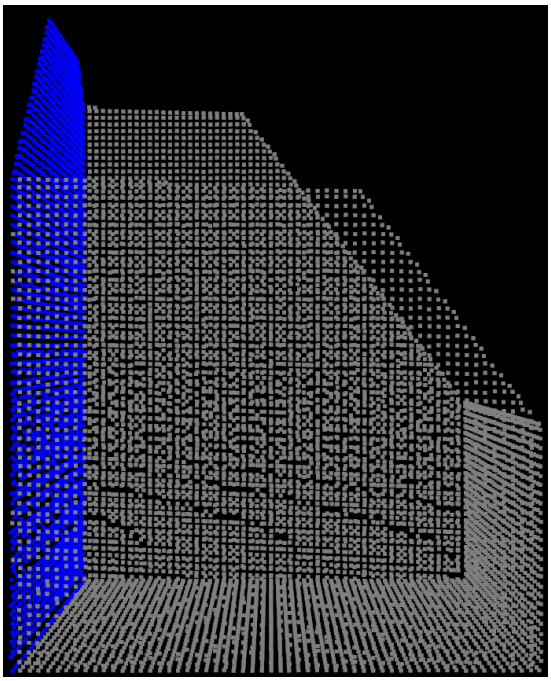


Figure 91: Example of a DT scan from an occluded test case where a wall has been added and removed. Note how the wall-removed model fails to detect the points corresponding to the removed wall and thus how no bounding box is fitted.

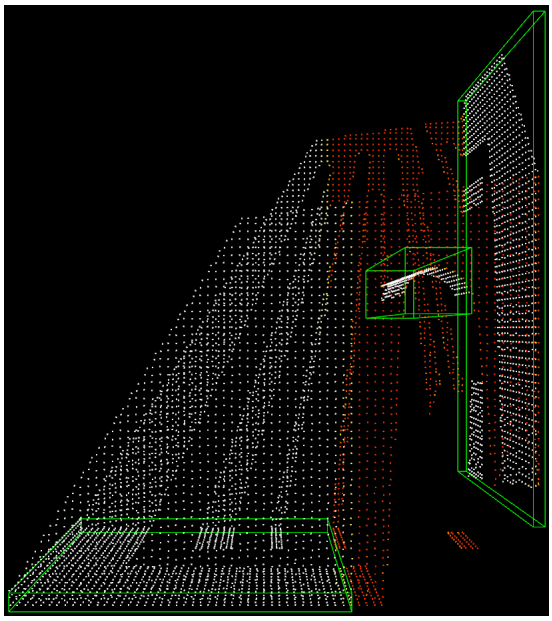


Figure 92: robot scan predictions and fitted bounding boxes for the multi-mismatch occluded test case where all four scenarios are active simultaneously. The added wall and pillar correctly get a bounding box, but there is also a false wall-added bounding box.

4) Different Object and Mismatch Type: *Click **here** to go back to Different Object and Mismatch Type*

Below some examples of multi-mismatch test cases are given where the object type and mismatch type are different between objects.

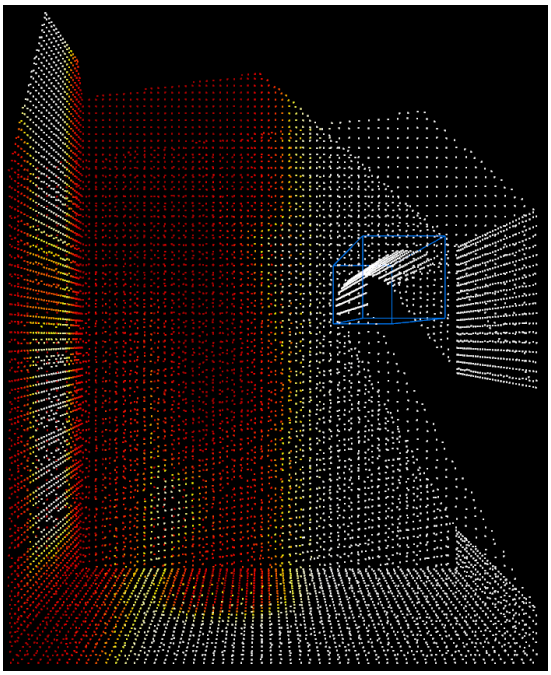


Figure 93: DT scan predictions and fitted bounding boxes for the multi-mismatch occluded test case where all four scenarios are active simultaneously. The removed pillar correctly gets a bounding box even though it is surrounded by shadow artefacts. The removed wall does not get a bounding box because the points have a too low predicted probability of being mismatched.