Towards scan-to-BIM automation: geometric and semantic reconstruction of structural components from point clouds

Vom Fachbereich
Bauingenieurwesen
der Rheihnland-Pfälzischen Technischen Universität Kaiserslautern-Landau
zur Verleihung des akademischen Grades

DOKTOR-INGENIEUR (Dr.-Ing.)

genehmigte

DISSERTATION

von

Fabian Kaufmann, M. Eng.

aus Kaiserslautern

Dekan	Prof. Dr.-Ing. Karsten Körkemeyer
1. Berichterstatter	Prof. Dr.-Ing. Christian Glock
2. Berichterstatter	Prof. Dr.-Ing. Karsten Körkemeyer
Tag der mündlichen Prüfung	19.05.2025

Kaiserslautern 2025

(D 386)

Fabian Kaufmann

Towards scan-to-BIM automation: geometric and semantic reconstruction of structural components from point clouds

Preamble

Pursuing a doctoral thesis, and the doctorate as a whole, is an inherently personal endeavour aimed at advancing one's standing in the academic community. However, no one can complete this journey alone. The support of supervisors and mentors, the exchange of ideas with colleagues, and the encouragement from friends and family are all indispensable. In this regard, I have been fortunate to receive tremendous support from my exceptional network, as well as from industry experts and other research institutions.

This doctoral thesis marks the emergence of a new research field within the Department of Concrete Structures and Structural Design at University of Kaiserlautern-Landau (RPTU). With a well-established reputation in concrete research and the technical assessment of existing concrete structures, it was time to extend these long-standing research activities into the burgeoning field of digital methods, particularly reality capturing and Building Information Modelling (BIM). However, venturing into uncharted territory is never without its challenges. Beyond gaining an overview of the existing research in this area, there was also a need to acquire and expand expertise in relevant tools and methodologies. In this regard, it was a privilege to engage with such diverse fields, including reality capturing, computer science, machine learning, and data analysis.

Civil engineering research, as an applied science with a long-standing tradition of industry engagement, must continually address the practical application of its findings. It is understood that research does not always yield tools and methods that can be immediately implemented. Nevertheless, the civil engineering field stands on the brink of adopting digital methods that will facilitate the broader application of Artificial Intelligence (AI), robotics, and automation. It is the responsibility of researchers to support this transformation by adopting a broader view of the topics at hand and expanding their perspectives. I hope this thesis is deemed valuable by the industry, as it encompasses the theoretical background, method development, implementation, and evaluation based on real-world data. It clearly demonstrates the potential of these approaches and identifies areas for further improvement and adoption of such tools and methods.

Moreover, I hope this work inspires other researchers and industry professionals to actively engage in the field of delivering Building Information Models (BIMs) of existing structures for specific applications. A doctoral thesis, by nature, extends the state of the art in specific areas. Yet, beyond this, there remain numerous intriguing topics to explore.

I am deeply grateful for the support I received throughout this work. Foremost, I thank my wife, Mirjam, and my children, Luisa and Noah, who have been unwaveringly supportive. During the countless hours of studying, researching, testing, and writing, my parents and in-laws generously cared for my children, allowing me the time necessary to complete this thesis.

I am also grateful for the tremendous support from my colleagues. In particular, I would like to mention Marius Schellen, who was always available for discussions, and Mahdi Chamseddine from German

Research Centre for Artificial Intelligence (DFKI), a remarkable researcher with extensive knowledge of 3D data and algorithms. I am thankful for their thoughtful insights, valuable suggestions, Python tips, and constructive criticism.

A special thanks goes to Dr. Jason Rambach for his guidance on improving this thesis and for our collaboration in securing funding for this work through the HumanTech project. My gratitude also extends to Prof. Dr.-Ing. Christian Glock for his supervision, guidance, and the opportunity to undertake this project, and to Prof. Dr.-Ing. Karsten Körkemeyer for his thorough review.

Contents

Abstract

As Building Information Modelling (BIM) continues to gain widespread adoption in the industry, its application in existing building works becomes increasingly relevant. To effectively apply BIM in alteration, renovation, and demolition projects, a model of the existing structure is crucial. While traditional methods relied on manual measurements and conventional surveying equipment, modern advancements have introduced efficient reality-capturing devices that combine Terrestrial Laser Scanners (TLS) and Structure from Motion (SfM) to rapidly acquire 3D data of entire buildings. However, the point clouds generated by these devices represent buildings as a dense collection of points, each covering physical objects with millions of data points. Since point clouds are unstructured data, they do not provide any further information about the building. Hence, it is essential to convert this data into Building Information Models (BIMs). Manually performing this conversion requires substantial effort, making the automation of the *scan-to-BIM* process critical for integrating existing structures into BIM workflows.

This work proposes a methodology for automatically generating BIMs from point clouds. The *ScaleBIM* framework consists of three main steps: (i) Semantic segmentation, where each point in the point cloud is assigned a semantic label, typically corresponding to a building element class. (ii) Geometry reconstruction, utilizing topology-aware refinement procedures to ensure the creation of a watertight BIM model that is functional for other use cases. (iii) Delivering the model in an open data format using open-source BIM authoring tools to maximize interoperability. These three steps are integrated into a comprehensive pipeline and rigorously tested on two datasets with varying characteristics. The pipeline reconstructs walls, doors and columns and delivers the respective BIM objects including the building component class and geometric shape representation.

Since the semantic segmentation procedure involves machine learning, annotated training data is necessary. To address this need, this work provides annotation guidelines and semi-automated annotation procedures, along with two datasets comprising 2.8 billion annotated points and manually created reference BIMs. Additionally, efficient and robust filtering, downsampling and geometry reconstruction techniques have been either extended or newly developed. For both geometry reconstruction and open BIM authoring, this work introduces algorithms and procedures, including the two Python modules *pystruct3d* and *openbimxd*. Along with pipeline integration and evaluation, new metrics have been developed to provide a more realistic quantification of reconstruction accuracy. Finally, concluding remarks highlight perspectives for future research.

Zusammenfassung

Mit der zunehmenden Verbreitung von Building Information Modelling (BIM) in der Bauindustrie gewinnt auch die Anwendung in der Planung und Durchführung von Bauprojekten im Bestand immer mehr an Bedeutung. Um BIM in solchen Projekten einsetzen zu können, ist ein Bestandsmodell erforderlich. Während in der traditionellen Bestandsaufnahme Handmessungen und konventionelle Vermessungsgeräte zum Einsatz kommen, ermöglichen moderne Verfahren zur Bestandserfassung – insbesondere die Kombination aus terrestrischem Laserscanning und Structure from Motion (SfM) – die schnelle Erfassung von 3D-Daten ganzer Gebäude. Die 3D-Daten liegen als dichte Punktwolke vor. Da es sich bei Punktwolken um unstrukturierte Daten handelt, enthalten sie keine weiteren Informationen über das Gebäude. Deshalb ist es notwendig, diese Daten in BIM-Modelle zu überführen und relevante Informationen abzuleiten. Da eine manuelle Überführung zeitaufwendig ist, wird für die Anwendung von BIM in Bestandsprojekten eine automatisierte Überführung in BIM-Modelle angestrebt.

In dieser Arbeit wird daher eine Methode zur automatischen Überführung von Punktwolken in BIM-Modelle vorgestellt. Das *ScaleBIM*-Framework umfasst drei wesentliche Prozessschritte: (i) die semantische Segmentierung, bei der jedem Punkt eine semantische Klasse, typischerweise eine Bauteilklasse, zugewiesen wird; (ii) die geometrische Rekonstruktion, bei der regel- und typologiebasierte Verfahren eingeführt werden, um eine konsistente Rekonstruktion zu gewährleisten; und (iii) die Erstellung von BIM-Modellen in offenen Dateiformaten unter Nutzung von Open-Source-Tools, um Interoperabilität sicherzustellen. Das entwickelte Verfahren wird anhand von zwei Datensätzen mit unterschiedlichen Merkmalen getestet. Alle Prozessschritte sind in eine konsistente Pipeline integriert, die es ermöglicht, Wände, Türen und Stützen zu rekonstruieren und entsprechende BIM-Objekte zu erstellen.

Da für die semantische Segmentierung maschinelle Lernverfahren eingesetzt werden, sind annotierte Trainingsdaten erforderlich. Um die Daten effizient annotieren zu können, wurden Richtlinien zur Datenannotation entwickelt. Zusätzlich können halbautomatisierte Verfahren die Datenannotation unterstützen. Auf dieser Basis wurden 2,8 Milliarden Punkte annotiert. Manuell erstellte BIM-Modelle dienen als Referenz zur Evaluation der Rekonstruktionsgenauigkeit. Weitere Methoden umfassen effiziente und robuste Verfahren für die Datenvorverarbeitung, wie z. B. das Filtern, Bereinigen und Reduzieren der Punktdichte. Sowohl die Verfahren zur Vorprozessierung und geometrischen Rekonstruktion als auch zur Erstellung der BIM-Modelle wurden in den Python-Modulen *pystruct3d* und *openbimxd* implementiert und veröffentlicht. Die gesamte Pipeline ist basierend auf beiden Modulen implementiert. Zur Bewertung der Rekonstruktionsgenauigkeit wurden neue Metriken eingeführt, die eine genauere Beschreibung der Abweichungen ermöglichen als bisherige Metriken. Abschließend werden Forschungslücken identifiziert und die Ergebnisse diskutiert.

Acronyms

AI	Artificial Intelligence
14NH	14 Nothelfer Hospital, Weingarten, Germany
AEC	Architecture, Engineering and Construction
API	Application Programming Interface
BIM	Building Information Modelling
BIMs	Building Information Models
BrIMs	Bridge Information Models
CAD	Computer Aided Design
CNN	Convolutional Neural Network
CV4AEC	Computer Vision in the Built Environment
DFKI	German Research Centre for Artificial Intelligence
DoF	Degree of Freedom
DBSCAN	Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise
EU	European Union
FN	False Negative
FP	False Positive
GWP	Global Warming Potential
GPR	Ground penetrating radar
hobb	Horizontal Oriented Bounding Box
HYSAC	Hypothesis Based Sampling Consensus
ICU	Intensive Care Unit
IFC	Industry Foundation Classes
IoU	Intersection Over Union
kNN	k Nearest Neighbour
k-d tree	k-dimensional tree
LiDAR	Light Detection and Ranging
LLMs	Large Language Models
MEP	Mechanical, Electrical and Plumbing
MLESAC	Maximum Likelihood Estimation Sample Consensus
mIoU	Mean Intersection over Union
MLE	Maximum Likelihood Estimation
Open3D	Open3D: a modern library for 3D data processing
OPTICS	Ordering Points To Identify the Clustering Structure
PCA	Principal Component Analysis
RANSAC	Random Sample Consensus
RPTU	University of Kaiserlautern-Landau
S3DIS	Stanford 3D Semantics Dataset
SfM	Structure from Motion
SVD	Singular Value Decomposition

SVM	Support Vector Machine
TLS	Terrestrial Laser Scanners
TN	True Negative
TP	True Positive
UAV	Unmanned Aerial Vehicle
vIoU	Volumetric Intersection over Union

Terms and definitions

This section defines and explains the basic terminology and concepts used throughout this work. While understanding these terms provides a foundation for the entire text, readers may choose to skip this section and refer back to it as needed for clarification of specific terms and concepts.

Algorithms and pseudocode

```
Input  : some data                                              // the input
Result : some result                  // intermediate result, will not be returned
Output: some output                                  // returned by the algorithm

// This is a comment

// List and array notation
List[objects]                                              // One dimensional array
Array[Numbers]                                               // N-dimensional array

// defining a function
Function an awesome function(argument 1, argument 2):
    if some condition then comment for if condition
        do cool things
    end
    for object in objects do comment to explain what the loop does
        do something with the object
    end
    return whatever the function returns
an awesome function(Often, functions take arguments)                  // Function call
return This is what the algorithm returns
```

Algorithm 1: Notation for algorithms.

Pseudocode is commonly used to elucidate the functionality of algorithms. The notation employed is demonstrated in Algorithm 1. For complex algorithms, pseudocode will be supplemented with detailed textual descriptions. Simpler algorithms, which can be described unambiguously, will not be accompanied by pseudocode. In all instances, the pseudocode will be presented at a high level to facilitate general understanding, rather than detailed implementation. The detailed implementation can be found in the source code accompanying this work (refer to Annex 3). Additionally, many algorithms are available in open source libraries.

Boolean mask

Given an array of shape $m \times n$, a boolean mask can be created as an array of the same shape, containing only boolean values, i.e. true or false. A boolean mask is typically constructed by evaluating a condition at each position in the array, setting the value to true or false based on whether the condition is met. This mask can then be used to extract or delete specific values from the original array.

Bounding Box

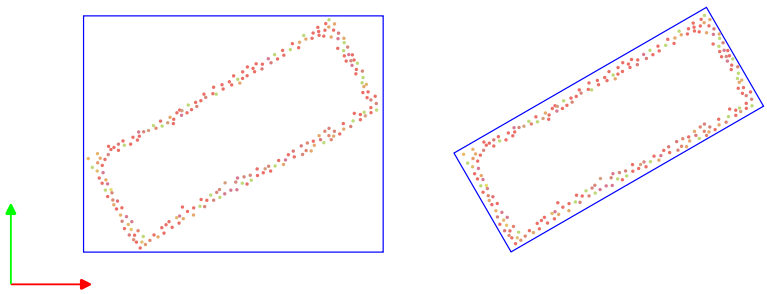


Figure 1: Main types of bounding boxes. Left: axis-aligned bounding box. Right oriented bounding box.

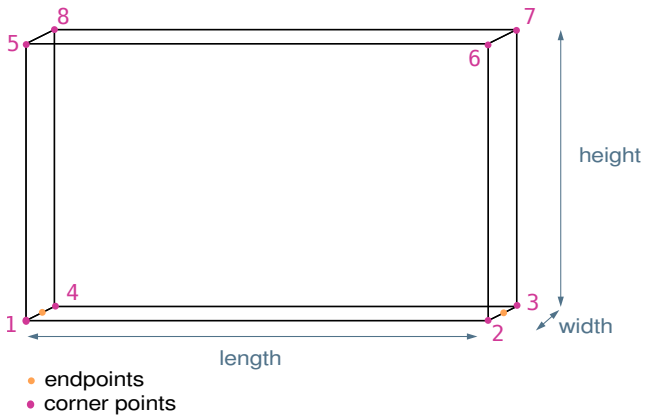


Figure 2: Definition of bounding box.

A bounding box can be defined as the cuboid enclosing a set of points. In bounding box fitting, two main concepts can be distinguished: axis-aligned and (minimum) oriented bounding boxes. A 2D example of each is presented in Figure 1. Note, that an oriented bounding box is sometimes referred to as the minimum bounding box and even minimum oriented bounding box is a common expression. Technically,

an oriented bounding box is an approximation of the minimum bounding box. Thus, the term oriented bounding box will be used in this work, as it is more descriptive and technically correct.

The following convention is used to define the bounding box and its dimensions (refer to Figure 2). The eight corner points are sorted in a counter-clockwise order from the bottom to the top, starting from the left-most point. All edges are perpendicular to each other. The length is the longer horizontal edge and the width is the smaller horizontal edge. The height is the vertical dimension of the bounding box. The endpoints of the bounding box are the centre axis points in the lower horizontal plane. In this work, the lower and upper surfaces are aligned horizontally, the other surfaces are vertical. This is referred to as a Horizontal Oriented Bounding Box (hobb) and is an approximation to the minimum bounding box. Beyond the dimensions and corner points, the endpoints, i.e. the points delimiting the lower baseline, are used during reconstruction to achieve various tasks.

Building Information Modelling

Building Information Modelling (BIM) is the technology used to digitally represent any physical asset in the Architecture, Engineering and Construction (AEC) domain. A model created according to the BIM methodology comprises a 3D representation of the object, which is associated with attributes and properties. Beyond semantic information, further domains include time, cost, quality, sustainability, safety, etc. These are referred to as other dimensions of BIM and, depending on the notation, time is considered as the fourth dimension (4D) and cost the fifth dimension (5D).

Eastman et al. [1] provide a widely accepted definition of BIM, identifying four main characteristics:

1. Building components represented digitally with 'computable graphic and data attributes [...] as well as parametric rules' for intelligent modification.

2. Data describing the behaviour of the components. This data can be used for analyses and related processes such as quantity take-offs and simulations.

3. 'Consistent and non-redundant data such that changes to component data are represented in all views of the component [...]'

4. 'Coordinated data', i.e. all views of the model share a common coordinate frame [1].

Hausknecht and Liebich [2] argue that the data exchange is another key element of BIM as an integrated approach to AEC project delivery. To this end, open data exchange standards such as Industry Foundation Classes (IFC) [3] have ever been a central aspect to consider and are one of the key technologies in this work.

The acronym 'BIM' is employed with various meanings across different contexts, necessitating a clear definition. In this study, 'BIM' specifically refers to the methodology and concepts utilized in generating respective digital models. When referring to a specific digital representation, the term 'BIM model' will be used. Conversely, 'BIMs' stands for *Building Information Models*, acknowledging that 'BIM' itself represents a singular methodology without a plural form. While this usage may diverge from strict

linguistic norms, it is widely accepted within the active community of professionals in the BIM field and will be adopted in this work.

Clustering

Clustering denotes a procedure that groups data according to its features, including spatial proximity, density and colour.

Colour coding of semantic labels

In the data, semantic labels are typically stored as integer values. To visualize this data, these integer labels are mapped to corresponding colours. The following table shows the label ID alongside the class name, the respective IFC class, RGB colour, and a field displaying each colour.

ID	Name	IFC class	R	G	B	Colour
1	slab	IfcSlab	170	0	0	
2	floor	IfcCovering	70	0	20	
3	ceiling	IfcCovering	70	0	120	
4	wall	IfcWall	0	0	170	
5	pipe segment horizontal	IfcPipeSegment	0	100	200	
6	fitting	IfcPipeFitting	0	150	150	
7	pipe segment vertical	IfcPipeSegment	0	200	100	
8	door	IfcDoor	100	0	200	
11	window	IfcWindow	200	0	100	
12	stair	IfcStair	160	30	130	
13	roof	IfcRoof	60	80	100	
14	column	IfcColumn	0	50	0	
15	beam	IfcBeam	0	170	0	
16	Truss	IfcMember	150	250	50	
17	chimney	IfcChimney	150	50	200	
18	railing	IfcRailing	140	40	30	
19	ramp	IfcRamp	250	150	150	
20	elevator	IfcTransportElement	200	150	250	
21	pavement	IfcPavement	50	200	250	
22	gravel pad	IfcGeographicElement	100	100	50	
23	vegetation	IfcGeographicElement	20	120	40	
25	space heater	IfcSpaceHeater	230	240	10	
26	light fixture	IfcLightFixture	90	130	180	
27	curtain wall	IfcCurtain Wall	225	0	250	

Table 1: Building category labels: ID, name, IFC class, RGB and colour.

The labels are organized into two categories. The *building* category includes all parts of the building and objects that are firmly attached to it (refer to Table 1). Note, that some IDs are not assigned due to former changes in the label definition and that some classes do not correspond to an IFC class.

Colour maps

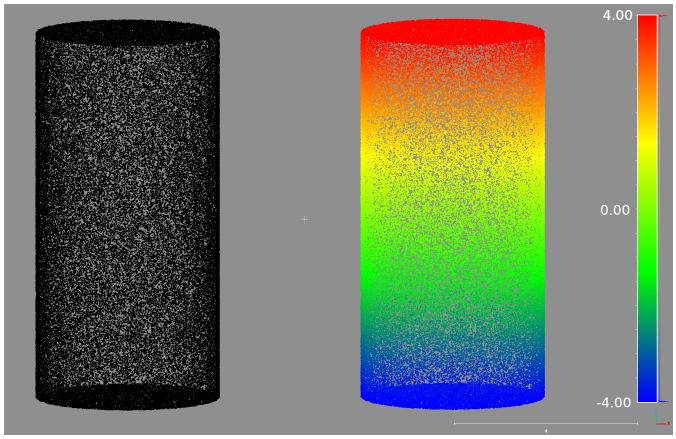


Figure 3: Colour rendering with blue - green - yellow - red colour map along the Z axis.

Beyond merely rendering label colours, a common application in 3D data visualization is the use of colour maps to represent additional data beyond the basic 3D point coordinates and colours. For example, colour maps can visualize secondary data such as intensity, height along the Z-axis, or point density. In this work, points without assigned colour values are rendered using a colour map. Otherwise, all points would appear in a uniform colour, making it difficult for the viewer to distinguish details.

Unfortunately, there is no universally accepted definition for such colour maps, as the colour rendering depends on both the range of values and the specific colour map template used. A commonly utilized colour map is *blue-green-yellow-red*, where the minimum value is represented in blue and the maximum value in red. A generic example created with CloudCompare [4] using this *blue-green-yellow-red* colour map can be seen in Figure 3.

Coordinate frame

Unless stated differently, the coordinate frame is defined according to Figure 4. The X and Y axes span the horizontal base plane, the Z axis pointing vertically upwards.

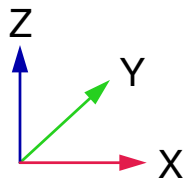


Figure 4: Coordinate frame definition and colour scheme.

Downsampling and Subsampling

Both the terms downsampling and subsampling describe procedures that extract a subset of the input data. To achieve this, different sampling strategies can be used:

- When applying a **random** sampling strategy, a subset of the input data is chosen by randomly selecting data points.
- A **spatial** sampling strategy extracts a subset of data points based on their spatial relations, e.g. the number of neighbours in a given radius.
- In **voxel downsampling**, a voxel grid is constructed. The mean of all points inside each voxel is then calculated as the subsampled point.

Industry Foundation classes and open BIM

One of the core principles of BIM data exchange is to use open standards such as Industry Foundation Classes (IFC) [3]. This standard is developed by Buildingsmart International and is constantly extended towards the infrastructure domain. In the context of BIM data exchange, open refers to using data exchange formats that are documented, published and openly accessible, i.e. any software can provide functions to import and export IFC data without any licensing costs. This principle needs to be distinguished from open source, which includes software whose source code is openly accessible and can be adopted and changed according to the licensing terms. Open source BIM authoring tools are used in the context of this work to interact with IFC data.

Point Cloud

A point cloud is a set of points in a n-dimensional Euclidean space, i.e. the coordinate values are associated with a metric unit such as metres. In the context of this work, a point cloud is referred to as a set of three-dimensional points covering a physical object thus representing it. Other values including colour, intensity, labels, etc. can be associated with each point. The point cloud is stored as a $n \times m$ matrix, where n is the number of points and m is the number of values including coordinates, colours, etc. Point clouds are unstructured, i.e. an unordered sequence of points. In such unstructured data, no

relationships of the data points can be derived from the order, e.g. Point at index i and the next point at index $i + 1$ are not necessarily neighbours.

To process point clouds efficiently, the data can be transformed into a structured tree representation which in turn allows for searching neighbours efficiently.

Primitive fitting and geometric segmentation

Geometric primitives are types of basic solids including planes, cuboids, cylinders, etc. Thus, a common procedure in scan-to-BIM pipelines is to fit such primitives to data. This process is also referred to as geometric segmentation, with points being assigned a class based on the primitive they are fitted to.

Scan-to-BIM

The term scan-to-BIM is used for a wide range of processes in the context of using point clouds to create BIM models. In the context of this work, scan-to-BIM refers to the fully automated process of transferring a point cloud into a BIM model.

Scores: Accuracy, Precision, Recall, F1 score, Intersection over Union

In the following, the most common scores including accuracy, precision, recall and F1 score will be introduced. These are primarily defined for binary classifiers and share some common definitions. A True Positive (TP) denotes a correct positive prediction, a True Negative (TN) a correct negative prediction, a False Positive (FP) a positive prediction with the actual true value being negative and the False Negative (FN) a negative prediction with the actual true value being positive. Despite the definition, this notation is used for multi-label classifiers in semantic segmentation, where a correctly predicted label would be considered a TP, and any other prediction is considered as a TN, etc.

Accuracy measures the overall correctness of the model by comparing the number of correct predictions (both true positives and true negatives) to the total number of predictions.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{\text{correct predictions}}{\text{all predictions}} \tag{0.1}$$

Precision, also known as Positive Predictive Value, measures the accuracy of the positive predictions by comparing the number of true positives to the total number of positive predictions. It answers the question: ”Of all the instances that were predicted as positive, how many were actually positive?”

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{\text{correct predictions}}{\text{all predictions}} \tag{0.2}$$

Recall, also known as Sensitivity or True Positive Rate, measures the ability of the model to correctly identify positive instances by comparing the number of true positives to the total number of actual positives. It answers the question: ”Of all the instances that were actually positive, how many were correctly predicted as positive?”

$$\text{Recall} = \frac{TP}{TP + FN} \tag{0.3}$$

The F1 Score is the harmonic mean of precision and recall, providing a balance between the two metrics. It is particularly useful when you need to account for both false positives and false negatives. It combines precision and recall into a single metric by calculating their harmonic mean, which gives a better measure of the incorrectly classified cases than the arithmetic mean.

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{0.4}$$

A widely employed metric for assessing segmentation accuracy is the Intersection Over Union (IoU). The IoU is commonly computed individually for each class, while the Mean Intersection over Union (mIoU) serves to evaluate the overall segmentation performance of a model.

The following equation illustrates how the IoU is calculated. In the equation, ’PRED’ denotes the predicted labels and ’GT’ denotes the ground truth labels.

$$IoU = \frac{PRED \cap GT}{PRED \cup GT} \tag{0.5}$$

Expressed differently, it is the TP divided by the sum of TP, FP and FN:

$$IoU = \frac{True\ Positives}{True\ Positives\ +\ False\ Positives\ +\ False\ Negatives} \tag{0.6}$$

The IoU metric has even been adopted to evaluate the result of scan-to-BIM. However, there are some drawbacks as this metric even penalizes small shifts of the reconstructed and ground truth geometry. A more detailed explanation and alternatives can be found in section 6.2.

Semantic segmentation

Methods that assign a semantic label to each data point in the set are referred to as semantic segmentation. Fundamentally, semantic segmentation can be performed on data of any dimension. However, the most common use cases in the context of scan-to-BIM are assigning labels to either images in 2D or point clouds in 3D. The semantic nature of the label is that it assigns a specific class to each point. In the context of building scenes these are typically building components or interior objects. This distinguishes semantic segmentation from other methods of segmentation that only assign a label denoting the cluster or primitive a point is assigned to.

Topology

The basic configuration of spaces, levels and components is referred to as the topology of a building. In the context of scan-to-BIM, a topologically correct reconstruction denotes the configuration of building components that fully enclose spaces and that are interconnected in a structurally correct manner. A topology-aware reconstruction procedure implements measures to enforce a topologically correct reconstruction.

Voxel

A voxel represents a 3D counterpart to a pixel, signifying a specific volume in limitless 3D space. Generally, voxels take a cubic shape. Converting 3D space into a grid of voxels provides a structured approach to representing 3D point clouds. This voxel grid facilitates various processes such as downsampling and noise reduction, among other applications.

1 Introduction

1.1 Motivation

Among the challenges politics and industry face, climate change [5] is one of the most demanding and yet most complex. Not only does climate change affect all domains of human life on earth, it also has the potential to threaten the foundations of human life, including local and global climate, extreme weather phenomena and ecosystems.

One of the sectors that accounts for a considerable amount of emissions is the construction industry itself. This industry involves the complete value chain of erecting buildings, including the production of building materials, construction, demolition, etc. and accounts for 12% of Global Warming Potential (GWP) in Germany [6].

Considering European Union (EU) legislation aims for a 55% reduction of greenhouse gases by 2030 and a carbon neutral economy and society by 2050 under the Green Deal [7], scenarios have been elaborated that deduce the annual reduction path per industrial sector [8]. Following the scenarios towards a carbon-neutral European industry by 2050, the construction industry as part of the sector 'industry' needs to significantly reduce its GWP emissions in the coming years (see Figure 1.1).

The implications of the scenarios are that beyond reducing the energy consumption of buildings, the emissions of the construction itself need to be reduced.

Among the production of building materials, concrete contributes about 8% to global emissions [9]. About two third of the overall GWP emissions in cement production are related to the chemical reactions themselves. These cannot be reduced by using alternative fuels or renewable energies. Hence, concrete structures need specific attention to reduce the emissions in the construction sector.

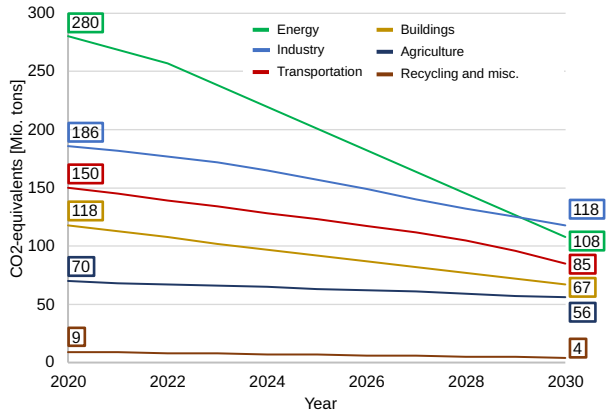


Figure 1.1: GHG emission reduction paths for industrial sectors [8].

Indeed, there is potential to save emissions using cements with reduced

GWP emissions as proposed in [6]. However, net-zero emissions in cement production are a long-term goal that one day may be realized. In the meantime, a considerable reduction needs to be leveraged by other means. A good option is to preserve structural components including concrete, thus avoiding the consumption of cement.

In a study co-authored by the author [10], three main principles for preserving structural components were identified: *Reuse, Rebuild, Recycle*.

In the *Reuse* scenario, the building or at least its structure will be entirely preserved, altered and repaired if necessary. Both the shell and the interiors can be exchanged to either increase thermal insulation or to comply with changed user requirements. Reusing is both the most intuitive and effective way, as almost all emissions for structural components can be avoided. However, it can only be considered if the structure is repairable, the location of the building is still appropriate, and the function is compliant with the structure.

If the existing building cannot remain at the current location, or a new building is needed at a different location, the entire structure cannot be preserved. Reclaiming components of existing structures to reassemble a new structure is then the second-best option referred to as *Rebuilding*. This involves assessing the existing structure, analysing joints and connections, identifying components to be reclaimed, and design, disassembly and reassembly. Although joints and connections might require repair and adjustments, most of the components can potentially be preserved, thus avoiding GWP emissions. In the *Rebuild* value chain, the most crucial step to avoid extra emissions is the transport of the reclaimed components, which should be minimized.

In this context, *Recycling* includes the demolition of a structure and reclaiming secondary raw materials, including recycled concrete aggregates. Unfortunately, producing new concrete with recycled aggregates involves cement, which in turn accounts for GWP emissions. Therefore, *Recycling* should be the last option.

A thorough assessment of the existing structure is critical to *Reusing*, *Rebuilding* and *Recycling*. For the assessment and subsequent design processes a digital representation of the structure needs to be acquired. To capture spatial data, efficient mobile and terrestrial systems fusing laser scanning and 3D image reconstruction can be used to deliver 3D data of the buildings. To use such data in design, construction and facility management it needs to be transferred to semantic models according to the Building Information Modelling (BIM) methodology. BIMs can be used for multiple use cases in early and later design stages, e.g. assessing the potential for reusing, rebuilding and recycling, design processes to reuse existing structures or to rebuild using reclaimed components, assess quantities for recycling, etc.

With respect to the amount of data that needs to be processed and the steps necessary to transfer raw 3D data into BIM models the full automation of these processes is desirable. Hence, this thesis will present an automated approach to transfer point clouds to BIMs. This will go beyond 'yet another scan-to-BIM approach', of which many exist as the literature study will reveal. This work will contribute a comprehensive framework including data annotation for semantic segmentation, two modules with basic reconstruction and BIM object creation algorithms, the integration of the algorithms into a comprehensive pipeline, experiments and metrics to evaluate the framework, finally concluding on lessons learned and directions of future research.

Beyond *Reuse, Rebuild, Recycle*, BIMs of existing buildings can be utilized to support numerous use cases including quantity take-off, tendering, re-design, construction progress tracking, safety monitoring and facility management. The variety of possible applications of automatically created models supports the motivation of this work.

1.2 Objective

Starting from 2015, the adoption of BIM gained some pace in the German construction sector, initiated by political guidelines published at that time [11]. Further to this, the fields of application and potential use cases of BIM were being discussed among the industry and research institutions [12, 13, 14]. Along with these discussions and standardization efforts, the application of BIM to support technical assessment, re-design, operation and maintenance of existing structures gained more interest.

Naturally, BIMs are typically not available for existing structures. Thus, it seemed natural to create BIMs out of existing structures. This procedure is referred to as *scan-to-BIM*, as 3D data as a basis to create the semantic models is 'scanned', i.e. acquired using reality capturing devices. However, scan-to-BIM procedures were and still are far from being automated, and a manual process partially supported by software workflows.

Regarding the technology level, significant advances have been witnessed in recent years, namely in the fields of semantic segmentation, geometry processing algorithms and automating scan-to-BIM itself. Fusing the most significant advances in all three fields can leverage the automation of scan-to-BIM procedures. Thus, the main contribution of this work will be to provide a comprehensive framework for scan-to-BIM automation. The specific contributions can be outlined as follows:

- To train semantic segmentation models, manually annotated training data is required. Thus, this work will contribute to this field with comprehensive data annotation guidelines, a workflow to support this manual process and a dataset to leverage semantic segmentation.
- In a joint effort with the Department of Augmented Vision at the German Research Centre for Artificial Intelligence (DFKI)[1] and as a part of the HumanTech project[2], a semantic segmentation model was trained using the previously annotated dataset.
- Several algorithms useful for geometric reconstruction have been developed and implemented to offer the foundation for comprehensive reconstruction pipelines.
- Likewise, a module was developed to combine the previously reconstructed geometry with semantic information and to create BIM objects using an open data standard.
- The procedures are combined to an end-to-end pipeline, which is extensively tested.
- Beyond the well-known evaluation metrics, enhanced metrics have been developed and implemented. Along with ground truth BIMs and aligned point clouds, these metrics serve to assess

[1] https://av.dfki.de/
[2] https://humantech-horizon.eu/

the quality of the reconstruction.

- Further to the results being discussed, directions for future research are suggested as a conclusion.

The contributions have been published in several scholarly papers, which are listed below:

Christian Glock et al. “Massivbau in Zeiten von Klimawandel und Ressourcenverknappung – Herausforderungen und Lösungsansätze/Concrete construction in times of climate change and resource shortage – challenges and solutions”. In: *Bauingenieur* 97.01-02 (2022), pp. 1–12. ISSN: 0005-6650. DOI: 10.37544/0005-6650-2022-01-02-33

Christian Glock et al. “Treibhausgas- und ressourcenreduzierter (Beton)Bau: Herausforderungen, Lösungsansätze, Anreizsysteme”. In: *DBV-Heft 50 Nachhaltiges Bauen mit Beton: Band 1*. Ed. by Lars Meyer. Berlin, 2022

Christian Glock et al. “Klima– und ressourcenschonendes Bauen mit Beton”. In: *BetonKalender 2024*. Ed. by Konrad Bergmeister, Frank Fingerloos, and Johann–Dietrich Wörner. Wiley, 2023, pp. 177–265. ISBN: 9783433034064. DOI: 10.1002/9783433611494.ch2

Fabian Kaufmann, Christian Glock, and Thomas Tschickardt. “ScaleBIM: Introducing a scalable modular framework to transfer point clouds into semantically rich building information models”. In: *Proceedings of the 2022 European Conference on Computing in Construction*. Computing in Construction. University of Turin, 2022. DOI: 10.35490/EC3.2022.194 *This paper was issued the Best Paper Award at the European Conference on Computing in Construction 2022*

Fabian Kaufmann et al. “Ontology-based semantic labeling for RGB-D and point cloud datasets”. In: *Proceedings of the 2023 European Conference on Computing in Construction and the 40th International CIB W78 Conference*. Computing in Construction. European Council for Computing in Construction, 2023. DOI: 10.35490/EC3.2023.241

Marius Schellen et al. “Annotation rules and classes for semantic segmentation of point clouds for digitalization of existing bridge structures”. In: *Proceedings of the 2023 European Conference on Computing in Construction and the 40th International CIB W78 Conference*. Computing in Construction. European Council for Computing in Construction, 2023. DOI: 10.35490/EC3.2023.228

F. Kaufmann, C. Glock, and T. Tschickardt. “Drone-based acquisition of as-built models for the automation of processes within the digital management of bridge assets”. In: *The 8th International Symposium on Reliability Engineering and Risk Management* (2022), pp. 91–98. URL: https://rpsonline.com.sg/rps2prod/isrerm2022/epro/html/toc.html

Thomas Tschickardt, Fabian Kaufmann, and Christian Glock. “Lean and BIM based flight planning for automated data acquisition of bridge structures with LiDAR UAV during construction phase”. In: *Proceedings of the 2022 European Conference on Computing in*

Construction. Computing in Construction. University of Turin, 2022. DOI: 10.35490/EC3.2022.146

The first of the two papers mentioned above outline the potential of digital methods to reduce GWP emissions in the construction sector with a special focus on concrete construction. The second two papers are directly related to this work and the contributions as outlined above. The last three papers link scan-to-BIM to drone-based acquisition procedures.

The *ScaleBIM* framework will be enhanced in this work beyond the paper published by the author in 2022 [16]. The framework contains an end-to-end processing pipeline from the aligned input point cloud to the BIM model. This comprises (i) semantic segmentation of the point clouds, i.e. detecting objects, (ii) instance segmentation and geometric feature extraction and (iii) consolidating the information and delivering BIM objects, thus forming a BIM model. The models will comprise the following information:

- The 3D geometry of the object represented according to BIM standards for geometric representations.
- Semantic information about the type of object, i.e. the object class such as wall, door, column, etc.

The framework is tailored to process data of multi-storey buildings, e.g. office, hospital or residential buildings and demonstrates how to reconstruct walls, doors and columns. Exterior and interior walls are the typical objects that enclose spaces. Doors are a good example of child opening objects in parent components such as walls. Thus, the method applied to reconstruct doors can be applied to windows and generally openings with minimal adjustments. The same applies for the reconstruction of columns and the approach hereby can be easily applied to beams and other structural components. Beyond that, with the *ScaleBIM* framework a comprehensive library named *pystruct3d* will be published that supports 3D reconstruction pipelines with advanced algorithms. *pystruct3d* can be used to implement reconstruction methods for other objects.

After retrieving the geometric features, BIM objects need to be created. The goal here is to use open BIM standards wherever possible to facilitate an open data exchange. This will help to integrate this work into the vivid community for open data exchange in the construction industry. As part of *ScaleBIM*, a library named *openbimxd* is delivered supporting BIM model creation, filtering and update.

In addition to delivering the necessary modules and algorithms, a primary objective is to seamlessly integrate these algorithms and methods into a cohesive pipeline for transforming point clouds into BIM models. This integration focuses on optimizing the entire process, incorporating advanced algorithms to enhance reconstruction accuracy and efficiency. The framework will be meticulously developed, theoretically elucidated, implemented, and rigorously tested using both established benchmark datasets and proprietary data sources.

1.3 Methodology

The basis for this work is an aligned and complete 3D point cloud of the object with a sufficient density. For parts of the framework such as semantic segmentation, colour values are mandatory to distinguish objects reliably. *ScaleBIM* is developed *sensor-agnostic*, i.e. point clouds captured with various sensors can be processed. However, the raw scans need to be aligned so that all scans share a common coordinate frame. Depending on the purpose of the BIM model created, translation into a global coordinate frame may be required, e.g. to compare the acquired model with an existing model or to generally use it in conjunction with other data or BIMs. Typically, the referencing is achieved using conventional surveying techniques providing global coordinates of markers placed before the scanning. The data acquisition and alignment itself is not part of this work.

Considering the accuracy of the input data, we assume that the acquired point cloud accurately represents the physical objects. Clearly, this might not apply as absolute measurement errors and inaccuracies during the alignment can influence the overall accuracy. Nevertheless, the input point clouds are even used to evaluate the resulting BIMs. Therefore, errors resulting from the data acquisition are excluded from the evaluation, thus presenting unambiguous evaluations of the results.

It is commonly observed that acquired point clouds of buildings are incomplete to a certain extent. The incompleteness depends on the scanning technology applied and the effort spent to acquire a point cloud as complete as possible. However, incompleteness occurs and results in:

- Partially scanned objects, i.e. only the surfaces visible to the scanner have been captured.
- Limitations in range of the acquisition system.
- Limited accessibility, e.g. of outside surfaces at high elevation.
- Objects occluding others, resulting in missing parts of objects' surfaces.
- Partially missing data due to over exposure to natural light, shadows or insufficient lighting conditions.
- Translucent objects, e.g. glass panes not reflecting light waves.
- Highly reflective or mirror surfaces resulting in scanning artefacts.

In *ScaleBIM*, methods developed are tailored to deal with incomplete data, thus achieving the most accurate reconstruction possible. If these methods still fail, predefined parameters are applied during the reconstruction, e.g. a predefined wall thickness in case only one vertical surface of a wall is present in the data. Likewise, it must be expected, that during the processing, data points can get lost, e.g. due to false classification in the semantic or geometric segmentation procedures applied. Thus, strategies will be implemented and evaluated to handle incomplete segmentation, still providing an accurate and complete BIM reconstruction.

During the reconstruction, the Manhattan world assumption is applied [21]. In a Manhattan world scenario, any object is aligned with a Cartesian coordinate system. Considering typical buildings, this

accounts for a vast majority of the objects observed. To apply the Manhattan assumption, the data must be axis-aligned, which can be achieved manually with little effort. The transformation can be applied inversely to transform the reconstructed model back to a global coordinate frame. In turn, the Manhattan world assumption [21] lets us leverage *geometry hypotheses*, thus reducing the degrees of freedom of reconstruction problems.

Based on the constraints and assumptions that have been described, a framework for automated scan-to-BIM pipelines is developed. This framework includes semantic and instance segmentation, geometry reconstruction and BIM model creation (refer to (I-V) in Figure 1.2).

Semantic segmentation is referred to as the method to assign a class label to every point in the point cloud. In the context of this work, a building component label is assigned. The semantic segmentation model was developed jointly in the HumanTech project[3] by Rambach, Chamseddine and Kaufmann [22]. This work contributes to the semantic segmentation by providing a comprehensive annotation guideline that introduces civil engineering expertise into data annotation. This leads to technically sound annotated data compliant with the structure of BIMs along with data annotated according to these guidelines.

The next step is instance segmentation, i.e. segmenting the set of points of one class into individual objects (see Figure 1.2 (III)). Hereby, point axis sorting, clustering and primitive fitting algorithms are applied to cluster, segment and group per-object points. As a side effect, the methods apply a filtering to the instances, thus improving the accuracy of the reconstruction.

The backbone of geometric reconstruction is a tailored bounding box fitting algorithm to retrieve the transformation, translation and rotation based on the previously segmented instances (Figure 1.2 (IV)). By applying *geometry hypotheses* the algorithm can be reduced to a 2.5D problem, i.e. the boundaries can be reconstructed from a 2D projection of the points. Before applying the bounding box reconstruction, the shape of the objects is determined. For occurrences of round shaped objects, a robust cylinder primitive fitting algorithm is applied. Some refinement procedures may be applied to increase the quality of the reconstruction, including modifying objects when exceeding dimension limits and applying topology correction to close gaps between adjacent perpendicular objects or to extend / clip objects that share the same centre line.

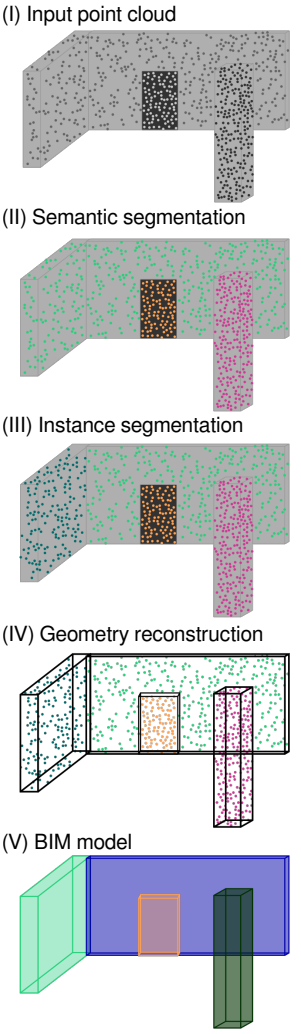


Figure 1.2: Overview.

From the refined reconstruction, all parameters to create BIM objects can be retrieved. The objects are created in the open IFC format [3], thus maximising interoperability of the reconstructed models. To create IFC objects, BlenderBIM and

[3] HumanTech is funded by the European Union under Grant Agreement No. 101058236

IfcOpenShell [23] were used, which offer comprehensive libraries for manipulating IFC data. Being open source and an open ecosystem, these libraries allow for future scaling of the method proposed here.

This work is structured as follows (refer to Figure 1.3). After this introduction, the state of the art and related work are discussed, and research gaps are identified (see chapter 2). The semantic segmentation procedure as one of the first steps in the *ScaleBIM* pipeline is discussed in chapter 3. After semantic segmentation, the proposed approach to geometric feature extraction, i.e. to identify the dimension, orientation, translation and transformation of geometric objects in the data, is described in chapter 4. In this chapter, the theoretical background is covered and enhanced methods will be proposed. The extracted information is combined into a BIM model based on open source tools and open data exchange standards as described in chapter 5. The procedures are integrated into a comprehensive pipeline using the developed and published libraries *pystruct3d* and *openbimxd*. The pipeline is tested on several scenes in various experiments. The results as well as the metrics used for evaluation are presented in chapter 6. Finally, concluding remarks and possible directions for future research are presented in chapter 7.

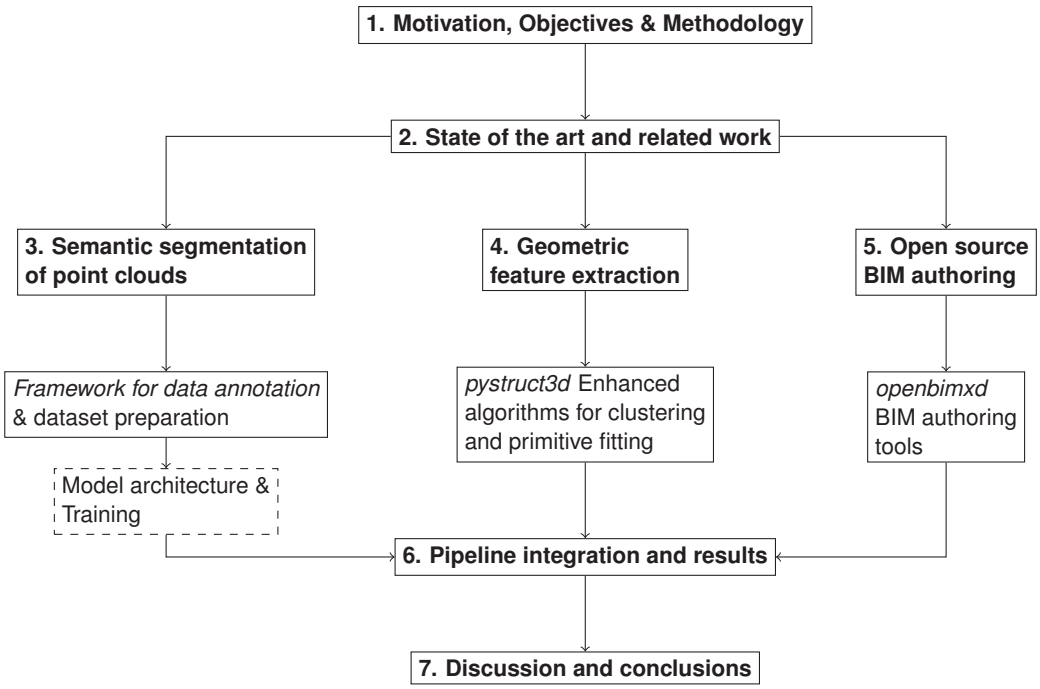


Figure 1.3: Structure of the proposed work including *key contributions*. Chapters are *bold faced*, libraries *italic*. Dashed items are not part of this work but will be integrated into the pipeline.

2 State of the art and related work

2.1 Introduction

As scan-to-BIM is a widely researched topic, numerous publications reflect advances of the state-of-the-art in this field. To provide useful insights and a comprehensive overview on the state-of-the-art, the literature reviewed here will be aligned with the scope of this work as described in section 1.2. This requires to introduce specific focus fields, thus excluding other topics less relevant. A list of topics positively considered will still be presented at the end of this introduction to let the reader understand the scope of this literature study.

At the initial stage of defining a specific focus, contributions related to the comparison of a point cloud with an existing BIM model, and the subsequent update of this model, are excluded. This process, known as scan vs. BIM, is nonetheless pertinent to a wide range of applications beyond the scope of this work, such as construction progress monitoring and construction safety monitoring.

Scan-to-BIM can be applied to a wide range of structures. Following the scope of this work, the literature overview will be limited here to residential, office and hospital buildings of a multi-storey topology and more general contributions applicable to such buildings. Detached houses as well as high-rise or buildings with a unique character such as historic buildings will not be included. It must be noted that under scan-to-BIM, many more types of structures are considered including urban scene [24, 25] reconstruction focusing on the volumetric reconstruction of buildings in large urban area point clouds and scan-to-BIM approaches to transfer point clouds of bridges into Bridge Information Models (BrIMs) as discussed in [26] or [27]. These will not be included in the literature study either.

Performing scan-to-BIM of industrial buildings requires specific approaches as, besides the structural building components, elements such as pipes, ducts, tanks and plants will be observed inside the building, potentially obscuring most of the structure itself. Such contributions will be only discussed here if structural building components are also being reconstructed. In particular, steel component reconstruction is not within the scope of this work and will thus be excluded from the literature overview.

Similar to the type of structure, the literature study will not cover approaches exclusively focusing on the geometric and semantic reconstruction of spatial elements such as rooms. However, a spatial reconstruction can be part of methods to reconstruct structural components.

The categorization of existing literature extends beyond merely identifying the structure and elements for reconstruction. As illustrated in Figure 2.1, the literature can be further segmented into two key classifications: *data source* and *scope and algorithms*. The *data source* category encompasses the input data, such as point clouds, images, drawings and textual information. On the other hand, the *scope and algorithms* category differentiates the scholarly contributions based on their specific roles in

the scan-to-BIM process. This includes, for instance, the techniques used for semantic segmentation, the strategies for geometry reconstruction (which involve clustering and geometric segmentation), and the design of partial or comprehensive automated systems for scan-to-BIM implementation.

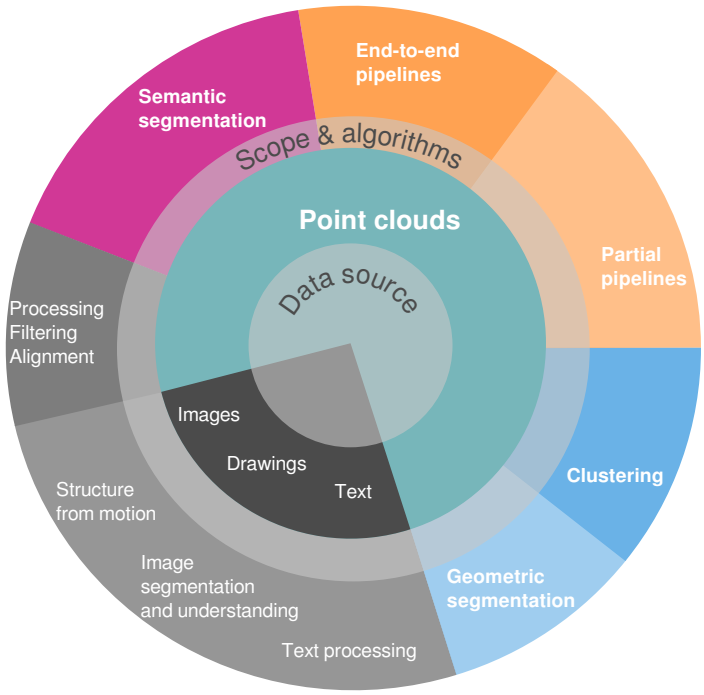


Figure 2.1: Overview on scan-to-BIM. Coloured topics will be considered in this literature study, grey scale categories are excluded as being not relevant to the objective and scope of this work.

A variety of data types, including images, drawings and text pertinent to an extant structure, can facilitate the creation of BIMs. This research specifically concentrates on point clouds as a primary data source for scan-to-BIM, hence it will exclusively address works that employ point clouds for this purpose. Schönfelder et al. conducted a review of 95 scholarly articles, revealing that point clouds were the predominant input source in nearly 70% of these studies [28]. Consequently, point clouds stand out as the most significant input medium in the field of scan-to-BIM.

The methodology of this literature review aligns with the strategy depicted in Figure 2.2. Initially, pertinent studies are selected from the 95 papers reviewed by Schönfelder et al., [28] with a focus on their use of data sources, applied methods, and the nature of information extracted. Notably, Schönfelder et al.'s review [28] is limited to machine learning based methods and papers published from 2017 to mid-2022. To expand the scope, this literature review will also incorporate knowledge-driven approaches and papers published outside the 2017 to mid-2022 window when they are identified as relevant. The identification of these additional contributions will be conducted through database and citation searches, and they will be systematically categorised into the established common categories.

In terms of methods and algorithms used to process point clouds, this literature study will be focused on

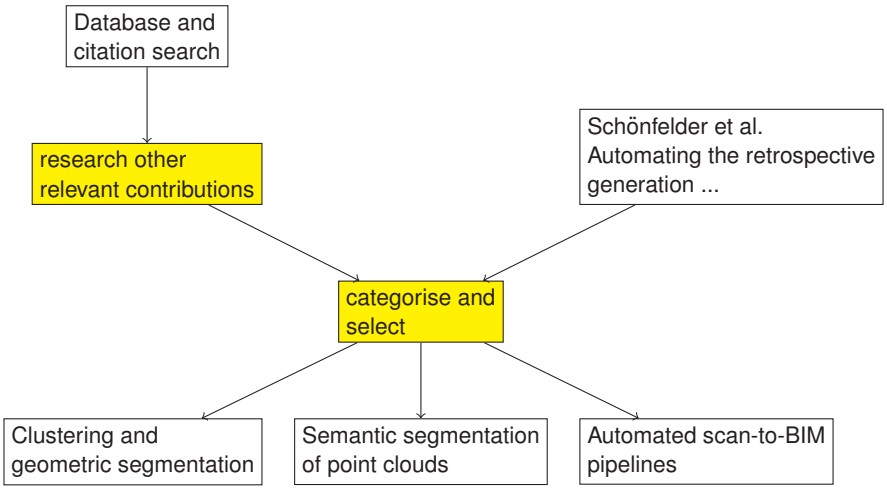


Figure 2.2: State-of-the-art methodology.

contributions describing approaches to semantic segmentation, clustering and geometric segmentation as well as automated pipelines in respective sections. Although the point cloud alignment, filtering and processing are vital steps to provide accurate input point clouds, literature in this field will not be reviewed here as these processes are not in the scope of this work (see section 1.2).

Along with the objectives of this work, in the review of the state-of-the-art of semantic segmentation mainly papers are considered that propose methods for the semantic segmentation of structural components or other elements affecting the structural behaviour of the building. Although other component classes may be included, studies that exclusively focus on non-structural components, e.g. indoor furniture, are excluded. Contributions to instance segmentation are excluded, as the approach proposed in this work only encompasses semantic segmentation. Another emerging field of research is the application of synthetic training data in semantic segmentation. Synthetic training data is data that was generated by a virtual model of the scene, typically a BIM model in construction related fields. Although promising approaches have been proposed, e.g. in [29] and [30], such approaches will not be considered as this thesis only uses real-world datasets to train the semantic segmentation algorithm applied.

As will be argued, in the fields of clustering and primitive fitting specific algorithms and approaches are widely adopted and have set a 'quasi-standard'. Thus, the review of contributions in this field will be less exhaustive compared to the other two main categories yet will cover the most widely used algorithms and present recent contributions that apply Machine Learning to both problems.

Automated pipelines cover scholarly papers that propose a stack of methods and algorithms to transfer an aligned point cloud into a BIM model including semantic information. In this context, studies on end-to-end pipelines as well as pipelines automating parts of the process will be considered. Hereby, layout plan generation is considered as an intermediate step in 3D BIM creation. Contributions focusing on the generation of specific models, e.g. non-geometric models for the facility management or for numeric simulations will also be excluded from this overview. In this category, both end-to-end pipelines covering

all steps from the input point cloud to the BIM model and partial pipelines only proposing one or several steps will be distinguished and analysed.

The main topics and the scope of this literature review can be summarized as follows. **Semantic segmentation**: data driven (mostly machine learning) and knowledge driven approaches to the semantic segmentation of point clouds of buildings and interior scenes focusing on structural components and using real-world training data in a supervised manner. **Clustering and primitive fitting**: widely adopted algorithms with an outlook to recent contributions applying machine learning to these problems. **Automated scan-to-BIM pipelines**: scholarly papers encompassing an end-to-end process to transfer point clouds to BIMs, or covering one or several steps of the process.

2.2 Semantic segmentation of point clouds

Data driven and knowledge driven methods can be identified as the two major approaches to semantic segmentation.

The emerging field of data-driven approaches covers all methods and algorithms that learn from large portions of data fed through the algorithm in the training step. This typically involves machine learning and deep learning techniques such as neural networks, decision trees, random forests, Support Vector Machines (SVMs), etc. In this work, we will focus on supervised learning techniques that require manually annotated data, with manual data annotation being laborious, expensive and error-prone. The training itself can be computationally expensive and tends to be a *black box*, with only limited possibility to examine reasons for poor or good results.

On the other hand, knowledge driven approaches typically use a combination of algorithms to extract features from the data. A multitude of implied heuristics and rules are then used to derive a class label from the observed features. Knowledge-driven approaches often perform well but do not generalize well to other datasets they were not developed for. Another challenge in knowledge-driven approaches is to set the input parameters correctly, which is challenging on unseen data.

This section is structured as follows. First, data-driven approaches are discussed. Introducing benchmark datasets will be accompanied by an initial overview exploring the field of semantic segmentation for scan-to-BIM based on the study of Schönfelder et al. [28]. This initial overview is extended with other relevant contributions. Second, knowledge-driven approaches are examined. Finally, the last subsection concludes on the technical aspects, identifies research gaps and provides a rationale for the approach chosen in this work.

2.2.1 Data-driven approaches

Benchmark datasets

Since data-driven approaches following the supervised learning paradigm require large manually annotated datasets, it is common practice to evaluate newly developed methods on benchmark datasets. In

the field of building point cloud segmentation, the most frequently used benchmark datasets are ScanNet by Dai et al. [31] and the Stanford 3D Semantics Dataset (S3DIS) by Armeni et al. [32]. Given their widespread use in many of the studies discussed below, it is worthwhile to briefly introduce both datasets at the outset.

The ScanNet benchmark [31] comprises 1,513 scans of single rooms along with 2.5 million RGB-D images from which the scenes were reconstructed. To acquire the data, the Structure[1] sensor was used. This low-cost device operates in conjunction with an iPad. As shown on the left side of Figure 2.3, the quality of the reconstruction is limited. The semantic annotation was performed using a crowdsourcing approach with a semi-automated annotation pipeline, comprising 20 semantic classes (floor, wall, chair, sofa, table, door, cabinet, bed, desk, toilet, sink, window, picture, bookshelf, curtain, shower curtain, counter, refrigerator, bathtub, other furniture) for the 3D semantic segmentation task.

The S3DIS dataset, introduced by Armeni et al. [32], comprises point clouds of five areas of different styles, including office, educational, and exhibition spaces with various room types. The data was captured using a Matterport[2] scanner. Annotations are provided for seven structural elements (ceiling, floor, wall, beam, column, window, door) and five common interior objects (table, chair, sofa, bookcase, board). Compared to the ScanNet dataset [31], the S3DIS dataset offers larger spaces instead of individual room scans, providing the opportunity to learn topology-related features in larger environments, such as spatial relationships and the objects enclosing these spaces. The quality of the scans is remarkably better due to the use of a more professional sensor.

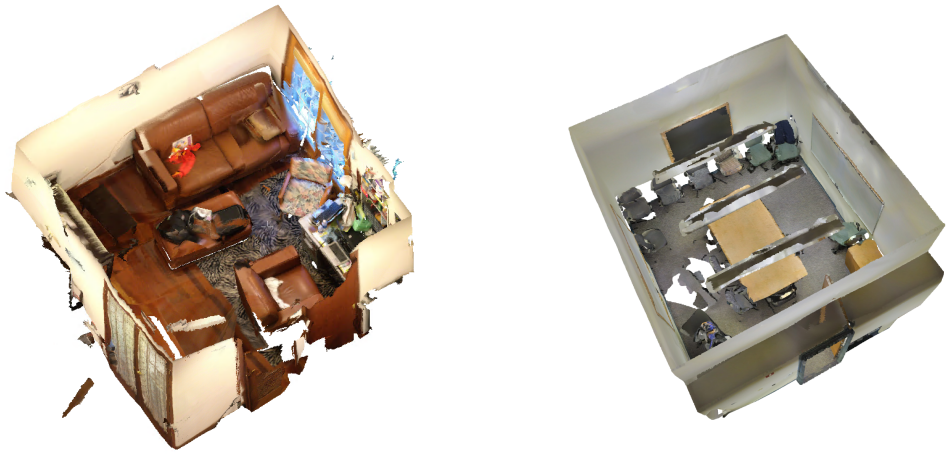


Figure 2.3: Exemplary scenes of benchmark datasets. Left: Indoor scene of the ScanNet dataset [31]. It can be observed that the captured mesh suffers from occlusions, noise and clutter and generally a poor accuracy. Right: Indoor scene of the S3DIS [32] dataset. The quality is much higher and less noise is present in the data. Occlusions are encountered similar to the ScanNet scene.

Comparing both datasets, the S3DIS [32] is better aligned with the scope of this work, as it offers larger areas of buildings instead of single rooms. Additionally, the provided labels are more focused on the structural components of buildings.

[1] In the paper, the type of sensor is not specified. Refer to `https://structure.io/` for information on currently available models.
[2] In the paper, the type of sensor is not specified. Refer to `https://matterport.com/` for information on currently available models.

Exploring semantic segmentation for scan-to-BIM

An initial overview on related studies on semantic segmentation relevant to scan-to-BIM can be derived from Schönfelder et al. [28]. Such papers are selected and presented in Table 2.1. Note that, according to the methodology description, only papers addressing scan-to-BIM were researched and discussed, and that the literature review is limited to a time window from 2017 to 2022.

Author	Year	Title	Short Description
Bassier et al. [33]	2019	Classification of sensor independent point cloud data of building objects using random forests	Planar patch segmentation using region growing on voxel octree representation of the point cloud and patch classification using random forests.
Dai et al. [34]	2018	ScanComplete: Large-scale Scene Completion and Semantic Segmentation for 3D scans	Volumetric autoregressive network for scan completion and semantic label prediction.
Koo et al. [35]	2021	Automatic classification of wall and door BIM element subtypes using 3D geometric deep neural networks	PointNet and MVCNN are used to classify subtypes of doors and walls.
Perez-Perez et al. [36]	2021	Segmentation of point clouds via joint semantic and geometric features for 3D modelling of the built environment	Point cloud segmentation using the region growing algorithm proposed in [37]. Semantic and geometric label inference using SVM and AdaBoost classifier, respectively. Conditional random fields to enforce coherence of geometric and semantic label, and Markov random field to predict the final semantic label.
Perez-Perez et al. [38]	2021	Scan2BIM-NET: Deep Learning Method for Segmentation of Point Clouds for Scan-to-BIM	Semantic segmentation and geometric label inference using two parallel CNNs. Ablation study on the effect of processing different sets of features. Semantic labelling accuracy 84.69%, geometric label accuracy 88.44%.
Runceanu et al. [39]	2018	Indoor mesh classification for BIM	Similar to [33]. Region growing mesh segmentation and random forest classifier to predict wall, floor and ceiling labels on the ScanNet benchmark dataset [31].
Yin et al. [40]	2021	Automated semantic segmentation of industrial point clouds using ResPointNet++	Adding deep residual learning to PointNet++ [41] for semantic segmentation of industrial buildings.

Table 2.1: Papers from the literature review of Schönfelder et al. categorised under semantic segmentation. The literature review is limited to the years 2017 to 2022 [28].

Bassier et al. [33] propose a two-step semantic segmentation approach. First, a planar patch segmentation using an octree-based region growing algorithm is performed and the semantic label is added through a random forest classifier. This approach is particularly interesting as the entire set of points is reduced to planes and a set of features for classification, thus reducing the amount of data processed in the classification step.

Runceanu et al. [39] propose a similar approach as Bassier et al. [33] applying a region growing mesh patch segmentation followed by a random forest classifier predicting the semantic labels wall, floor and ceiling. As Runceanu et al. [39] point out, the first version of the ScanNet benchmark dataset [31] was used and the work is limited to three classes.

Perez-Perez et al. [38] propose an end-to-end pipeline for semantic segmentation and geometric label prediction. Beyond the use of XYZ and RGB, the effect of processing other features such as normals, curvature, roughness and point to centroid distance is investigated. To this end, better results are observed in the ablation study when more features are processed. The architecture combines two parallel CNNs, one for semantic segmentation and the other for geometric label assignment. Both predictions are cross-validated in the final layers to provide the final inference. Coherence rules are applied, e.g. to ensure that the semantic label *wall* coheres to the geometric label *vertical*.

In an additional study, Perez-Perez et al. [36] use a pipeline of segmentation algorithms to perform a geometric and semantic label prediction. In the proposed methodology, the semantic labels are inferred using Markov random fields based on the coherence of semantic and geometric labels established by conditional random fields. The initial semantic and geometric labels have been predicted by a SVM and AdaBoost classifier, respectively. Compared to end-to-end semantic segmentation methods it can be noted that the chain of classifiers applied results in a more explainable behaviour of the method. Thus, Perez-Perez et al. estimate the individual distributions of each step while explaining that the method faces the limitation of not labelling clutter.

Besides semantic segmentation, a frequently encountered problem is incomplete scans, i.e. only parts of the objects are present in the data. To deal with such issues, Dai et al. [34] have proposed a combined methodology for completing scans and semantic segmentation of the completed scene. This is achieved by a combination of neural networks and a volumetric autoregressive network on different detail levels. Although scene completion is a problem most relevant to RGB-D data, and a high level of completeness of the input data is assumed for this work, the work of Dai et al. [34] demonstrates the potential of scene completion.

Although using synthetically generated multi-view images and point clouds from BIM objects, Koo et al. [35] propose an interesting approach to subtype classification of walls and doors, i.e. distinguishing doors into single, double, sliding or revolving doors and walls, respectively. The proposed method is a promising addition to existing semantic segmentation methods that are limited to the building components' super-types. Generally, the proposed approach paves the way towards a modular approach to semantic segmentation, stacking a set of classification models for different levels of detail.

Yin et al. [40] enhance the popular PointNet++ architecture by adding a local aggregation operation layer, stride residual bottleneck blocks in the encoder, and shared multi-layer perceptrons to the decoder. The proposed ResPointNet++ architecture thus outperforms the PointNet++ architecture by 42% mIoU in predicting the classes beam, pipe, pump and tank in an industrial building dataset. It must be noted that the size of the dataset seems to limit the general capabilities of the proposed architecture, although a high accuracy was achieved. On the other hand, as Yin et al. [40] explain, other classes such as floor and ceiling were excluded from the model training and evaluation, although these seem to be relevant in scan-to-BIM of industrial buildings.

Until now, papers related to semantic segmentation from the literature review of Schönfelder et al. [28] have been discussed. Deliberately or not, mainly papers using shape descriptions and a semantic label inference applying random forests or Markov random fields are covered in their review, except for ResPointNet++ [40]. Beyond that, the overview of Schönfelder et al. [28] seems limited in a two-fold way regarding semantic segmentation as an essential step in many scan-to-BIM pipelines. First, it ignores libraries for neural network processing of large-scale point clouds. Second, benchmark challenges such as ScanNet [31] were not researched to gain an overview on the advances in this field. This

may have been caused by the search method used by Schönfelder et al. [28], i.e. a database search on specific keywords such as scan-to-BIM followed by categorising and sorting the initially researched papers. Thus, the following sections on Libraries for large-scale point cloud processing will provide a more comprehensive overview of the field.

Libraries for large-scale point cloud processing

Libraries for large-scale point cloud processing will be researched and discussed based on a citation search, but mostly by identifying the key contributions from benchmark challenges. These approaches will be compared based on their results in the benchmark challenges ScanNet [31] and S3DIS [32], as both provide data most similar to the scope of this work. It must be noted that not all submissions to benchmark challenges can be discussed here. As many of the submissions rely on few extensive libraries while adopting or slightly improving the architectures initially proposed, only the main contributions and papers will be discussed here. This will enable the reader to understand the major advances in the field of neural network based semantic segmentation of large-scale point clouds. An overview on the papers discussed in the following is given in Table 2.2. Papers are sorted according to the mIoU accuracy on the ScanNet dataset in descending order (the best result on top). In some cases, results were only reported on one or neither of the benchmark datasets (marked with '-').

Author	Year	Title	Short Description	ScanNet	S3DIS
Wu et al. [42]	2023	Towards Large-scale 3D Representation Learning with Multi-dataset Point Prompt Training	Multi-dataset training with a prompt adapter and language-guided categorical alignment.	76.6%	72.7%
Wu et al. [43]	2022	Point Transformer V2: Grouped Vector Attention and Partition-based Pooling	Grouped vector attention for efficient point transformers.	75.2%	71.6%
Zhao et al. [44]	2021	Point Transformer	Self-attention point transformer network.	72.5%	73.5%
Graham et al. [45]	2018	3D Semantic Segmentation with Submanifold Sparse Convolutional Networks	Submanifold sparse convolutional networks for computationally efficient 3D semantic segmentation.	72.5%	-
Choy et al. [46]	2019	4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks	Sparse tensors implementation to use 2D convolutional layers for 3D and 4D data. Evaluation and ablation study on benchmarks.	72.1%	65.35%
Thomas et al. [47]	2019	KPConv: Flexible and Deformable Convolution for Point Clouds	KPConv represents the input points by Kernel points and associated filter values.	68.4%	67.1%
Hu et al. [48]	2020	RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds	Efficient random sampling and local feature aggregator for large-scale point cloud segmentation.	64.5%	70.0%

Table 2.2: Papers on data-driven approaches to semantic segmentation beyond the analysis of Schönfelder et al. [28] with results on ScanNet [31] and S3DIS dataset [32]. Results are calculated using the mIoU.

Author	Year	Title	Short Description	ScanNet	S3DIS
Rethage et al. [49]	2018	Fully-Convolutional Point Networks for Large-Scale Point Clouds	Uniform sampling strategy followed by PointNet layers as a low-level feature descriptor.	44.7%	-
Su et al. [50]	2018	SPLATNet: Sparse Lattice Networks for Point Cloud Processing	Lattice filters to extract spatially-aware features.	39.3%	-
Qi et al. [41]	2017	PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space	Partition sampling and recursive PointNet as local feature aggregator.	33.9%	-
Tchapmi et al. [51]	2017	SEGCloud: Semantic Segmentation of 3D Point Clouds	Combining a 3D Fully-connected neural network with trilinear Interpolation and a fully connected random field.	-	48.92%
Qi et al. [52]	2017	PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation	PointNet learns to extract critical points containing information w.r.t. classification or semantic segmentation.	-	47.71%
Zhong et al. [53]	2023	Understanding Imbalanced Semantic Segmentation Through Neural Collapse	Simplex equiangular tight frame (ETF) as a final classification layer for unbalanced class distributions.	-	-

Table 2.3: Papers on data-driven approaches to semantic segmentation beyond the analysis of Schönfelder et al. [28] with results on ScanNet [31] and S3DIS dataset [32]. Results are calculated using the mIoU [continued].

3D point clouds have a sparse nature, i.e. not every voxel in the 3D space is populated, resulting in 'irregular, non-uniform spatial distributions of points' [43]. When applying standard convolutional layers to 3D data, the sparse nature of the data will cause a computational overhead as practically every voxel will be processed regardless of whether it is populated or not. The papers in Table 2.2 will thus be analysed to reveal the recent advance in the state-of-the-art of point cloud semantic segmentation. The advance in the state-of-the-art has been threefold: (i) Multi-view CNNs have been proposed to translate point clouds into two-dimensional dense grids to apply network architectures for image processing. (ii) Network architectures consuming points directly have been developed to overcome the computationally expensive projection and re-projection in the multi-view CNN approaches. Under this branch, many methods are gathered such as PointNet [52] based approaches, sparse tensors [54], and others. (iii) Transformer architectures have been applied to the 3D domain pointing towards a new direction in further advances of semantic segmentation of point clouds. Note that the processing of volumetric point cloud representations has been widely researched by [55], [56] and others. Due to high computational expense such approaches have not gained much attention in the last years and will thus not be considered any further.

The case for multi-view CNNs is similar. Although Qi et al. have demonstrated that both volumetric and multi-view CNNs can have a similar accuracy, this early study only covers basic classification tasks on minor-scale point sets [55]. A major limitation of multi-view CNNs for large-scale scene understanding is that it is non-trivial to identify the camera locations to deliver highly descriptive rendered images of the entire scene while avoiding occlusions. Either the complexity of the scene needs to be limited or a

high number of images needs to be rendered and processed to deliver a prediction on all points. The inevitable overlap of the rendered images implies the problem that from differing label predictions one needs to be chosen for the final prediction. The probability of the inferred prediction could be considered yet adding more computational overhead to the method. Thus, the page has turned for Multi-view CNNs and to overcome their limitations, networks directly consuming points will be discussed in the following sections.

PointNet and related networks

PointNet was a milestone in the development of neural network architectures and one of the first capable of processing points directly. Qi et al. [52] implemented the simple yet effective idea to train the network to select critical points from the point cloud that are particularly informative for the specific task. However, the network is also capable of learning local and global features that are exploited for per-point classification problems such as part segmentation or semantic scene segmentation. PointNet is limited regarding its ability to learn and aggregate local features that are important for scene understanding. To address this issue, Qi et al. [41] proposed PointNet++ enhancing PointNet by adding a better partitioning of the point cloud and applying PointNet recursively as a local feature learner, thus abstracting large local regions. Compared to subsampling approaches, PointNet++ can handle point clouds with varying densities, the typical output of most 3D capturing devices. Rethage et al. [49] propose applying a uniform sampling strategy followed by PointNet layers as a low-level feature descriptor. The proposed fully-convolutional point network architecture seems to learn global features and relationships better than PointNet++ [41], as it reaches a 10% higher mIoU in the ScanNet benchmark [31].

From the same era, when computational efficiency of point processing CNNs was a real issue, hybrid approaches were developed to exploit the reasoning capabilities of Neural Networks on a coarse level while preserving enough detail for accurate per-point predictions. Such an approach is presented by Tchapmi et al. [51] combining a 3D fully-connected neural network with trilinear Interpolation and a fully connected random field. Although the proposed SEGCloud method processes a voxelized input point cloud, it is noticeable that such hybrid approaches have occasionally pushed the accuracy on point cloud classification and segmentation tasks, and might be interesting for future development.

Su et al. [50] propose the use of sparse lattice filters to extract spatially-aware features from the input points. This approach is interesting as it allows an easy mapping of images into 3D points and vice-versa, i.e. points and corresponding images can be both exploited if features of both modalities add up to a better classification.

Point convolution networks

Facing the limitations of PointNet style architectures regarding computational cost on higher resolutions, possible information loss by applying strict grid-style representations and others has led to a comeback of the end-to-end networks, with multiple new architectures for a more efficient, yet detail-preserving 3D point cloud feature learning. In this field, special attention was drawn to modify convolutional networks to facilitate efficient 3D data processing.

Sparse 3D convolutional networks have been introduced by Graham [54]. However, their first study was limited to 3D object recognition. Graham et al. [57] further develop their approach towards sub-manifold sparse convolutional networks that are tested on 3D object detection. The later contribution of Graham et al. [45] incorporates an approach to 3D semantic segmentation. A particular advantage is that the approach tries to preserve 'the same level of sparsity throughout the network', thus reducing the computational expense while preserving the details of the input data by implementing sub-manifold sparse convolutional networks. In the ScanNet benchmark challenge [31], a semantic segmentation accuracy of 72.5% mIoU could be achieved.

Introduced by Thomas et al. [47], KPConv represents the input by kernel points and associated filter values. This approach allows deformable convolutions to be implemented, proving to be more efficient on larger scenes. Choy et al. [46] propose the Minkowski U-Net architecture which uses sparse tensors, and generalized sparse convolutions that allow any 2D convolutional layer to be used in 3D and 4D data. The term *spatio-temporal* refers to 4D data, i.e. 3D data with additional time information such as RGB-D video streams or a temporal sequence of Light Detection and Ranging (LiDAR) scans.

Although RandLA-Net by Hu et al. [48] does not outperform the previously discussed approaches in mIoU accuracy on the discussed benchmark challenges [31, 32], there are advances in computational expense and memory consumption. This is achieved by using efficient random sampling and a local feature aggregator. RandLA-Net takes 185 seconds to infer on a scene of the SemanticKITTI [58] dataset, while being capable of inferring on 1.03 million points. This is considerably faster and more efficient than, e.g. KPConv [47], taking 717 seconds for the inference on the same scene while the maximum number of inference points is limited to 0.54 million [48]. The same computational setup was used for both experiments. Although there could be gains in computation time with more capable hardware, the network architecture is a limiting factor.

Point transformers

Another approach that allows for an efficient processing of large-scale point clouds in neural networks are Point Transformers. Zhao et al. [44] propose an end-to-end transformer-based architecture with an attention mechanism and position encoding. The position of points is encoded by farthest point sampling, where the distance of each point to the centroid is used. The XYZ coordinates are fed through the entire network to maintain full precision for semantic segmentation. Beyond the initial Point Transformer network, the method has been improved by adding grouped vector attention, an improved position encoding scheme and a partition-based pooling strategy [43]. Point transformers enable the network to shift the attention towards features containing information, thus allowing more flexibility to aggregate local and neighbourhood features compared to more rigid methods as discussed above. This makes Point Transformers the architecture of choice for the semantic segmentation network in this work. Details will be described in chapter 3.

Exploiting datasets and data characteristics to improve the network output

Although there seems to be great potential in developing new point convolution architectures, approaches involving multiple datasets in the training yield impressive results. One of the most recent

ones is Point Prompt Training introduced by Wu et al. [42]. The strategy to allow the features of two datasets being learned by the network is twofold. (i) A prompt adapter lets the model adapt to the specific dataset domain, (ii) the language-guided categorical alignment helps to align the categories specified differently in each dataset. Although Prompt Point Training uses a sparse U-Net [46], leading results on multiple benchmark challenges are reported [42] (see also Table 2.2).

Typically, the distribution of points among the classes is unequal. Quantitatively, in the ScanNet dataset, floor and wall comprise 35.7% and 38.8%, respectively, followed by chair comprising 3.8% of all points [31]. As discussed by Zhong et al. [53], this may affect the final classification layers in particular. To mitigate the effects of uneven class distribution, Zhong et al. propose a simplex equiangular tight frame (ETF) as a final classification layer, thus leveraging the accuracy in the ScanNet200 benchmark [31] by 6.8%. As proposed by [42], a standard MinkowskiNet [46] is used as a backbone [53].

2.2.2 Knowledge-based approaches

Recent years have witnessed significant progress in data driven point cloud segmentation, thanks to advancements in Machine Learning. This has led to improvements in result accuracy, computational efficiency, and the ability to handle large-scale datasets, as detailed in the previous section. In contrast, there have been fewer publications focusing on knowledge-driven approaches for semantic segmentation. A summary of works related to knowledge-driven strategies is presented in Table 2.4. These strategies employ logical frameworks or algorithms to accomplish semantic segmentation without the inevitable training associated with supervised Machine Learning techniques.

The approach proposed by Bassier et al. [59] comprises four steps: (i) A preprocessing is applied for noise removal using a size filter. (ii) After a normal segmentation, horizontal and vertical surfaces are classified for an initial labelling. (iii) The initial labels are optimized including grouping of horizontal and vertical surfaces, followed by (iv) window and door detection. The approach is evaluated in two experimental setups using manually annotated point clouds of one school and one residential building. Given that only two building point clouds with varying topology and features were used for testing, the performance in a more general context cannot be assessed.

Macher et al. [60] propose a three-step approach to semantic labelling of point clouds. (i) Floor and ceiling are segmented exploiting the distribution along the Z-axis. (ii) Per storey, rooms are segmented in a 2D approach, and (iii) planes composing the rooms forming wall candidates are fitted to the data. The distribution-based approach to floor and ceiling detection represents an interesting approach that can be applied to other objects, too. It is also computationally cheap to build a histogram and find peaks in the histogram programmatically, which is the general approach to distribution-based segmentation. Parts of this concept will be adopted in this work as well.

Beyond Macher et al. [60], Anagnostopoulos et al. [61] propose a segmentation algorithm for floors, ceiling and walls. Initially, a plane segmentation is performed using the Random Sample Consensus (RANSAC) algorithm [64]. Following the Manhattan world assumption [21], planar patches along x-z or y-z planes are classified as walls, those in the x-y plane as floor or ceiling. Based on the same principle, clutter is identified for planar patches of different orientation. Although the reported accuracy of 100% seems impressive, the generalization capabilities of this approach must be questioned as the RANSAC algorithm [64] is prone to segmentation errors when applied to unseen noisy data.

Author	Year	Title	Short Description	Experimental results
Bassier et al. [59]	2016	Automated Semantic Labelling of 3D Vector Models for Scan-to-BIM	Reasoning framework including initial surface classification, grouping and window/door detection	Evaluation on one house and one school point cloud, overall element precision 86.2%
Macher et al. [60]	2015	Point clouds segmentation as base for as-built BIM creation	Three steps approach for floor/ceiling, room and vertical plane segmentation.	Only a qualitative result of the method is provided for a house and academic building.
Anagnos-topoulos et al. [61]	2016	Detection of walls, floors, and ceilings in point cloud data	RANSAC plane segmentation and Manhattan world planar patch classification.	100% of floor, ceiling, exterior and interior walls. No further evaluation of the accuracy.
Michailidis et al. [62]	2017	Bayesian graph-cut optimization for wall surfaces reconstruction in indoor environments	Occlusion-aware 2D wall opening segmentation	Tested on several office point clouds, only one misclassification is reported. Results are described qualitatively, the quantitative accuracy is not studied any further.
Ponciano et al. [63]	2021	Object Semantic Segmentation in Point Clouds — Comparison of a Deep Learning and a Knowledge-Based Method	Multi-view Deep Learning and a knowledge-based approach for semantic segmentation.	Both approaches evaluated on urban scene point clouds. Scores (mIoU): knowledge-based 65%, Deep Learning 51%.

Table 2.4: Papers on knowledge-driven approaches to semantic segmentation. The listed papers were not evaluated on a common benchmark, hence experimental results are only outlined here.

Other than labelling building objects, Michailidis et al. [62] propose an occlusion-aware approach to segment wall and door openings in wall surfaces. After an initial wall patch segmentation, window and door boundaries are identified using α-shapes boundary estimation. The boundaries are then refined by RANSAC line fitting. To detect occlusion, the wall surface are transferred to a voxel representation labelling the voxels empty, occupied and occluded. These preliminary features are used to perform wall segmentation through graph-cut optimisation.

Comparing the results of data-driven and knowledge-based approaches remains a seldom opportunity, as only few studies compare such approaches. One of such studies was carried out by Ponciano et al. [63] comparing a multi-view deep Learning approach with a knowledge-based method. In the comparative study, the knowledge-based approach outperformed the Deep Learning method by 14% with a mIoU of 65%. The knowledge-based approach used semantic web technologies to model the relationships of data, object, scene, geometry and algorithm for the semantic segmentation framework. Unfortunately, both approaches were only evaluated on point clouds of street scenes. However, the study shows the potential of semantic web for advanced knowledge-based reasoning frameworks for semantic segmentation.

Although knowledge-based methods include significant user control, are computationally cheaper and better comprehensible, some limitations apply. These can be summarized as follows. The approaches

suffer from weaker generalization capabilities, struggle to manage complex features and require input parameters that cannot be inferred from the unseen data trivially. Although no data-driven approach with high generalization is witnessed, i.e. a segmentation model yielding good results on different types of buildings and scenes, it is still obvious that the pathway towards general data-driven semantic segmentation methods is already laid, which should culminate in a universally applicable point cloud segmentation model.

2.2.3 Technical aspects and conclusions

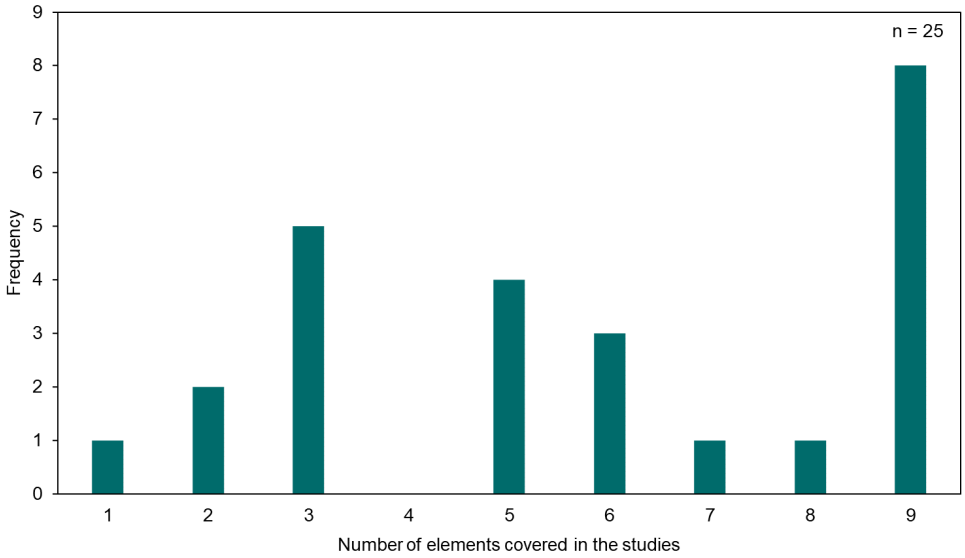


Figure 2.4: Number of elements covered in the studies.

Given the advances in semantic segmentation of point clouds described in the previous sections, it is vital to assess the usability of these methods for scan-to-BIM systems as proposed in this work. The first step is to assess the accuracy. All studies that report results on either the ScanNet [31] or S3DIS [65] datasets are compared. Out of 25 studies considered, on the ScanNet benchmark [31], a median mIoU of 72.1% (n = 11) is reported, and on the S3DIS dataset [65] 70.0%, respectively (n = 9). To assess the reported median mIoU of around 70% on both datasets, some more context is helpful. Figure 2.4 shows a plot of the number of elements covered in the studies and reveals that 8 out of 25 studies cover 9 elements. The mIoU is related to the number of elements in the studies. For 2 elements, a mIoU of 50% corresponds to a random prediction. Thus, an overall mIoU of around 70% can be interpreted as a decent accuracy which makes the proposed methods usable in scan-to-BIM pipelines. This is particularly important for this work, as semantic segmentation serves as the critical first step in the proposed framework. Poor segmentation accuracy can only be compensated to a limited extent.

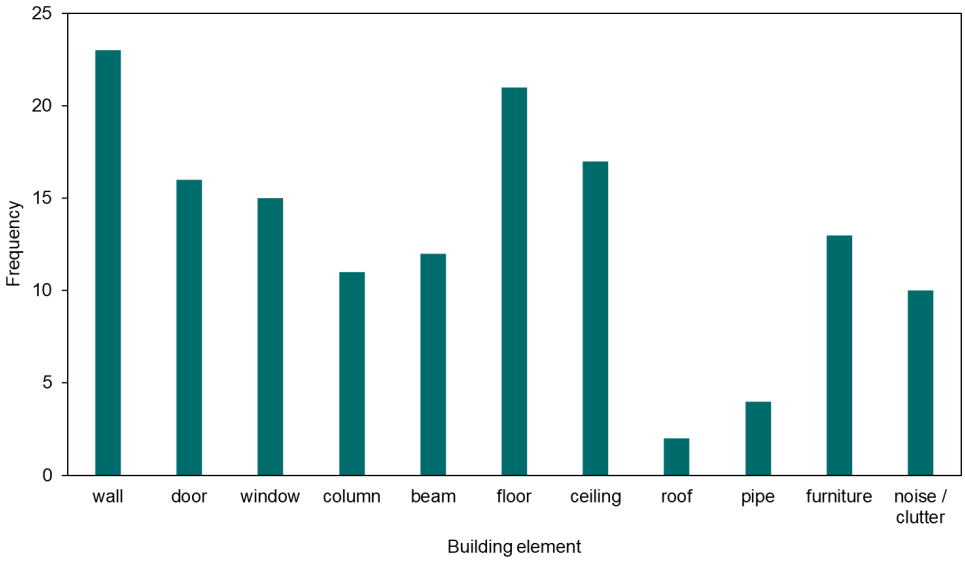


Figure 2.5: Components examined in the reviewed semantic segmentation research.

In this work, Point Transformer V2 [43], the S3DIS [65] dataset and the CV4AEC scan-to-BIM benchmark dataset [3] will be used to train a model for semantic segmentation. As of the end of 2023, Point Transformer V2 [43] achieved the highest mIoU on the S3DIS [65] dataset of all studies reviewed here, and the second highest on the ScanNet benchmark dataset [31]. Note that the S3DIS [65] dataset contains a wider variety of structural elements than the ScanNet benchmark dataset [31].

Regarding the components examined, there is a good portion of structural ones such as wall, column, beam, floor, ceiling, etc (see Figure 2.5). Opening elements such as door and window are covered by the studies as well. However, only 2 studies segment roofs, which is related to the fact that the most common benchmark datasets [65, 31] provide mostly indoor scenes naturally excluding outside surfaces and components. Besides structural components, furniture and its sub-classes are part of 13 studies. This reveals two major research gaps: (i) the segmentation methods and datasets need to be expanded towards the outside components of buildings, and (ii) the methods and datasets include interior objects but avoid segmenting point clouds of empty, and even partially demolished buildings revealing structural elements and joints.

Overall, the research gaps can be summarized as follows. There is a lack of datasets with semantic annotations of structural building component classes that are critical for training and generalization. Such classes are essential to scan-to-BIM pipelines yet are underrepresented in the available segmentation benchmarks [65, 31]. Computational efficiency remains another issue, despite considerable advances. Overall, a large foundation model for point cloud segmentation is not available, marking a major research gap. However, there have been significant advances in other data modalities, such as

[3] https://cv4aec.github.io/

the *Segment Anything* model [66], which could be used for segmentation in a multi-view scenario.

2.3 Geometry processing algorithms for scan-to-BIM

To bridge the gap between a segmented point cloud and a complete BIM reconstruction, the geometric parameters of the objects observed need to be determined. This involves both clustering of object instances and primitive fitting to determine the geometric shape and the translation, rotation and transformation of the object. There will be a strong emphasis on the state-of-the-art of the overall scan-to-BIM process in section 2.4, covering both the application of semantic segmentation and geometry extraction algorithms in an integrated manner. Hence, there is no need to repeat this analysis solely on geometry extraction methods. However, there are key methods for both clustering and primitive fitting that this work relies on. These will be introduced in the following, presenting the rationale for the choice and providing the foundations for the application and adaptation of these methods.

2.3.1 Clustering

Clustering comprises the process of grouping together subsets of data points that share similar features. Applications can be rather generic, e.g. sorting all points of walls that span along a certain axis in the Euclidean space, or more complicated approaches exploiting features such as the point density. The goal is to separate instances of specific objects within the overall set of points assigned to a specific class, e.g. wall instances within all wall points. The most common methods are k-means clustering and the Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise (DBSCAN).

The k-means algorithm was initially proposed by MacQueen et al. [67]. Starting with k initial random points, each new point is added to the group with the nearest mean. If a point is added, the mean is adjusted accordingly. Hence, at each iteration, the *k-means* represent the mean of each cluster. Beyond introducing the methodology, MacQueen et al. explore various fields of applications and tested the algorithm with the computational means available in 1967, the date of publication. Since then, computers have gained significantly more computational power, which facilitates the application of *k-means* on large datasets in the euclidean space such as point clouds. In the context of scan-to-BIM, the major limitation of *k-means* is that the desired number of clusters needs to be known before the algorithm is applied. In scan-to-BIM pipelines, the number of clusters often corresponds to the number of object instances (doors, windows, columns, ...) but cannot be inferred trivially.

Unlike *k-means*, the DBSCAN algorithm (refer to Algorithm 2), developed by Ester et al. [68], does not require the number of clusters to be specified in advance. In DBSCAN, cluster formation is based on incorporating all points that are 'density-reachable,' meaning all points within a neighbourhood of comparable density. Regions with lower point density, or fewer points within the local radius ε[4], below a certain threshold, are not included in the clusters. This approach offers several advantages. The only necessary input parameters are ε, which sets the local radius for density calculation, and *MinPts*, indicating the minimum number of points within the radius ε. There is no need to input an expected

[4] Here, the Greek letter is used, whereas the term 'Eps' was used in the original research paper

number of clusters, which is often hard to infer in real-world scenarios. Additionally, the algorithm naturally excludes noise and clutter by not assigning lower density areas to any cluster. In terms of computational cost, DBSCAN is notably efficient, with its runtime scaling linearly with the number of sample points [68]. Therefore, DBSCAN is chosen as the most suitable clustering method in this work.

Petschnigg et al.'s [69] findings underpin this choice, as they argue against using methods that require pre-specification of the number of clusters in reconstruction processes. They advocate for DBSCAN [68] as a preferred option. Petschnigg et al. [69] slightly favour Ordering Points To Identify the Clustering Structure (OPTICS) [70], a variant of DBSCAN [68] in their research on reconstructing car factory simulation models from point clouds. However, OPTICS [70] is an algorithm to compute an implicit cluster structure of the dataset. Although this might be useful for many applications, in this work the explicit clusters are needed. Given that OPTICS [70] applies the same principle of 'density-reachability' as DBSCAN [68] with comparable computational efficiency [70], no significant performance improvement is anticipated, leading to the adoption of DBSCAN in this study. Furthermore, it will be proved that a global parameter setting for DBSCAN can be determined, especially when the point density is equalized by an initial voxel downsampling, and that some additional filtering is achieved as a positive side effect.

2.3.2 Primitive fitting

Even in a clustered point cloud, the geometric parameters including dimension, translation, rotation, and transformation of the clusters observed are still unknown. One common approach to retrieve the geometric parameters is referred to as primitive fitting. Theoretically, parameterized primitives are fitted to the data, and the retrieved parameters describe the primitive distinctly in the 3D space. In a practical example, a cylinder is fitted to a pipe segment in the point cloud and the parameters retrieved (axis, centre point, radius, length) can be subsequently used for automated BIM authoring.

The purpose of primitive fitting is twofold. On the one hand, the geometric parameters can be obtained. On the other hand, primitive fitting inherently helps to avoid considering outliers and noise in the reconstruction step. During plane fitting, remainders of objects attached to the wall are inherently eliminated, with the plane reconstructed providing exact parameters, e.g. of a wall surface, along with the inlier points.

One could argue that methods to estimate the bounding geometry need to be considered in this literature review as well. However, the fundamental difference is that the bounding geometry of a set of points provides the geometric parameters of the boundaries considering all points as inliers. This might contain noise, artefacts and outlier points that need to be distinguished from the inlier points for an accurate reconstruction, which can only be achieved by applying primitive fitting methods, not by estimating the bounding geometry solely.

A comprehensive overview on primitive fitting methods is given by Kaiser et al. [71] who identify stochastic, parameter spaces and other clustering techniques as theoretical concepts to the problem of primitive fitting. The main algorithms used in the concepts are RANSAC as a stochastic approach, Hough transformation exploiting parameter spaces and primitive-driven region growing. The following list briefly describes the main algorithms, along with potentials and limitations. More details and references can be found in [71].

```
Input  : wall points                          // sorted by direction, either X or Y
Input  : ε                                                    // the search radius
Input  : min points                        // minimum number of points in a cluster
Output: cluster labels                  // ID of the cluster the point is assigned to
// random seed point initially, then from points without cluster ID
Function expand cluster(wall points, seed point, cluster ID, ε, min points):
    // find the subset of points within a region with radius ε
    candidates = region query(Axis sorted wall points, seed point, ε)
    if number of(candidates) ≤ min points then
        label(candidates) = Noise
    end
    else all candidates are density reachable from point
        label(candidates) = Cluster ID
        while candidates do Do while loop as long there are candidates
            for point ∈ candidates do
                local neighbours = region query(wall points, point, ε)
                if number of(local neighbours) ≥ min points then
                    for point ∈ local neighbours do
                        if point unclassified then
                            label(point) = cluster ID
                        end
                    end
                end
            end
        end
    end
    return Boolean
// apply on all wall points
Function dbscan(wall points, ε, min points):
    for point ∈ wall points do
        if point is unclassified then
            if expand cluster(wall points, point, cluster ID, ε, min points) then
                cluster ID = cluster ID + 1
            end
        end
    end
    return Label array
```

Algorithm 2: DBSCAN: density based clustering according to [68].

- **RANSAC** The most popular method for primitive fitting including plane fitting is RANSAC shape detection. The basic principle is to randomly sample a number of points n that define the respective geometric primitive. For planes a minimum number of $n = 3$, for other primitives more seed points are needed. With the randomly sampled points, a primitive is defined and the deviations from the point set are calculated. Previously defined limits help to determine if the fitted primitive can be accepted or must be declined. By iteratively applying this principle, the best fitting primitive can be approximated. According to Kaiser et al. [71], variants mainly propose more elaborate methods to identify the best fitting primitive. RANSAC has a strict filtering side effect, which is beneficial to the reconstruction accuracy. Beyond that, Kaiser et al. [71] argue that RANSAC based methods are robust to outliers.

- According to Kaiser et al. [71], the **Hough transformation** creates a collection area based on a set of parameters where similar geometric shapes meet. This area is divided into uniform sections, and every piece of data contributes to the voting for each geometric shape it aligns with. The set of parameters that receives the most votes indicates the object that most accurately represents the given data. The main limitations are that the Hough transformation can be memory consuming and computationally expensive in the three-dimensional space, as the parameter space needs to be discretized.

- According to Kaiser et al. [71], in **Primitive-driven region growing** approaches a random seed point is initially selected. Adjacent points' features are then analysed to determine if the points observed can be assigned to a primitive. The features considered comprise colour, depth or normal orientation. A common application is plane fitting, where either the normal or the surface curvature can be used as the planar heuristics. Naively, each adjacent points' normal or surface curvature is computed, and points are assigned to the region when the normal or curvature deviation is below a previously set limit. Compared to RANSAC, the filtering effect is less strict as there is no distance-to-primitive limit in region growing approaches. Even slight curves in a plane primitive could be preserved in the segmented points. Kaiser et al. [71] argue that region growing based methods are not robust when applied to incomplete data. This is intuitively evident, as the expansion of a region will terminate when no adjacent points are found, thus failing to retrieve the complete primitive and correct parameters.

In this literature review, RANSAC will be further examined as it is one of the most robust yet efficient algorithms for primitive fitting. Beyond that, recent Machine Learning approaches will be discussed carving out potentials and limitations. A rationale will be given regarding which method will be adopted in this work.

In this study, the RANSAC primitive fitting approach (refer to Algorithm 3) introduced by Schnabel et al. [64] is used, which is the most efficient and robust to the knowledge of the author. Initially, a set of random seed points is sampled from the input points. A local neighbourhood constraint is applied, as adjacent points are more likely to deliver a well-fitting primitive. The seed points are used to derive a plane equation and the plane inliers are retrieved in parallel. If plane inliers exist, and the number is above the minimum points per plane parameter and the points' distances are lower than the distance threshold ϵ, the plane is accepted, and the points are removed from the remaining points. The algorithm is repeatedly applied until no further planes can be segmented.

Beyond implementing the RANSAC principle for planes, spheres, cylinders, cones and tore, an efficient sampling strategy is introduced to reduce the computational cost, as this is mostly dictated by the num-

ber of iterations [64]. To the experience of the authors, the efficient RANSAC is capable of segmenting dense point clouds of large scenes and is even robust to noise and outlier. It can handle up to 20% noise and up to 95% outlier ratio as Schnabel et al. report [64]. Nevertheless, there is still potential to make the algorithm more efficient. Generally speaking, the better the initial sample, the fewer iterations are required to fit a shape exactly to the data. Hence, we seek to further improve the RANSAC principle with even more advanced seed sampling strategies. Our approach will be introduced in subsection 4.6.3.

Input : input points
Input : number of random seed points `// used to find the plane equation,` ≥ 3
Input : min points per plane `// discard planes with few points`
Input : ε `// maximum distance of point to plane`
Output: plane equation
Output: plane inliers

remaining points = input points `// remaining points not yet assigned to plane`
while *remaining points* $\geq$ *min points per plane* **do** Loop as long as points remain
  `// find random seeds with local neighbourhood constraint, as points in local neighbourhood are more likely to result in a good fitting plane`
  seeds = `random seeds(`*remaining points, number of random seed points*`)`
  plane equation, plane inliers = `fit plane(`*seeds, remaining points*`)`
  if `any(`*plane inliers*`)` **then**
    if `number of(`*plane inliers*`)` $\geq$ *Min Points per plane* **then**
      if *distance* $\leq \epsilon$ **then** check distance of all points to plane
        `remove(`*remaining points, plane inliers*`)`
      end
    end
  end
end
return *plane equation, plane inliers*

Algorithm 3: RANSAC plane fitting according to [64].

Li et al. [72] claim that the key challenge of applying RANSAC to real-world data is to select the correct input parameters for the algorithm. Indeed, this is challenging in multi-primitive segmentation scenarios in large scenes, as the parameters once chosen might not give good results on all primitives and features of the data, e.g. partially occluded or partially scanned objects. However, our use of RANSAC is to apply it to previously clustered sets of points, solving the problem of parameter estimation by applying heuristic approaches.

While the efficient RANSAC method [64] has long been the benchmark for fitting primitives, it is important to acknowledge the emerging role of Machine Learning techniques in this domain. These ML-based approaches fall into two categories: supervised methods, which utilize pre-classified data for training, and unsupervised methods, which operate on unlabelled data. To provide insight into the strengths and weaknesses of each, one notable example from both supervised and unsupervised methods will be examined.

Li et al. [72] propose SPFN, a supervised primitive fitting approach tailored to point clouds of mechanical objects rather than entire scenes. The approach leverages PointNet++ [41] as a feature extractor. The

extracted features' primitive membership, normals and type are then used to infer the parameters of the primitive observed. The method is trained on a synthetically generated dataset of Computer Aided Design (CAD) objects and compared to efficient RANSAC [64], outperforming the latter in several tests. Despite the outperforming nature of SPFN it is not preferable in this work, as it is primarily developed and trained to fit primitives to CAD objects, not to entire scenes with noise and clutter.

Saporta and Sharf [73], however, propose an unsupervised deep learning approach to fitting 3D primitives to points. Similar to [72], PointNet++ [41] is leveraged for point segmentation without pretraining, i.e. in an unsupervised manner. By applying a Matrix for point-to-primitive relations recursively, primitives are fitted to the data. After each iteration, points assigned to a primitive are sliced, facilitating a coarse-to-fine fitting. The approach was tested on the same dataset as in [72], revealing a lower noise range and smaller point-to-primitive distance.

2.3.3 Technical aspects and conclusions

The methods further investigated and applied in this work are chosen based on the literature survey above and a qualitative evaluation of specific potentials and limitations. The rationale is described here.

For clustering, i.e. grouping of points with similar features and especially in a local neighbourhood, DBSCAN as introduced by Ester et al. [68] could be identified as an efficient yet robust solution to the problem and will thus be used in this study. Likewise, efficient RANSAC [64] will be used and extended with a more advanced sampling strategy.

Beyond clustering and primitive fitting, other methods such as convex hull estimation, bounding box fitting, voxel-based filtering and region growing are frequently used in scan-to-BIM pipelines. However, these are not subject to dedicated research. Thus, relevant approaches will be discussed in the following chapters as part of automated scan-to-BIM pipeline, and the methods used in this work will be introduced in chapter 4.

2.4 Automated scan-to-BIM pipelines

To align with the scope and objective of this work, literature needs to be studied that integrates the previously discussed approaches to semantic segmentation and geometry processing into pipelines. To this end, we will explore research that focuses on fully (end-to-end) or partially automating the conversion of point clouds into BIM models. Within the partial automation category, studies that offer automated solutions for specific steps of the overall process, like geometric reconstruction, will be reviewed. In the end-to-end category, studies covering all steps from the input point cloud to the retrieved BIM model are discussed.

As Patraucean et al. [74] argue, scan-to-BIM frameworks can be distinguished into those updating or enhancing a prior available BIM model, and those that do not. The latter is the only category studied here as it aligns with the scope and objective of this work. In the field of scan-to-BIM, to the knowledge of

the author, only the CV4AEC challenge https://cv4aec.github.io/ offers a benchmark for scan-to-BIM pipelines. As this benchmark is not widely adopted, the results' evaluation will mainly be described qualitatively.

Beyond the overview on partial and end-to-end scan-to-BIM pipelines given here, [74] and [75] review the state-of-the-art in scan-to-BIM creation with a wider perspective including point cloud preprocessing, representations, Mechanical, Electrical and Plumbing (MEP) component modelling, and geometry and information representations. For functional, informational, technical, organizational and legal aspects as well as research gaps for BIM application in existing buildings, [76] can be consulted. Beyond the processual aspects, [77] review the state-of-the art in data acquisition, data structures, processing and 3D reconstruction. Although being far from conclusive, the review of Gourguechon et al. [78] provides a nicely structured overview on the overall scan-to-BIM process including sound distinctions of various approaches. However, only a few deep learning approaches including semantic segmentation of point clouds are discussed.

2.4.1 End-to-end pipelines

Regarding automated as-built modelling, Gourguechon et al. [78] distinguish free space based and walls based approaches. The first denotes room identification before the reconstruction of the surrounding objects, the latter directly reconstructs walls. Likewise, Bassier et al. [79] propose distinguishing between wall-based and room-based reconstruction. Following the distinction proposed, this section will be separated into similar categories. Under the first category, studies on room-based BIM reconstruction from point clouds are discussed. Reconstructing free-space elements can leverage reconstructing the structural components encapsulating this space, thus reducing complexity and computationally expensive tasks such as semantic or geometric segmentation. The second category encompasses all studies focusing on the semantic and geometric reconstruction of structural components. In contrast to [78, 79], this does not only involve walls but also structural component such as columns, slabs and opening elements such as windows and doors.

Room-based reconstruction

An overview on room-based reconstruction approaches is given in Table 2.5. The studies all follow the principle of building component reconstruction based on an initial room reconstruction. The initial room reconstruction leverages the application of topology-based reconstruction rules, e.g. all rooms need to be encapsulated by walls, floor and ceiling and accessible by doors or other openings. As the studies prove, the room-based reconstruction can be a straightforward and efficient process, given a robust algorithm to reconstruct the initial spaces. This, however, can be challenging, as rooms are often cluttered with various objects, including furniture and other interior items.

Ochmann et al. [80] claim that their approach is one of the first to study the reconstruction of fully parametric and editable BIMs, instead of using meshes as a BIM geometry representation. First, all points are assigned a room label, which is inferred by exploiting the scanner positions assuming that every room was scanned by one or more scanner positions. Planes are segmented to identify wall candidates, which are projected onto the horizontal XY-plane, where wall centre line candidates are

Author	Year	Title	Short Description	Experimental results
Ochmann et al. [80]	2016	Automatic reconstruction of parametric building models from indoor point clouds	Room labelling, plane fitting, 2D wall candidate line identification, room-based confirmation of wall centre lines, ray-casting based door and window detection.	Promising results based on a qualitative visual comparison of manually and automatically generated BIM.
Macher et al. [81]	2017	From Point Clouds to Building Information Models: 3D Semi-Automatic Reconstruction of Indoors of Existing Buildings	Sub-space segmentation for slab and wall reconstruction robust to clutter and occlusions.	Precision and recall of floor, ceiling and wall segmentation and classification above 80%. Deviation of manually extracted and automatically generated floor plane points below 2 cm.
Xiong et al. [82]	2023	Knowledge-driven inference for automatic reconstruction of indoor detailed as-built BIMs from laser scanning data	Walls, windows, doors, floor, ceiling and electrical fixtures reconstruction based on a room segmentation, topology reconstruction, geometry regularization for BIM reconstruction.	Experiments on 7 scenes of academic buildings. Quantitative evaluation: Room segmentation mIoU and other metrics, reconstruction on point-to-BIM distance, correctness and completeness.
Bassier et al. [83]	2018	IFC wall reconstruction from unstructured point clouds	IFC wall reconstruction including segmentation, semantic labelling, wall reconstruction with parallel room reconstruction and topology optimization	Over 90% recall and precision for a multi-storey office building point cloud.
Ochmann et al. [84]	2019	Automatic reconstruction of fully volumetric 3D building models from oriented point clouds	Initial RANSAC plane detection, noise filtering, cell labelling as room or outside area and optimized geometric reconstruction with topologically correct wall connections.	Qualitative evaluation on several datasets.
Tang et al. [85]	2022	BIM generation from 3D point clouds by combining 3D deep learning and improved morphological approach	Volumetric BIM reconstruction of floor, ceiling, wall, window, door by semantic segmentation, surface extraction, space division and regularization.	Evaluation on segmentation precision and BIM reconstruction by calculating the cloud-to-BIM distance. 2 publicly available datasets, one self-captured dataset with non-Manhattan-world scenes.

Table 2.5: Room based BIM reconstruction pipelines.

fitted to the data. The wall centre lines are then intersected, and only those are accepted that separate different areas segmented before. Finally, doors and windows are detected. As the approach exploits the scanner location for room segmentation, mobile scanner data cannot be processed with such an approach. However, such scanning systems including mobile mapping and handheld solutions are becoming more and more common for construction data acquisition.

In a later study, Ochmann et al. [84] overcome some limitations of their previous work, including the necessity of using per-room scan positions for room segmentation. Beyond that, the method is extended

towards processing multi-storey buildings. The method comprises an initial RANSAC plane detection, noise filtering, cell labelling as room or outside area and a geometric reconstruction by plane intersection, and solving an integer linear program. The proposed method delivers a topologically correct set of walls and wall connections.

In the exhaustive study by Macher et al. [81], the data acquisition process as well as the semantic reconstruction of BIMs from interior point clouds are considered. For semantic reconstruction, a segmentation into sub-spaces, i.e. rooms is followed by a plane segmentation, classification and 3D reconstruction of walls and slabs. To segment the point cloud horizontally, the histogram along the Z-axis is computed, with peaks indicating floor or ceiling. The subspace, i.e. room segmentation, is performed in 2D, with a 30 cm horizontal slice just below the ceiling being projected onto the XY-plane to avoid the lower more cluttered areas. On this slice, a region growing algorithm with specific constraints is applied to segment the sub-spaces. Based on the subspace segmentation, floor and ceiling points are isolated using a plane segmentation algorithm. Based on the subspace boundaries, wall points are isolated, and a plane segmentation is performed. Maximum Likelihood Estimation Sample Consensus (MLESAC) is used for floor, ceiling and wall plane segmentation [86]. The planes are grouped together and converted to a mesh. FreeCAD's Architecture workbench allows the mesh to be converted to parametric BIM geometry objects. These are finally exported to the Industry Foundation Classes (IFC) format. The approach is evaluated on point clouds of one detached house and one academic building.

Beyond the approach of Macher et al. [81], Xiong et al. [82] propose an initial room segmentation and topology reconstruction to automatically provide BIMs, but include windows, doors, floor, ceiling and electrical fixtures into the reconstruction pipeline. An initial semantic segmentation based on a region growing algorithm and patch classification provides wall points, the centre lines of the wall entities are then used to perform the room segmentation. For topology reconstruction, the geometry is first regularized using constraints such as collinearity, orthogonality or parallelism. The topology is then modelled using nodes representing the elements and the respective connections. Among other elements, electrical sockets are detected in panoramic images and then transferred into parametric BIM objects.

Bassier et al. [83] propose a hybrid approach for IFC wall reconstruction comprising segmentation and semantic labelling, wall surface fitting and grouping. In parallel, partial rooms are fitted to the data. These are used to correct the topology of walls, floor and ceiling. In the topology optimization, a neighbourhood search is performed with the walls centre planes, with orthogonal or parallel planes below predefined distance thresholds being extended or clipped to provide correct topology. The proposed method was tested on a multi-storey building. Although the approach can handle multi-storey buildings and provides topologically correct reconstructions in the experiments, it must be noted that typically walls in BIM models are assigned to levels and are thus split per storey. Although a non-practical solution is provided, the approach for topology correction is a valuable contribution.

Beyond the studies of Macher et al. [81] and Bassier et al. [83], Tang et al. [85] introduce an initial semantic segmentation, before surface extraction, space decomposition, regularization and BIM modelling. In their study, Tang et al. use RandLA-Net [48] to perform a semantic labelling of the input point cloud. The classes include wall, beam, ceiling, door, floor and window. All points labelled as wall, beam, ceiling and floor are used for further plane fitting and space decomposition, which is performed by generating a 2D binary map and an improved morphological algorithm. Finally, a volumetric BIM model is generated which preserves the topological relationships of the objects [85].

Bassier et al. [79] argue that room-based reconstruction approaches are limited to indoor scenes, and

that these cannot be adapted to other structural components such as slabs. Although room-based reconstruction methods typically use advanced strategies to cope with clutter, this still seems to be the major limitation, as clutter is mostly present inside the rooms. With advances in semantic segmentation, reconstruction methods focusing directly on the structural components have become increasingly popular and the major focus of scan-to-BIM research. Such methods will be discussed in the next section.

Structural component reconstruction

Compared to the room-based reconstruction approaches, it seems more natural to directly convert structural components into BIM objects. This process involves the instance segmentation of components, assigning a semantic label to either the input data or geometrically segmented instances, and the generation of BIM objects. The order of the steps mentioned varies among the studies. An overview on the studies discussed is presented in Table 2.6 and Table 2.7.

The approach to BIM wall object reconstruction presented by Bassier et al. [79] focuses on the extraction of IfcWallStandardCase objects of multi-storey building point clouds, including representations for curved and polyline walls, as well as a topology reconstruction framework to intersect or merge neighbouring wall objects. The study introduces an approach to define wall types, i.e. widths of walls and to assign each wall of an arbitrary thickness to a wall type. The proposed method to topology reconstruction comprises the identification of candidate connections based on the k nearest neighbour search of the shortest Euclidean distance, some restrictions including the avoidance of IfcSpace geometries, and distance thresholds retrieving a set of valid connections and propose to repeat the topology reconstruction procedure multiple times. Noticeably, Bassier et al. [79] report that the accuracy of the automatically created walls even outperforms the manual modelling. Hereby, the accuracy is evaluated by the minimum Euclidean distance of reconstructed IfcWallStandardCase and ground truth mesh.

In the field of wall reconstruction, 2D and 3D approaches are competing. Based on their 3D wall reconstruction approach [79] already discussed, Bassier et al. [87] elaborate a comparative study on 2D and 3D wall reconstruction approaches. In this study the 3D reconstruction approach comprises planar segmentation, a SVM classifier, a per-object and per-storey clustering and reconstruction of the orientation, wall thickness, positioning and wall boundary. In the 2D approach the authors perform a histogram-based storey segmentation, then convert the points to a 2D point density raster image by projecting the points along the Z-axis onto the XY-plane. Followed by a line segmentation and clustering, wall points are extracted and reconstructed. Both approaches were compared regarding their reconstruction accuracy. In the 3D method the 95% confidence intervals is below +/- 5 cm for 88% of the reconstructed walls, in the 2D method only 55% are within the same error margin. Comparing both approaches is sensible, however regarding the variety of methodological options in 2D and 3D scan-to-BIM, 3D methods should not be favoured generally.

Beyond competing 2D and 3D methods, both 2D and 3D approaches can be combined as Gankhuyag et al. [88] propose. In the study, a two-step preprocessing procedure is used. First, the input point cloud is downsampled, reducing the number of input points while preserving the geometric integrity. Second, the formerly downsampled point cloud is now upsampled to add missing points in the wall planes for a better reconstruction accuracy. After this preprocessing step, a 30 cm horizontal slice from below the ceiling is projected onto the XY-plane for an image-based wall line and junction detection. Subsequently,

Author	Year	Title	Short Description	Experimental results
Bassier et al. [79]	2020	Unsupervised reconstruction of Building Information Modelling wall objects from point cloud data	IfcWall object reconstruction: mesh reconstruction and classification, IfcWallStandardCase creation and topology reconstruction.	Tests were performed using the S3DIS dataset [65]. 78 to 83% of points' distances $\leq 5cm$ to the automatically created model.
Bassier et al. [87]	2020	Comparison of 2D and 3D wall reconstruction algorithms from point cloud data for as-built BIM	Comparative study of one 2D method comprising histogram-based storey search, 2D projection of points, line fitting and reconstruction, as well as one 3D method using plane segmentation, planar patch classification, clustering and reconstruction.	The 3D method proves to be more precise, while the 2D approach's recall outperforms the 3D method.
Gankhuyag et al. [88]	2021	Automatic BIM Indoor modelling from Unstructured Point Clouds using a Convolutional Neural Network	Downsampling, Upsampling to add missing points, XY-projection of a horizontal slice for 2D wall and wall junction detection, IFC modelling of walls and doors.	96% F1 score on 2 areas of residential buildings.
Anagnos-topoulos et al. [89]	2016	Object Boundaries and Room Detection in As-Is BIM Models from Point Cloud Data	Wall boundary reconstruction and correction as well as space detection based on a previously proposed semantic segmentation method [61].	83% accuracy reported for space detection.
Mirzaei et al. [90]	2022	Automatic generation of structural geometric digital twins from point clouds	Slicing method for beam and column segmentation, PointNet [52] based profile type inference, structural member's size and function detection. BIMs generation through Autodesk Revit API.	Section label (I-, U-, Box-, round hollow section) inference accuracy above 81%, length accuracy 99%. The method was tested on one car park and one academic / office building.
Wang et al. [91]	2015	Automatic BIM component extraction from point clouds of existing buildings for sustainability applications	Building envelope as-is BIM by region growing plane segmentation, boundary detection and component classification.	Comparison of manually measured and recognized area of components on one case study.
Kim et al. [92]	2021	3D as-built modelling from incomplete point clouds using connectivity relations	Semantic and instance segmentation, geometric reconstruction, network update and BIM object generation.	Semantic segmentation IoU 87.05% for column and 99.76% for floor points. Semantic segmentation and modelling performance tested on two real-world point clouds.
Kim et al. [93]	2020	PinSout	Semantic segmentation using PointNet with synthetic training data, plane fitting for polygonal 3D reconstruction.	No quantitative evaluation.

Table 2.6: Structural component reconstruction pipelines.

Author	Year	Title	Short Description	Experimental results
Son et al. [94]	2017	Semantic as-built 3D modelling of structural elements of buildings based on local concavity and convexity	As-built BIM of concrete structural works with initial material segmentation, concavity/convexity-based instance segmentation, shape retrieval and volumetric IFC-compliant modelling.	99% of the elements recognized on two test scenes. No other accuracy metrics evaluated.
Thomson et al. [95]	2015	Automatic Geometry Generation from Point Clouds for BIM	IfcWallStandardCase and IfcSlab reconstruction by plane segmentation, euclidean clustering, boundary estimation and optional spatial reasoning for refinement.	Weighted sum of centroid deviation, area error and sine of angular deviation, evaluated on one corridor and one office room of an academic building. Per-instance accuracy and common accuracy metrics reported.

Table 2.7: Structural component reconstruction pipelines [continued].

doors are detected by applying an iterative volumetric 3D search on the wall points to detect openings. For all detected objects, IFC objects are created. The proposed method's accuracy is beyond 90%. However, doors can only be detected when open [88].

Thomson et al. [95] propose a three-step approach to deliver IFC objects of walls and slabs. First, RANSAC plane segmentation is applied with horizontal or vertical constraints to segment floor / ceiling and walls, respectively. Second, a Euclidean clustering with a planarity constraint is used to group planar patches of the same object. Finally, a convex hull algorithm is used to retrieve the object's boundaries. Optionally, spatial reasoning is applied, e.g. to reject small walls, extend walls or merge planes. As one of few studies, Thomson et al. propose novel accuracy metrics, which are defined as the weighted sum of centroid deviation, normalized area error, and sine of angular deviation.

The studies previously discussed focussed on walls as one of the core structural members. Beyond that, skeleton structures composed of vertical columns and horizontal beams are common for industrial buildings, car parks, etc. Hence, Mirzaei et al. [90] proposed a framework to automatically reconstruct BIMs of beams and columns. The proposed framework comprises a slicing method to identify beam and column points. Subsequently, PointNet [52] is trained on a synthetic dataset to infer the type of structural member including box beam, I-Beam, U-Beam and round hollow sections. The size and function of the structural member is inferred before a BIM model is created using the Autodesk Revit.

Beyond building components such as walls, space objects are typically part of BIM models to represent rooms. To this end, Anagnostopoulos et al. [89] propose a framework for wall reconstruction, object boundary extraction and correction as well as space detection. The method relies on the object detection algorithm previously proposed in [61] and reconstructs IfcWallStandardCase and IfcSpace objects. As reported, the space detection is limited to such fully encapsulated by walls. The proposed method is tested on one point cloud of an office building and evaluated against a manually created ground truth BIM using CloudCompare's cloud-to-cloud comparison [4].

Most of the studies discussed propose to use 3D geometric or semantic segmentation as one of the first steps towards an as-built model. As their study focuses on as-built modelling and update of ongoing structural concrete works, Son et al. [94] apply a material segmentation algorithm as the first step

to identify all points related to concrete structural components. The scene is then further segmented, applying concavity and convexity criteria along the boundaries of the objects. The segmented patches are classified into columns and beam/girder using a SVM classifier. Beams and girders are further classified using connectivity relations. Finally, shapes are classified, geometric parameters are retrieved, and the model is delivered in an IFC compliant format. This study is considered an end-to-end pipeline as only the very last step of delivering IFC objects has not been investigated, which can be considered a problem of technical implementation rather than research.

The study of Wang et al. [91] is on the verge of being within the scope of this work, as the generation of as-is BIMs of the building envelope and openings rather than a geometric and semantic building component reconstruction is studied. Nevertheless, such building envelope as-is BIMs are of great interest for sustainability applications such as acoustic, lighting and thermal [...] simulations. Other than the previous studies, Wang et al. initially use a region growing plane segmentation. The building envelope's contours are detected using an edge and boundary extraction algorithm. Any building component detected will be assigned a class label including wall, window, door, roof and foundation with subclasses [91]. Especially the rule-based classification algorithm seems to be limited as it uses simple rules and assumptions such as that all openings are closed during scanning, which is unrealistic for more complete point clouds from inside and outside the building. The delivered as-is BIM is a surface model, rather than a topologically correct building model.

The study most closely related to this work is conducted by H. Kim et al. [92]. In their study, a segmentation model trained with synthetic data is used to infer semantic labels, followed by an instance segmentation procedure using Euclidean clustering. After a planar patch segmentation, the vertices and edges are extracted by intersecting the planar patches per element. In case of incomplete scans, the dimension is inferred by referring to similar elements or default values. A graph network is used to model the relationships of the elements and a network update algorithm is applied to find missing relationships and to extend the elements according to related elements. The fundamental difference in our work is that Kim et al. use synthetic data. Similarly, strategies are applied to reconstruct the geometry of objects only partially scanned. In this context, the algorithm proposed by Kim et al. [92] to compare the dimension of similar objects to infer the dimension of partially scanned components seems promising, and the same applies for the proposed component relationship reconstruction method.

Similar to the previously discussed study, T. Kim et al. [93] propose to use a segmentation model based on PointNet [52] that has been trained with synthetic data of exhibition halls. A polygonal 3D model is reconstructed using RANSAC plane fitting. The topology is refined using a room decomposition strategy. The building model is then represented in the CityGML format. Although the described methodology seems sound, the description misses a lot of details and the use of CityGML seems to no be appropriate to represent building models of indoor scenes. This is also reflected by the fact that the study of Kim et al. [93] is the only one to use CityGML as an output data format.

In coherence with our work, it can be observed that there is no clear tendency to apply semantic segmentation as the first step. The reasons are manifold. Semantic segmentation architectures capable of efficiently processing large datasets have been developed in the last years and have not been available before. The availability of annotated datasets remains another hurdle. In many of the studies discussed a common limitation is either a limited generalization and the need to set and tune input parameters, which can be challenging when processing previously unseen data.

2.4.2 Partial pipelines

Beyond studies on the end-to-end scan-to-BIM process, many other studies cover one or multiple steps of the entire process. Two main categories could be identified when reviewing such studies: *Object detection and building element retrieval* and *Geometry extraction*. This structure reflects the typical sequence in scan-to-BIM pipelines, thus contributions in both fields could be combined with BIM authoring tools to form a complete pipeline.

Object recognition and building element retrieval

This category covers all studies that research methods for building component detection and building element retrieval (refer to Table 2.8). Other than semantic segmentation, this does not only involve a per-point assignment of a semantic label but other methods for object detection such as bounding box or centroid representations assigned with a semantic label [96], or even the retrieval of similar elements [97]. In contrast to the previous sections, at least two individual procedures, e.g. semantic label prediction and geometry retrieval, are combined.

Author	Year	Title	Short Description	Experimental results
Assi et al. [96]	2019	Energy function algorithm for detection of openings in indoor point clouds	Wall instances are transformed to binary images, windows are segmented by region growing and region classification using two energy functions	Evaluation on datasets of a museum and residential building, 100% detection of unobstructed windows, 69% overall.
Chen et al. [97]	2019	Exemplar-Based Building Element Retrieval from Point Clouds	Semi-automated method to retrieve similar elements in point clouds.	Experiments on point clouds of a five-storey academic building, precision 96%, recall 82%.
Krispel et al. [98]	2019	Formalizing Expert Knowledge for Building Information Models: Automated Identification of Electrical Wiring from 3D Scans	Retrieving power line hypotheses using implicit knowledge from standards and explicit knowledge from switches and sockets detected in images.	No quantitative evaluation.
Xu et al. [99]	2020	Automatic As-Built BIM with 3D Object Detection by Learning Building Structure Knowledge	3D object detection CNN learning corner features and structural knowledge.	On a synthetic dataset, an average precision of 56.01% is reported.
Xu et al. [77]	2021	Three-Dimensional Object Detection with Deep Neural Networks for Automatic As-Built Reconstruction	Extension of [99] towards more classes.	79.03 mIoU reported on SUNCG dataset.
Babacan et al. [100]	2017	Semantic segmentation of indoor point clouds using convolutional neural network	Fast CNN for indoor point cloud classification with synthetic and real training data to support subsequent plane extraction.	Overall accuracy of 81% for semantic segmentation on real office point cloud.

Table 2.8: Partial pipelines: Object detection and building element retrieval.

Xu et al. [99] study the problem of 3D object detection on a synthetic dataset. Beyond object-level features, the proposed method includes corners and structure knowledge, thus improving the detection accuracy in cluttered, noisy and complex environments. Results are only reported on a synthetic dataset, thus the transfer to real-world data remains to be evaluated. Xu et al. [77] extend the methodology towards other building component classes, proposing a two-stage network architecture for region segmentation and subsequent object detection. The method is evaluated on a large synthetic dataset of single-storey buildings. In both studies, the main limitation is that the method has not been tested on real-world data.

To detect openings, Assi et al. [96] perform opening segmentation by transforming the wall instances to a binary image. Subsequently, a region growing is applied. Each region is classified based on two energy functions, one of which excludes regions unlikely to be a window based on geometric configuration, the other describes the regions properties, identifying regions with rectangular boundaries as windows. The method takes a segmented point cloud as an input. For each wall, the centroid, Cartesian equation and the point indices must be known. The degree of automation seems limited, as multiple libraries and software are mentioned. As the authors state, the work is part of a larger project for scan-to-BIM automation.

Chen et al. [96] propose a method to retrieve similar building elements from a point cloud based on an initial selection by the user. First, a deep learning model is trained to predict a feature vector. Subsequently, K-means clustering and region growing are applied to retrieve individual elements. After the exemplar selection by the user, an element matching is performed by feature correlation computing and peak finding. Although this method is not fully automated, it provides an interesting approach to detect similar objects in point clouds, which can be beneficial to later reconstruction steps. By applying an automated method to find a seed building element, there is the potential to automate the method, especially within specific element categories.

An interesting but seldom studied subject is the retrieval of obscured objects such as power lines. Krispel et al. [98] propose a methodology for power line retrieval based on an initial BIM created using the method described in [80], followed by applying implicit knowledge formalized through shape grammar to infer power line hypotheses. These are validated using the detected switches, sockets and other visible parts of the electrical installation based on images. Despite being a conceptual study, there is potential to adopt the methodology. However, any rules to derive the power line location could be useless, since the compliance to these rules cannot be assured. Thus, the mentioned procedure should be supported by other data captured, e.g. thermal imaging or Ground penetrating radar (GPR).

Babacan et al. [100] demonstrate in their study that semantic segmentation would support the subsequent plane extraction, a notion that this work also shares and exploits. First, Babacan et al. [100] propose a fast CNN architecture for indoor point cloud classification using both synthetic and real training data, combined with a semantic clutter filtering. For planar extraction RANSAC [64] is applied, proving that the number of irrelevant planes declines if the point cloud has been previously segmented.

Geometry extraction

After object recognition and building element retrieval, methods for geometry extraction can be applied to retrieve the dimension, translation and rotation of the object observed. Such methods involve the

reconstruction of basic geometric representations, such as planar extraction of components in a previously semantically completed point cloud, as proposed in [101]. An overview is given in Table 2.9.

Author	Year	Title	Short Description	Experimental results
Coudron et al. [101]	2020	Semantic Extraction of Permanent Structures for the Reconstruction of Building Interiors from Point Clouds	Semantic scene completion for improved semantic plane fitting and planar structural component extraction.	Semantic scene completion: 69.5% mIoU, results of component extraction reported.
Petschnigg et al. [69]	2021	From a Point Cloud to a Simulation Model-Bayesian Segmentation and Entropy Based Uncertainty Estimation for 3D Modelling	Semantic segmentation using Bayesian neural networks, clustering, CAD object fitting and model delivery in Unreal Engine 4.	Comparison of Bayesian PointNet with PointNet, comparative evaluation of clustering algorithms, evaluation of retrieved model.
Bassier et al. [102]	2020	Topology Reconstruction of BIM Wall Objects from Point Cloud Data	Four step approach to retrieve topologically correct connections of three walls based on wall centre lines.	On scenes one and two of S3DIS [32], 86% of the connections were reconstructed with 76.8% recall and 92.2% precision on average.
Maalek et al. [103]	2019	Automatic Recognition of Common Structural Elements from Point Clouds for Automated Progress Monitoring and Dimensional Quality Control in Reinforced Concrete Construction	Floor, concrete column and rebar parametric surface model reconstruction from construction site data.	99% for precision and accuracy of the column extraction reported for an in-situ concrete construction site scanned every week.
Shi et al. [104]	2019	Semantic Geometric Modelling of Unstructured Indoor Point Cloud	Reconstruction of indoor scenes by hybrid 3D segmentation, room extraction and opening recognition.	90% precision and 86% recall reported on synthetic and real-world scenes of an academic building.
Xiong et al. [105]	2013	Automatic creation of semantically rich 3D building models from laser scanner data	Planar patch segmentation, classification, occlusion handling and opening detection.	Precision and recall of more than 82% reported for patch segmentation and classification algorithm.
Chen et al. [106]	2019	Deep Learning Approach to Point Cloud Scene Understanding for Automated Scan to 3D Reconstruction	Edge-based object segmentation, classification, bounding box merging and 3D CAD object matching.	Evaluation on S3DIS dataset [65], mIoU of 50.3% for semantic segmentation, mean object accuracy 52.4%.
Armeni et al. [32]	2016	3D Semantic Parsing of Large-Scale Indoor Spaces	Parsing the point cloud into spaces using a density histogram and convolution with filter banks, learning-based building element detection.	Quantitative evaluation on dataset of three buildings, overall accuracy of semantic element detection 49.93%.

Table 2.9: Partial pipelines: Geometry extraction.

A very common procedure to extract geometry out of point clouds is plane segmentation. However, performed on a raw point cloud plane segmentation is likely to segment false positives or even plane artefacts in highly cluttered environments. To overcome these problems, Coudron et al. [101] propose

to apply a semantic scene completion before plane segmentation and planar structural component extraction. The semantic scene completion allows the clutter to be removed, supporting the plane extraction and plane labelling.

Beyond fitting geometric primitives or extracting geometric features, the following studies propose partial pipelines covering multiple steps of the overall scan-to-BIM problem.

Petschnigg et al. [69] propose a pipeline for point cloud segmentation and geometry extraction combining a Bayesian neural network for semantic segmentation based on PointNet [52], clustering and 3D CAD model fitting to retrieve the object's pose, roll, pitch and yaw. The model is delivered in the Unreal Engine [107] for further simulation studies. Although the study is focussed on the factory planning, among the retrieved objects typical structural components are encountered. The framework itself can be adapted to other types of structures and is similar to our approach as it applies semantic segmentation and clustering. Fitting CAD models seems appropriate to industrial environments with previously established types of objects present in the scene, yet inflexible regarding the variety of previously unseen structural objects in buildings. Hence, in our work, other strategies to retrieve the objects' geometries will be applied.

Although focusing on concrete construction progress monitoring, Maalek et al. [103] propose an interesting method for parametric surface reconstruction of floors, concrete columns and rebars. After a plane segmentation, the parametric surface is retrieved by identifying the planes relevant to an object instance, with specific approaches to handle rebar protruding from columns. Note that during construction the rebar is still visible. No methods to capture obscured objects have been applied.

Similar to the previous study, Chen et al.'s [106] study is motivated by the necessity of partially automating construction progress estimation. However, essentially a method is presented for edge-based object segmentation, object classification, object bounding box merging and 3D CAD object matching. For object segmentation, the point cloud is subsampled and converted into a graph representation, with vertices representing points and edges representing the point distance. With additional edge features (point proximity, colour, etc.) a neural network architecture is developed and trained for object cluster segmentation. The object clusters are classified using a modified PointNet [52] with local context. Object bounding boxes of the same class are merged based on spatial relations such as distance and direction. Finally, PointNet [52] is used to compare feature vectors of the observed point cluster and synthetic points of the CAD models, thus identifying the best fitting CAD object, which is aligned to the point patch.

Despite being one of the first studies to deal with aligned point clouds of entire building storeys, Armeni et al. [32] propose a quite modern approach to deliver semantic spaces and object bounding boxes. First, the entire scene is disjoint into spaces assigned with a semantic label (floor, hallway). Subsequently, building elements including wall, door, window, column, beam, floor and ceiling are detected using a learning-based SVM approach.

To deliver a semantic geometric representation of the indoor point cloud input, Shi et al. [104] propose a four-step pipeline comprising data preprocessing, hybrid 3D segmentation, room layout reconstruction and enriched wall-surfaces object detection. During preprocessing the normals and surface curvature is calculated. These features are used to select seed points for RANSAC plane fitting, i.e. points with low surface curvature. Next, rooms are reconstructed applying wall plane refinement, room clustering and boundary estimation. Finally, opening elements such as doors and windows are detected.

Xiong et al. [105] propose to use a region growing algorithm with planarity constraint to segment planar patches. A learning-based algorithm is used to assign a semantic label (wall, floor, ceiling, clutter) to each patch. Finally, a ray tracing-based routine is applied to handle occlusions, and openings such as windows and doors are detected. Although the proposed approach to deal with occluded regions seems to be a valuable addition to the scan-to-BIM process, the proposed strategy to add points in occluded regions to the point cloud seems quite ineffective, as it represents the occlusions using points with respective demand of memory and processing capabilities. Applying a spatial reasoning to the occluded regions on a higher representation such as planes with boundaries or bounding boxes seems more appropriate and beneficial to scan-to-BIM pipelines.

Beyond the geometric reconstruction, an important task is to deliver a set of BIM objects with correct topology, i.e. correct connections and relationships along corners and boundaries of rooms. Bassier et al. [102] study the problem of topology reconstruction in the case of walls. After a preprocessing stage where IfcBuildingStorey and IfcSpace objects are reconstructed, a four-step approach is proposed: after retrieving the neighbourhood, candidate connections are established, from which the topologically most sound is selected. Finally, walls are reconstructed considering the retrieved connections. Candidate connections are also established for three-wall configurations. By defining a cost function a hierarchy of connections is established by a weight function.

2.4.3 Technical aspects and conclusions

In this section, the previously discussed studies will be analyzed from a technical perspective. Research gaps will then be identified by combining this technical analysis with a broader view of industry needs.

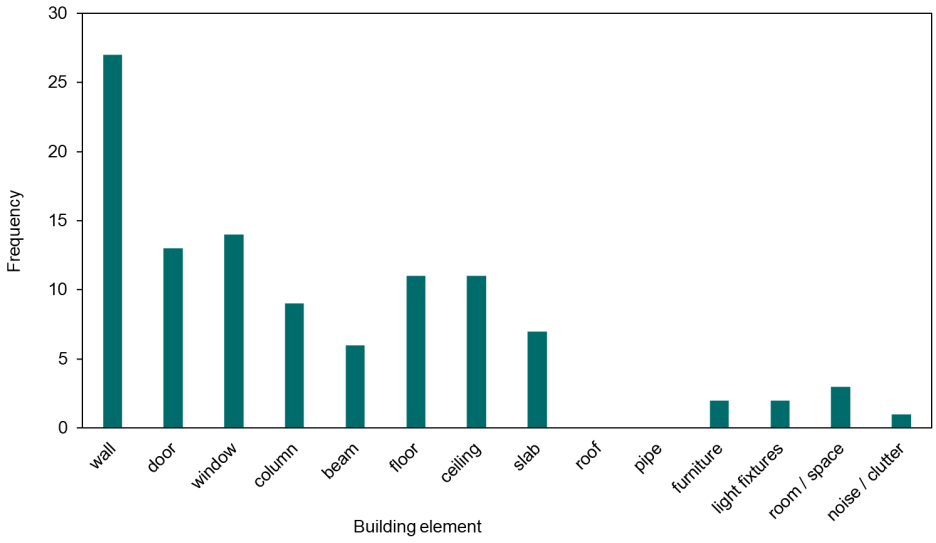


Figure 2.6: Components examined in the reviewed scan-to-BIM research.

Figure 2.6 illustrates that out of the studies reviewed, 27 address the reconstruction of walls. Despite their similar significance in understanding a building's structure, doors and windows, as subordinate elements to walls, receive less attention with only 13 and 14 studies, respectively. Other structural components like columns (n = 9), beams (n = 6), and slabs (n = 7) are also infrequently discussed. The studies do not clearly differentiate between floors, ceilings and slabs. From the perspective of structural engineering, slabs are typically defined as the structural, usually cuboidal, layers situated between floors and ceilings. This distinction is essential for developing a structural model of the observed building. Remarkably, roofs and pipes are not examined in any of the reviewed studies. This omission might be due to the frequent exclusion of industrial buildings, which often contain pipes. Only three studies focus on reconstructing room or space objects. Although noise and clutter are generally not reconstructed, one study considers them during the semantic segmentation phase. To process real-world scenes, however, clutter needs to be considered to avoid over segmentation of objects with unknown classes.

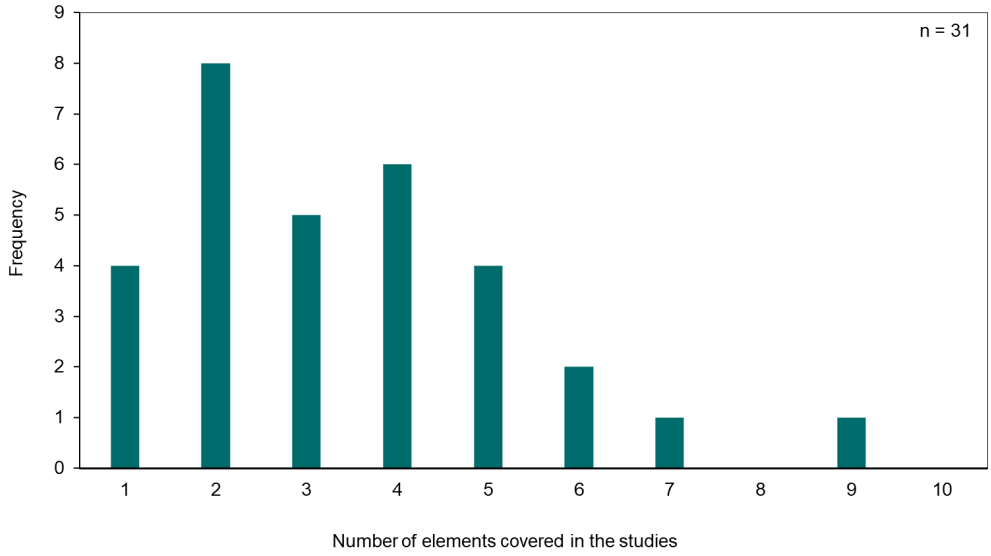


Figure 2.7: Number of elements covered in the scan-to-BIM studies reviewed.

The most obvious research gap can be identified in Figure 2.7, which shows the number of elements covered in the studies. It can be observed that eight studies out of 31 cover two elements, six cover four elements. This reveals that there is a research gap in studies seeking to reconstruct multiple elements. More specifically, scalable and adaptive frameworks are needed to reconstruct more elements.

Several studies propose an initial semantic segmentation of the input point cloud [92, 93, 69], an approach that we will follow in this work as well. For our work, especially the histogram-based information extraction as presented in [87] for wall reconstruction and level identification [81] will be adopted, e.g. as a seed finding algorithm and for level fitting. One of the rather seldom studied topics are methods for topology reconstruction. A method for wall topology reconstruction is presented in [102] and the concept of wall centreline intersections for topology refinement will be adopted in our work, too.

Beyond that, the main research gaps are identified as a lack of scan-to-BIM pipelines to process multiple

elements, with a limitation of many studies being to generalize to other elements, as well as a lack of more general, high-level frameworks. Although a high quality of reconstruction could be achieved by many studies, most of them do not provide a topologically watertight reconstruction (except [84], [102], [82]). To provide functional BIMs, topological correctness is crucial to ensure correct object relations and connections, especially for subsequent use cases.

2.5 Research gaps and thesis contributions

As the scope of this work is to develop a comprehensive pipeline to transfer point clouds into BIMs, it is obligatory to summarize the research gaps with respect to all the fields studied in this literature review.

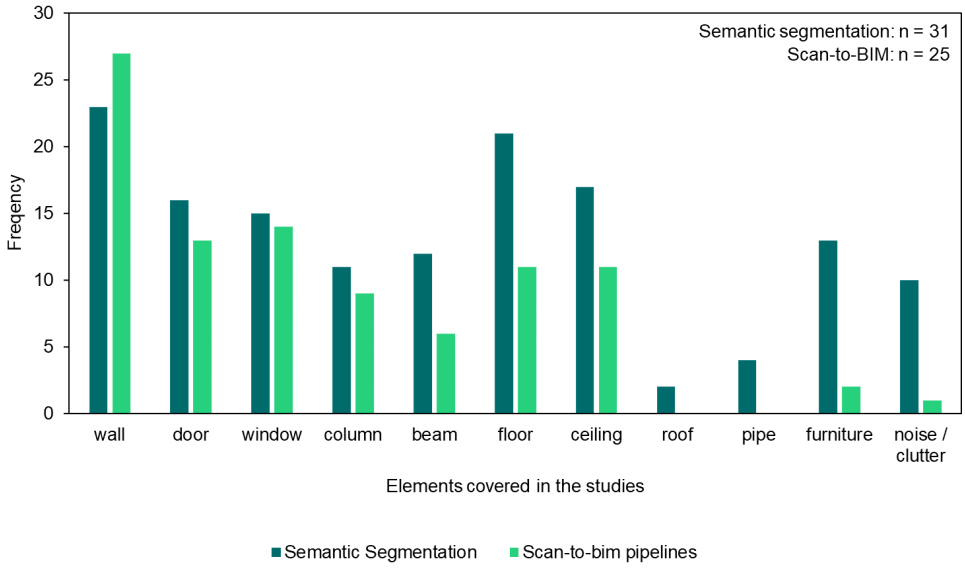


Figure 2.8: Components examined both in semantic segmentation and scan-to-BIM pipeline research.

Schönfelder et al. [28] highlight a significant gap in the literature; specifically, only a few studies comprehensively address the scan-to-BIM process from point cloud data to the final BIM model. This gap is further evidenced by the additional literature reviewed in this study. A similar observation applies to the field of building topology reconstruction, which is rarely explored. Furthermore, Schönfelder et al. advocate for the adoption of advanced deep learning techniques, particularly transformer models, to address these deficiencies. This work aims to tackle all three identified research gaps.

Beyond merely developing a comprehensive scan-to-BIM pipeline with topology reconstruction, this work proposes semantic segmentation using a transformer architecture, detailed in chapter 3. According to Figure 2.8, existing studies cover structural components for both semantic segmentation and

scan-to-BIM processes. Clearly there is a substantial alignment between these fields, with numerous studies proposing methods for the semantic segmentation of walls and scan-to-BIM reconstruction. This alignment extends to most other component classes as well. Integrated with advancements in geometry processing (subsection 2.3.3) and automated scan-to-BIM techniques (subsection 2.4.3), the primary research gaps are summarized as follows:

- Scan-to-BIM methods frequently follow the **Manhattan-world assumption** [21] which naturally excludes non axis-aligned data, curved walls, sloped ceilings, etc. On the one hand, it is easy to axis-align the input data and processing under the Manhattan-world assumption simplifies the algorithms and can lead to more efficiency. On the other hand, such methods are less general and adaptive to other cases. The reconstruction of non-axis-aligned objects could be erroneous or fail completely. Thus, methods need to be developed and applied that reliably identify erroneous or ignored objects and apply corrective methods to ensure a complete and sound reconstruction.

- Many studies **focus only on a few components**. Given the varieties of buildings and the components encountered there, methods adaptive to other components are needed. This requires scan-to-BIM frameworks that provide implementations of robust processing algorithms. With such frameworks, pipelines for multiple objects can be developed and implemented.

- The degree of **ML** / **AI methods** applied in pipelines after semantic segmentation could still be expanded. This includes learning-based methods for geometric segmentation, clustering, topology refinement, etc. Beyond the mere application of such methods, datasets are required to train the respective models.

- It would be beneficial to increase research efforts towards **end-to-end machine learning approaches** for scan-to-BIM. Such approaches would use manually created BIMs and the respective point cloud as training data. The main benefit is, that BIMs of building point clouds can be more easily created manually than per-point annotations as required for semantic segmentation. As-built BIMs and respective point clouds might even exist for recently finished buildings, thus providing training data for end-to-end approaches without any extra effort.

- Although gains in **semantic segmentation accuracy and efficiency** have been reported over the past years, the process of training and inference remains computationally expensive, as dense point clouds need to be fed through the network multiple times. A higher level representation of the input point clouds, such as graph representations, respective datasets and neural network architectures, could help to boost the efficiency of semantic segmentation.

- **Datasets and methods for entire buildings**, as discussed in the reviewed research, revealed a focus on indoor scenes. Thus, the focus should be shifted towards datasets, algorithms and pipelines for complete buildings of varying topology, including office, residential and hospital buildings.

- Due to the fact that incomplete data as an input and/or within the pipeline processing is likely to occur, **methods to deal with incomplete data** need to be established. Many studies in this literature review develop their method on perfect (synthetic) input data or perfect results of intermediate process steps. Although this is often the best option to advance isolated methods in the overall process, it limits the practical applicability of such methods.

- BIMs as a result of scan-to-BIM pipelines need to be **topologically watertight**. Methods to ensure this need to be established. With topologically correct reconstruction, relationships of the components can be represented in BIM correctly, ensuring functional resulting BIMs.

- The mIoU metric is commonly used to evaluate the results of semantic segmentation algorithms. In the studies reviewed, the accuracy of scan-to-BIM reconstructions has been evaluated using qualitative visual inspection, quantitative evaluation such as mean distance of points and BIM geometry, mIoU on bounding boxes and other metrics. The main research gaps are that a geometric mIoU penalizes misalignment, wrong dimensions and wrong orientation to the same extent without providing details on the type of error. Thus, **more advanced evaluation metrics** could help to evaluate scan-to-BIM results in a more detailed way, helping to improve the methods based on a dedicated evaluation.

- Many studies have been conducted with a computer science background. Unfortunately, some of these studies define and solve problems that are not practically relevant. Formalizing and **integrating construction engineering knowledge** into interdisciplinary research would help to provide practically relevant and sometimes more straightforward solutions.

Our research will leverage the Manhattan-world [21] assumption to gain efficiency in the reconstruction procedures. We aim to show that this approach yields superior reconstruction quality, despite neglecting specific corner cases. However, this is a viable option from a civil engineering perspective. This study intends to go beyond the constraints of earlier studies by introducing an adaptive framework. Emphasis will be placed on structural elements and environments with minimal clutter, pertinent to civil engineering applications. BIMs that offer detailed reconstructions of these structural elements are particularly valuable. Recognizing incomplete input data as a common challenge, our methods are designed to address such gaps. We will ensure the reconstruction is topologically watertight. Moreover, this research contributes to the development of more precise evaluation metrics for scan-to-BIM processes. My background as a trained civil engineer will be integrated into the framework's design.

This work aims to make the following contributions in addressing the identified research gaps:

- **Data annotation guidelines** and a **dataset** of empty buildings with rich semantic annotations and ground truth BIMs to leverage training of Machine Learning methods with a focus on structural components.

- **Improved geometry reconstruction algorithms** for efficient filtering, and primitive fitting on incomplete data.

- **Topology reconstruction algorithms** to provide topologically correct wall instances.

- **A general framework including two Python libraries** dedicated to geometric reconstruction and open BIM authoring facilitating the development of scan-to-BIM pipelines to reconstruct multiple objects.

- **A comprehensive scan-to-BIM pipeline** using the developed methods to reconstruct walls, doors and columns including implementation, testing and evaluation.

- **Metrics** dedicated to describing the error in the reconstruction explicitly, overcoming some unde-

sired behaviour of frequently used metrics such as the IoU.

In the following chapters, the approaches to key process steps including semantic segmentation, geometric feature extraction, open BIM modelling and pipeline integration will be introduced.

3 Semantic segmentation of point clouds

3.1 Introduction

In the previous section, a wide range of approaches to semantic segmentation were discussed that can be distinguished into knowledge-based and data-driven approaches. A data-driven deep learning approach using a transformer architecture as proposed by [43] was chosen. Besides the architecture itself, deep learning approaches require extensive training data to learn features that distinguish the classes, thus facilitating a high-quality prediction.

The training process will be briefly described here, and more details can be found in our jointly written paper [22]. The contribution of this work is an annotation framework and guideline aiming to increase the technical quality of the annotations and annotated data. Dedicated effort was taken to inject technical insights of the civil engineering domain into the annotation process and semantic label definition.

The following sections will thus cover the process of data annotation, describe the used deep learning network architecture and finally, the results achieved.

3.2 A framework for data annotation

To provide high-quality training data it must be annotated manually. In this procedure, two main challenges have been identified. (i) It is a demanding yet repetitive task, which makes it error-prone due to a lack of concentration and motivation. (ii) Depending on the skills and background of the person annotating, the distinction of classes may vary significantly, which can result in inconsistent annotation, thus affecting the learning of the model. Naturally, it is hard to overcome the challenges of insufficient motivation and skill. However, providing comprehensive data annotation guidelines focusing on the features and distinction of the classes can help to improve the consistency of the annotation. These guidelines will be introduced in the following section.

3.2.1 Data annotation guidelines

The annotation guidelines are designed to serve as a common ground for labelling both point clouds and RGB-D data, so special care was taken while defining the different classes and their annotation guidelines. The main goal was to provide consistent annotation on different data domains (3D point clouds, RGB). To the knowledge of the author, this annotation framework and the delivered dataset is the first containing different data modalities of the same object. In this work, however, the focus

is on point cloud semantic segmentation for scan-to-BIM. Thus, the following remarks mainly cover point cloud annotation. The complete annotation guidelines can be found in Annex 4. Note, that the guidelines are subject to constant updates. Thus, it is advised to consult this repository: `https://gitlab.rhrk.uni-kl.de/kaufmann/humantech-data-annotation` for the most recent version. Beyond the building category as introduced here, the annotation guidelines even contain labels for the interior and construction category as explained in the following paragraph. More detail on the building category and especially RGB-D annotation can be found in the respective research paper [17] published by the author.

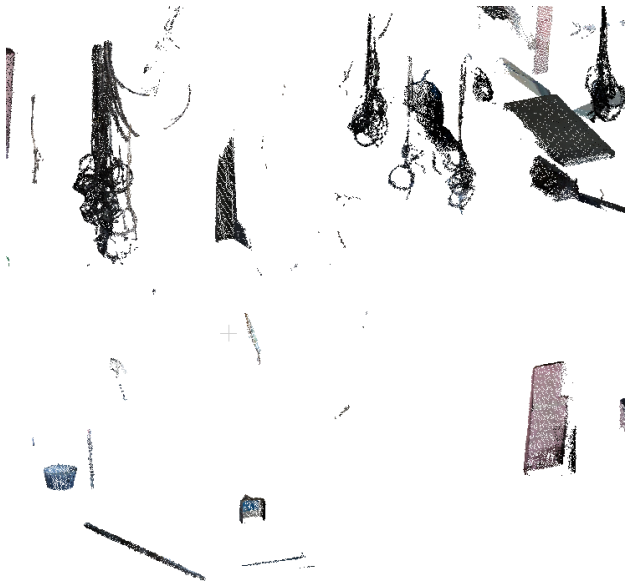


Figure 3.1: Sample of points annotated as clutter. The clutter objects include cables during installation, boards and other objects related to construction.

Initially, the classes observed had to be distinguished in various categories to organize the class definitions. Three different categories were observed when defining the labelled classes:

- Building Category: this includes the subset of labels that belong to the building itself, such as walls, doors, columns, windows, etc. All items under the building category are fixed to the building.
- Construction Category: this category includes objects used in construction, such as formwork and scaffolding.
- Interior Category: this includes a large set of labels that can be seen inside a building or on a construction site, such as tables, chairs, computer hardware, fire extinguishers, etc. Interior objects are movable and thus not fixed to the building.

In addition to those categories, some special labels were defined for miscellaneous categories such as invalid data. In point clouds, clutter, noise, and unidentifiable artefacts should be annotated as invalid

data as seen in Figure 3.1. Those labels are important when training and validating neural networks to reduce false positives and are required for real-scene data.

The guidelines introduced for annotation differ from existing datasets by conforming to the IFC classes whenever possible and creating a common framework for both 3D (point clouds) and 2D (RGB images and RGB-D images) labels. This guideline can thus be used for labelling data for different stages of construction and making it possible to generate BIM models from the data. Additionally, a consistent annotation of 2D and 3D data will facilitate the data comparison on a semantic level and thus allow us to investigate the specific advantages of different sensors.

From the 43 labels defined and described in the annotation guidelines, the label classes wall, door, window, floor, ceiling, pipe and pipe fitting will be presented here to describe the novelty of the annotation guidelines. The specific entity descriptions per label linking to the IFC standard [3], annotation rules and examples are given.

Walls

In the context of this work, **walls** consist of two parallel surfaces that are vertical, as described in the IFC entity definition: *'The wall represents a vertical construction that bounds or subdivides spaces. Walls are usually vertical, or nearly vertical, planar elements, often designed to bear structural loads. A wall is however not required to be load bearing.'* [3]. Note, that in contemporary architecture walls with a more complex geometry are encountered. As these are not the standard, such geometries are not in scope of this work and not to be annotated.

No additional description is required for walls, and examples of such annotations can be seen in Figure 3.2. Specific annotation rules for walls are defined:

- Preserve details on the edges and select them with utmost precision.
- Walls may be made of glass, and larger glass areas should be considered as walls.
- Typically, walls are supported by a floor or slab, and their vertical limit is the ceiling or bottom slab surface.
- Walls usually connect the floor and ceiling, except for openings and small edges.
- If a vertical member is connected to the floor but lacks a horizontal top surface below the ceiling, it is still annotated as a wall.

The first statement seems trivial but a precise segmentation along the edges of walls and horizontal surfaces is hard to achieve (see Figure 3.2 right). To support this process, RANSAC [64] plane fitting can be used to refine the segmentation by fitting planes to the batches. The outliers can be either assigned the label of the adjacent objects or clutter, respectively.

As mentioned, walls can also be made of glass or glass panes, which can be used for interior walls to separate rooms and spaces in a translucent way. In such cases, it is important to specify the difference

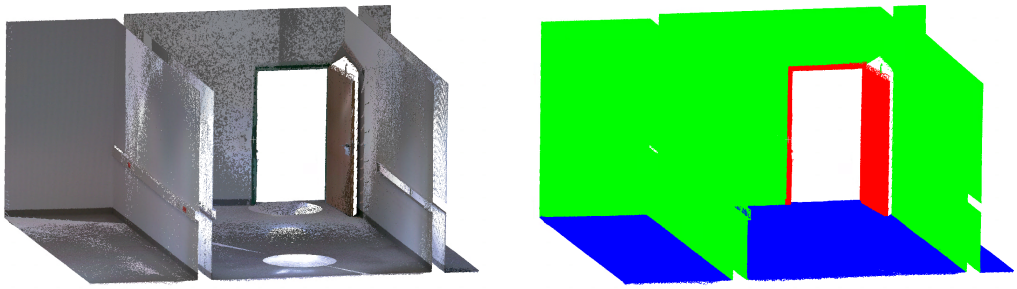


Figure 3.2: Example of walls, floor and door annotation. Left: original texture. Right: annotation with wall and floor labels separation along the edges.

between glass walls and doors. According to the proposed guideline, a door may include an area of fixed glass panes. If the area of the fixed glass panes or other panels around the door leaf does not exceed 1.5 times the door leaf area, it will be annotated as a door. In other cases where the fixed glass panes are larger than this threshold, they must be annotated as walls.

Skirting boards are part of the wall class. Unless there is a dedicated class for the objects, all objects attached to the wall such as pictures, light switches, etc. are annotated as wall.

Doors and Windows

Doors and **windows** can be distinguished by their nature given in the *IfcDoor* entity definition: *'The door is a building element that is predominately used to provide controlled access for people and goods. It includes constructions with hinged, pivoted, sliding, and additionally revolving and folding operations. A door consists of a lining and one or several panels'*, and windows are primarily used *'to provide natural light and fresh air'* [3] and may be horizontal or vertical such as skylights. While both door and window may contain a lining there are cases where only the glass pane will be fitted directly to the wall without a visible lining. In these cases, only the panel would be annotated as window.

Technically, it is possible to annotate openings in images, although the mask may obscure objects visible through the opening. However, annotating openings in point clouds is not possible since there are only points on substantial objects and not on voids. Hence, void objects or classes that are not represented by data points cannot be annotated. Therefore, space or opening objects are not included in the proposed guidelines and are excluded from annotation, although these classes may be relevant for detecting falling hazards and other workflows. In later processes, such as scan-to-BIM workflows, openings must be identified based on the objects in which the openings are located, such as walls and slabs. In cases where a glass pane has no visible lining, it cannot be annotated in point clouds since glass panes are only present as artefacts, if at all.

Floor and Ceiling

The examples presented above are strictly related to the IFC ontology as there are equivalent classes in the IFC standard. For many other classes, such as furniture and construction equipment, material and even some building parts, only generic IFC classes exist. As an example, the classes floor and ceiling will be explained in the following.

For both **floor** and **ceiling** the IFC class *IfcCovering* applies, which is defined as *'an element covering other elements while being fully dependent on the element covered . . .'* [3]. Obviously, this applies to floor and ceiling, which are fixed to the slab as the structural member. As the generic class definition is not sufficient, additional descriptions are required to define the entity as follows:

- Floors are the bottom horizontal enclosing element of spaces.
- Ceilings are the top horizontal enclosing element of spaces. Ceilings include suspended ceilings and coverings of structural elements.

Both definitions indicate that the floor and ceiling cover the structural elements or installations, as per the *IfcCovering*, but also define that any layer other than the slab enclosing a space horizontally should be annotated as either floor or ceiling. Typically, both the floor and ceiling are horizontal, and inclined surfaces should be annotated with a different class, such as ramp or roof, if applicable. Whenever the structural component is visible, e.g. an exposed slab, the respective label should be chosen. However, in point clouds, it can be difficult to distinguish these objects if there are no obvious visual features, such as rough concrete surface and texture. In such cases, additional documentation may provide more information about the state of the building, such as if it is still under construction.

Pipe Networks

Pipe networks consist of pipe segments and fittings. Hence, three classes are introduced to annotate respective data: Pipe segment vertical, Pipe segment horizontal and pipe fitting. Vertical and horizontal pipe segments are distinguished to facilitate a better segmentation and easier processing in scan-to-BIM pipelines. Pipe segments are used to *'typically join two sections of a piping network'* according to the IFC entity definition of the *IfcPipeSegment* class whereas an *IfcPipeFitting* connects individual pipe segments [3].

Annotating pipe fittings in point clouds presents a unique challenge. This difficulty arises particularly when the point density is insufficient to clearly delineate the juncture between the pipe fitting and the pipe segment. In scenarios where this demarcation is indiscernible, the end of the pipe fitting is determined by the alignment of its axis with that of the pipe segment. Specifically, this occurs either when the curvature of the pipe fitting ceases or when the linear axis of the pipe segment transitions into a curve. In contrast, due to the finer resolution of image pixels, the separation line between pipe fitting and segment is more likely to be discernible in RGB data annotation. When this line is not visible, the aforementioned principle is applied. If the separation line remains elusive, it should be marked at a right angle to the adjoining pipe segment of the fitting. Figure 3.3 illustrates the method for annotating pipes and pipe fittings in point clouds.

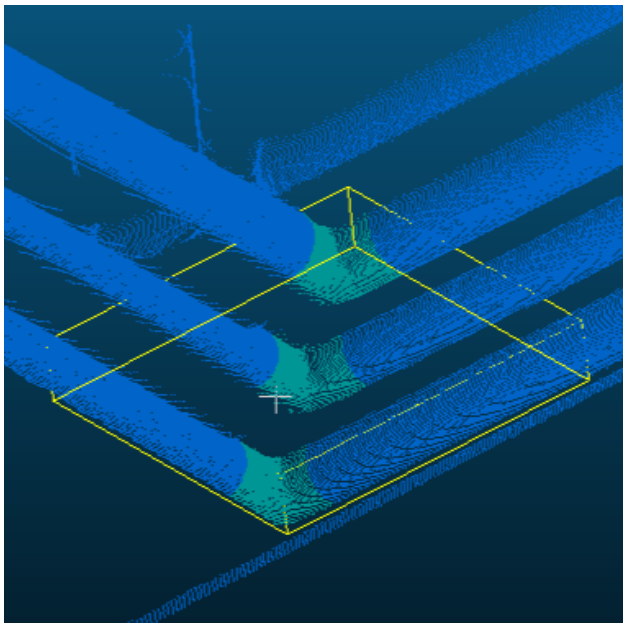


Figure 3.3: Examples of horizontal pipes and pipe fittings in point clouds: Pipes and pipe fittings are labelled in different colours in the point cloud.

3.2.2 Partially automated and manual data annotation

Data annotation can be partially automated. One approach is to use a ground truth BIM to assign labels to the respective point cloud. This can leverage the annotation output but there are still limitations to the automated process that require manual intervention. The procedure to automatically obtain labels from a ground truth BIM model involves transferring the BIM geometric representation into a bounding box, followed by identifying all points inside the bounding box. The procedure and the resulting labelled point cloud is shown in Figure 3.4. As part of the procedure, the bounding box is enlarged by an offset to ensure that slight misalignment of the point cloud and the BIM model do not result in a loss of labels. However, it is likely that points of adjacent objects are assigned the wrong label, e.g. slab points adjacent to the wall are assigned the wall class. This could be overcome by applying a more complex procedure including plane fitting to exclude adjacent points from other objects.

Thus, manual annotation remains the preferred procedure. In the context of this work, CloudCompare [4] is used which provides functions for the manual segmentation of point clouds. The labels are stored in a scalar field, i.e. an integer representing the class is assigned to each point. The annotations can be visualized using colour maps. For training, the point cloud is stored in the LAS file format or the compressed LAZ format [1]. The labels are exported into a separate NumPy array [108], which has proven to be the most efficient procedure. The full process is covered in a tutorial video [2].

[1] https://www.ogc.org/standard/las/
[2] https://youtu.be/n-FK7MJuiV0?si=y1ORJhp60CLBei2M

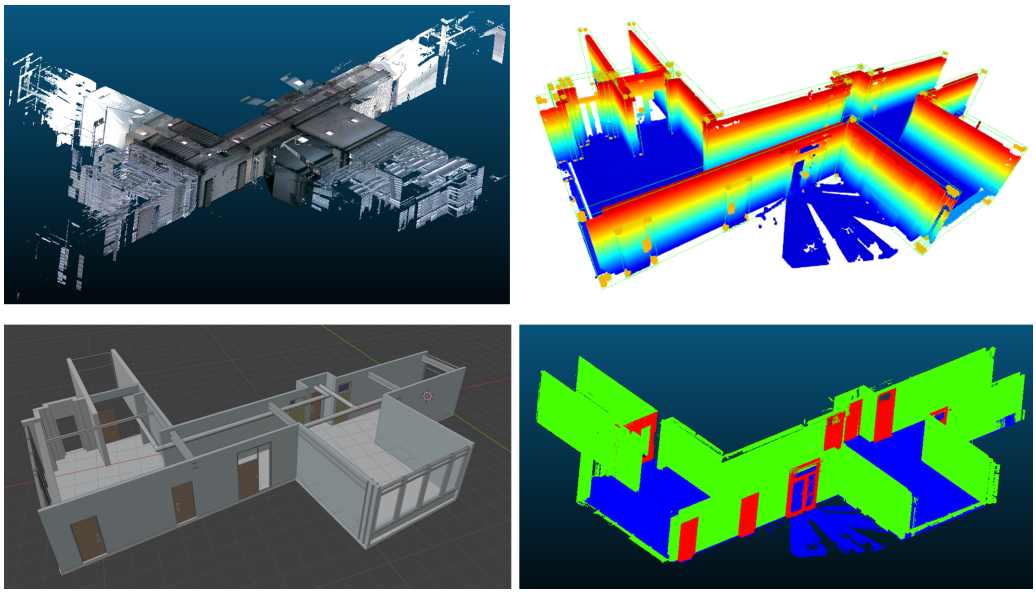


Figure 3.4: IFC to label. Top left: original point cloud, Top right: bounding boxes and point cloud, Bottom left: BIM model, Bottom right: retrieved labels.

3.3 Dataset

Datasets that are openly accessible, like S3DIS [32] and ScanNet [31], form the cornerstone of many research studies in semantic segmentation. When focusing on the goals of this study, which include developing a process for converting building point clouds into BIMs with an emphasis on structural elements, it is evident that these datasets have specific drawbacks. The datasets mainly comprise indoor scenes, such as those in hotel rooms and office buildings, often crowded with furnishings and other items. This setup means that in these scenarios, items like furniture, attachments, curtains, and more can obstruct the view of structural components, potentially resulting in partial data if the focus is solely on structural component categories. The labels in these datasets predominantly address visible items, categorizing furniture, decorative pieces, installations, and similar objects.

Beyond the aforementioned datasets, the one presented with the Computer Vision in the Built Environment (CV4AEC) scan-to-BIM challenge `https://cv4aec.github.io/` has been used in this work. Unfortunately, the CV4AEC dataset only provides a coarse BIM model as a ground truth, which even suffers from severe misalignment (see Figure 3.5). Thus, all 25 scenes published in the 2023 challenge have been manually annotated using the provided annotation guidelines. This dataset mainly comprises office buildings and three scenes of an underground car park. Obviously, interior objects including furniture and decorative objects are present in the dataset. Some scenes, however, present unobscured structural components such as walls, beams and columns, which adds valuable features to the aforementioned datasets. Overall, the dataset encompasses 2,209,019,107 points, with 1,711,578,939 annotated semantic classes and 497,440,168 labelled as clutter.

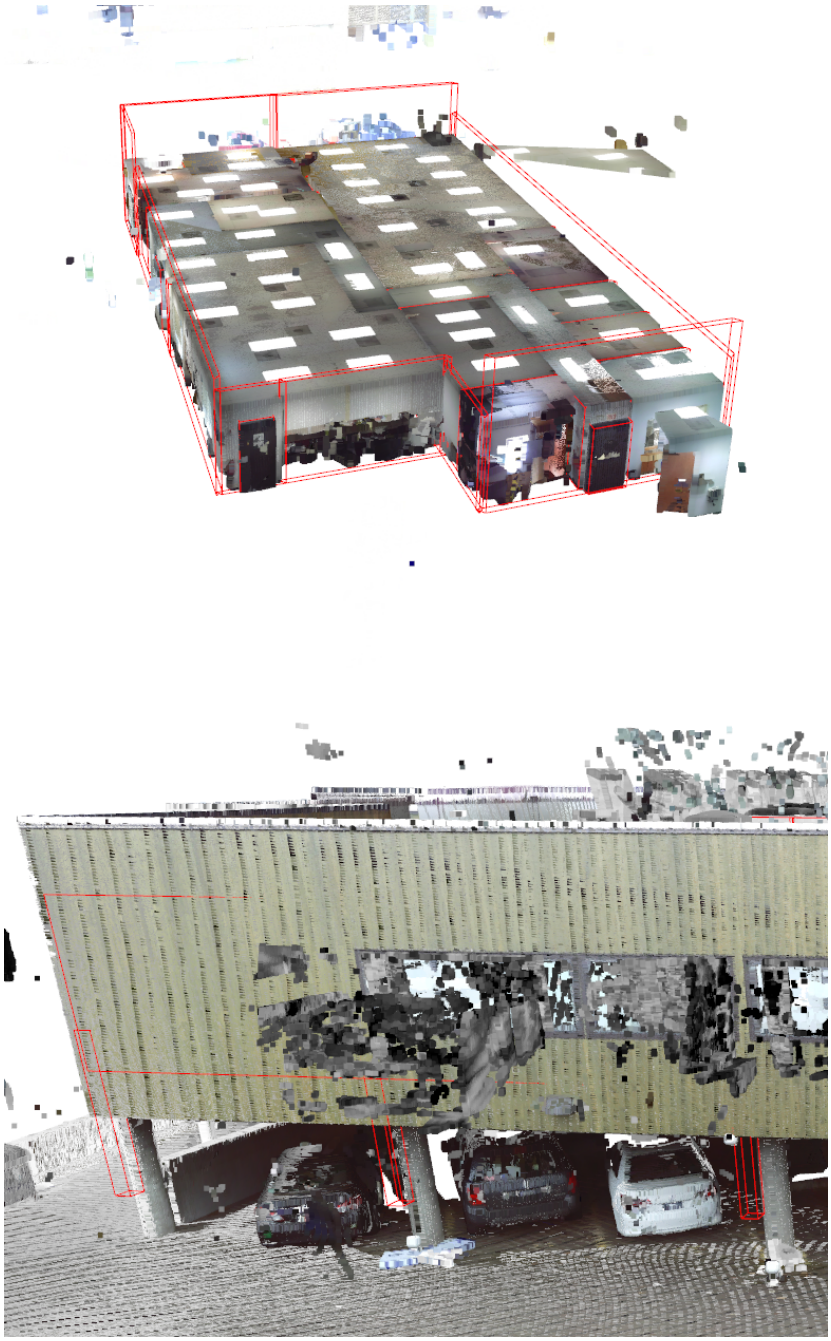


Figure 3.5: Ground truth and points of the CV4AEC dataset. The ground truth is presented as red bounding boxes. Left: misalignment of walls. Right: misalignment of columns.

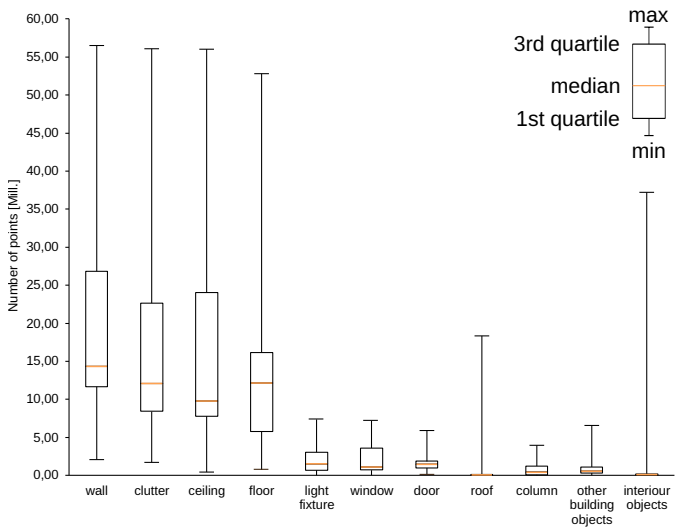


Figure 3.6: Distribution of label classes in the manually annotated CV4AEC dataset. The whiskers display the range of the minimum and maximum values, the box displays the range of the third and fourth quartile, the Median is displayed in orange.

The distribution of the labels classes is presented in Figure 3.6. It can be observed that the biggest portion of points (n = 576,576,065 or 26.10%) have been assigned the wall class, followed by clutter corresponding to 22.52%. The classes wall, clutter, ceiling, floor, light fixture, window, door and column have a similar distribution, whereas only roof points, representing 1.92% of the dataset, have an uneven distribution. This is related to the fact that an inclined roof's points are only present in one scene. The same applies to interior objects that have been annotated in some scenes and assigned the clutter class in other scenes. This can be ignored, as only building objects are of interest in this work. Other building objects encompass miscellaneous classes such as pavement, pipe segments, fittings, trusses, etc. This group has been condensed as it only represents 1.29% of the entire dataset.

More details including screenshots of the point clouds can be found in Annex 5. An interesting observation is inconsistencies in the colours. Some scenes provide RGB colour values, others grey scale and some do not include colour values at all.

Although overall 2.2 billion points with semantic annotations are available in the CV4AEC dataset, the limitation of occupied indoor scenes remains (refer to Annex 5). Thus, the goal was set to capture and annotate data of unoccupied indoor areas including the outside surfaces of the buildings. Such data was captured in an abandoned hospital building located in Weingarten, Germany. Three buildings were captured using a Leica BLK 360 1st generation laser scanner. One building included patients' rooms, one the remainders of an intensive care unit, and one surgery rooms. Although the 14 Nothelfer Hospital, Weingarten, Germany (14NH) dataset involves multiple modalities including 360-degree RGB-D data, data of a mobile scanning robot and of an Unmanned Aerial Vehicle (UAV), the data of the Leica BLK 360 is primarily used in this work, as it is the most accurate and similar to the other datasets available. Overall, the dataset encompasses 644,928,003 points, with 626,424,047 class labels and

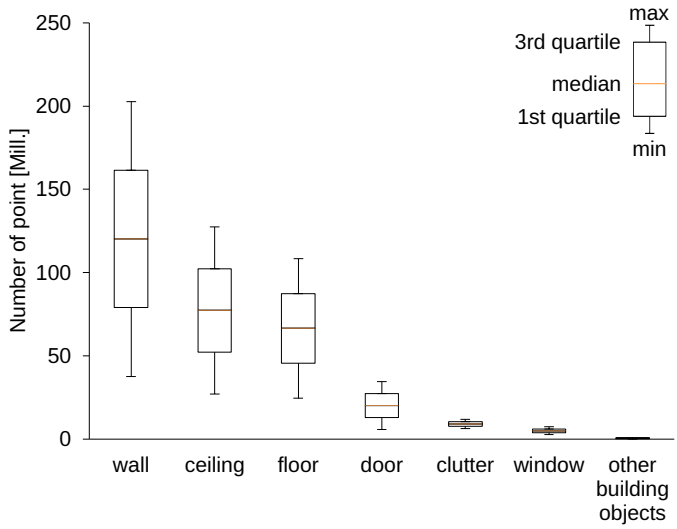


Figure 3.7: Distribution of label classes in the manually annotated 14NH dataset. The whiskers display the range of the minimum and maximum values, the box displays the range of the third and fourth quartile, the Median is displayed in orange.

18,503,956 labelled as clutter. More details on the distribution of the classes can be found in Figure 3.7. Compared to the CV4AEC dataset less clutter is present, as the rooms were empty during scanning. Regarding the main labels wall, ceiling and floor the distribution and ratio is comparable.

Additional ground truth BIMs serve as a common ground to evaluate scan-to-BIM pipelines. Screenshots of the point clouds and ground truth BIMs can be found in Annex 6.

With a sufficient amount of training and validation data available, the next step is to set up a segmentation model architecture and train it.

3.4 Point transformer-based segmentation model

The literature study revealed that PointTransformerV2 [43] is the architecture with leading results in several benchmark challenges. However, this is not the only rationale for using this architecture in this work. To underpin this reasoning, some considerations on the concept of transformer architectures are helpful.

During the evolution of neural networks, it became obvious that the concept of memory would be helpful to facilitate advanced reasoning capabilities, as this concept would allow the network to interconnect different features in the data. An intuitive example would be understanding both a main and subordinate clause in a case when the subordinate clause gives more information on an item in the main clause ('**Neural network architectures** are needed that are capable of processing **large point clouds**'). Ob-

viously, the concept of memory can be helpful to gain a deeper understanding of this sentence and of data in general. Following this notion, recurrent neural networks and long-term-short-term memory architectures have been developed. Both have certain limitations, e.g. high computational expense, limited memory capacities and slow training due to difficult parallelization of the training. To improve the previous concepts, Vaswani et al. [109] initially developed and described the concept of transformer architectures in their paper 'Attention is all you need'. As the title suggests, the authors propose the concept of self-attention, i.e. to allow the network to connect information at different positions of a sequence relating each position to other relevant information. This concept enables neural networks to understand large sequences, e.g. a long sentence and even a text, and to perform tasks such as translation to the input sequence. The transformer architecture paved the ground for Large Language Models (LLMs) such as GPT-4 [110].

The concept of transformers has been transferred to the problem of point cloud scene understanding by Zhao et al. [44]. Wu et al. [43] propose PointTransformerV2 with advances in three fields. (i) the *grouped vector attention* allows the network to correlate attention groups more efficiently. (ii) An improved position encoding scheme allows the network to take advantage of the geometric context in point coordinates. (iii) Finally, a 'partition-based pooling strategy' allows to aggregate the features more efficiently, resulting in a better learning and inference.

The development of the semantic segmentation model has been a collaborative effort of the department Augmented Vision (AV) of the German Research Centre for Artificial Intelligence (DFKI) and the author, where the main contribution of this work is to capture, annotate and prepare the dataset for training. However, some basic information on the training procedure will be given here. More details can be found in the joint publication [22].

3.4.1 Training and prediction workflow

The training process employs a dual-phase strategy. Initially, the model undergoes pre-training on the S3DIS dataset [32], which facilitates the acquisition of general architectural features such as the vertical alignment of walls. Subsequent fine-tuning targets the specialized CV4AEC dataset, during which the model is refined while retaining most of its initial weight configuration — excluding the detection head. This approach ensures that the model maintains its core feature extraction capabilities while adapting to the unique attributes of the CV4AEC dataset. Notably, the CV4AEC dataset includes RGB colour values, although some scenes are exclusively in greyscale. As a form of data augmentation, some colour values were removed from the data during its pre-training on the S3DIS dataset. Additionally, point normals were integrated as supplementary features to enrich the training process. Several experiments have been carried out to evaluate different approaches:

- Before manual annotations for the CV4AEC dataset were available, the ground truth as presented with the challenge `https://cv4aec.github.io/` was used. From the ground truth, bounding boxes could be derived and all points inside the bounding box could be assigned the respective class label. Using these labels proved that the misalignment of the ground truth and the point cloud confuses the network during the learning process, resulting in poor training performance. This supported the motivation to manually annotate the data as previously described.

- Normals were computed and incorporated into the training dataset. These normals aid the neural

network in identifying different objects by offering insights into the data's orientation and curvature. Consequently, a cluster of vertically aligned points with similar orientations on a plane can be more readily identified as a wall, compared to using only points and colours.

- In an intermediate stage, only ten of the 25 manually annotated scenes of the CV4AEC data set were used. An improved training performance could be observed.

- The final model was fine-tuned on the 17 CV4AEC training scenes. This model is used for the final results presented here.

The dataset was segmented into varying-sized chunks based on density, typically ranging from 3 to 10 meters, with an overlap of approximately 10 to 20%. This segmentation ensured compatibility with the memory constraints of the available GPUs. In a manner akin to training, the data was partitioned into chunks for the inference phase. Given the overlapping nature of these chunks, it became necessary to amalgamate the labels of overlapping points. In instances of multiple labels for a single point, the label with the highest prediction confidence was selected. The prediction confidence can be determined by choosing the label most frequently inferred by the segmentation model.

3.4.2 Results

A segmentation model's performance needs to be evaluated on unseen data. One would expect a good performance on unseen data with similar features as in the training data. Evaluating on data with features different from the training data helps to determine how well the model generalizes, i.e. how well the model can infer on unseen data. Examples of unseen features include different shape of door and window frames, varying size, topology and arrangement of rooms, walls, and doors, changing room heights and facade objects, etc.

In Figure 3.8, a prediction on a point cloud of a church building is visualized. It can be observed that large areas of the walls have been misclassified as clutter (black), and that the network confused floor (dark red) and ceiling (purple). On the other hand, an inference on the A building of the 14NH dataset demonstrates a very good prediction result, with clearly separated wall, floor and ceiling sections and doors (refer to Figure 3.9). Although this example shares common features with the training data, the A building was empty, so no indoor objects were present in the data. This proves that the network generalizes well against the presence of indoor objects.

Overall, 0.69 mIoU could be obtained when evaluating on the S3DIS datasets, a promising result. However, it must be concluded that the inferred labels will remain incomplete. As can be seen in Figure 3.9, some parts of the door lining were falsely classified as wall. This will affect the subsequent steps of geometric feature extraction, as methods robust to incomplete data need to be developed to facilitate a high BIM model reconstruction accuracy.

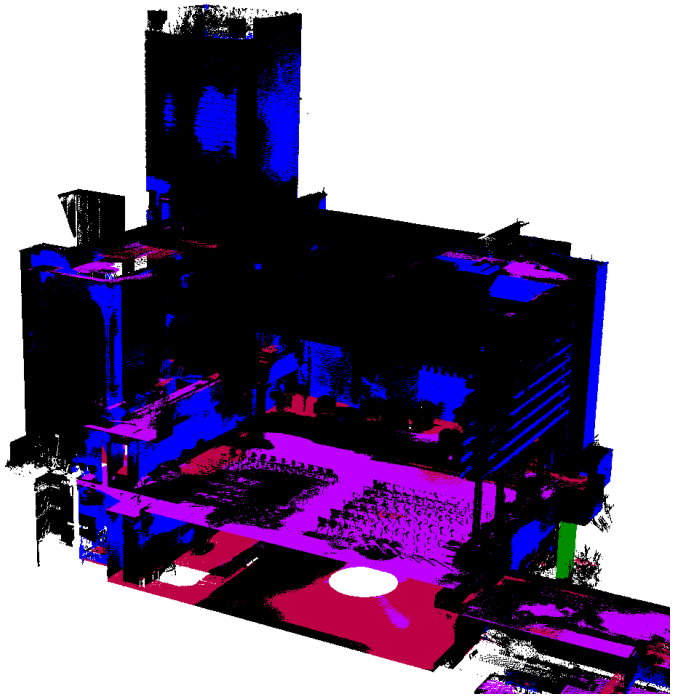


Figure 3.8: Generalization limits of the Point transformer v2 segmentation model. Inference on a point cloud of a church building with notably different features than in the training data. A considerable number of points are falsely classified as clutter (rendered black).

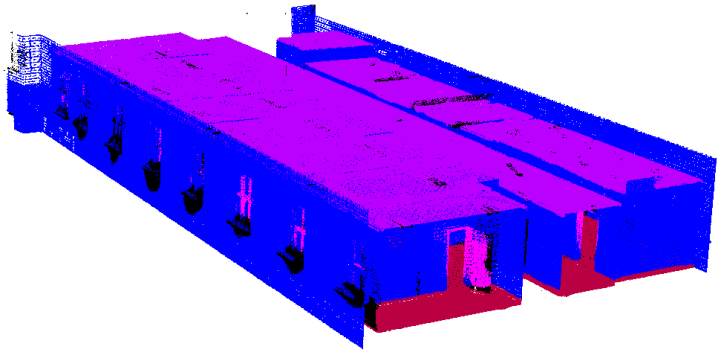


Figure 3.9: Inference on the A building of the 14NH dataset. As compared to the training data, the rooms were almost empty, which proves generalization of the model regarding the presence of indoor objects.

4 Geometric feature extraction

4.1 Introduction

The input for geometric feature extraction is a segmented point cloud. With the semantic labels, all points of a specific component class can be extracted from the point cloud and processed individually. Thus, tailored feature extraction strategies can be developed and applied. Yet, these strategies share the following common sequence of steps:

- **Clutter filtering**: Noise and clutter outside the building perimeter are the result of objects scanned through openings of the building, reflections from translucent objects, vegetation outside the building, objects in the field of view of the scanner and other artefacts. In actual point clouds of buildings such clutter is typically present and should be removed as one of the first steps to avoid reconstruction artefacts. Clutter filtering can be applied to the entire set of input points or to a subset of points of a particular class.

- **Instance segmentation**: During semantic segmentation, class labels are assigned to all points. Through filtering, all points of a component class can be extracted from the point cloud. However, individual objects cannot be distinguished. Thus, a method for instance segmentation needs to be applied. To achieve this, clustering strategies exploiting the object orientation, point density and relation to parent objects are developed and used. From this step, points of one object instance are retrieved.

- **Parameter extraction**: Per object instance, the geometric parameters are extracted including shape, dimension, translation and rotation. This can involve one or multiple steps of primitive fitting, filtering and bounding box fitting. For cuboid shaped objects, a bounding box defined by eight corner points is used to represent the result. For other shapes, similar representations can be defined.

- **Topology refinement**: The resulting parameters extracted apply for one object solely, the relation to other objects has not yet been considered. Thus, topology refinement algorithms are applied to correctly reconstruct the objects' relations. In this process, objects are modified to form closed corners or combined when they share a common centre line. Child objects are aligned to the parent object and modified to share same boundaries if applicable

- **Additional filtering and refinement**: Based on general knowledge about structural building design, specific objects above or beyond dimension limits can be either modified or discarded. This includes the modification of the object's dimensions or discarding elements with dimensions too small or large.

Considering that any of the reconstruction tasks is a 3D problem, at least 9 Degrees of Freedom (DoF) (3D translation, 3D rotation, 3D dimension) need to be solved. If the type of primitive applying to an object is unknown, this adds one more degree of freedom to the problem. Beyond the mere dimensions, the general shape of the object first needs to be defined. However, objects might not only consist of one primitive shape but might be a combination of primitive shapes or topologically even more complex.

Developing geometric feature extraction algorithms that can be applied in a very general manner, i.e. on arbitrary shaped objects without any constraints of orientation, is a complex task. Such algorithms are likely to be computationally expensive, too. From a practical point of view, it can even be argued that an efficient reconstruction algorithm is more useful than a general yet inefficient one, provided that uncertain reconstructions raise an error that can be handled in a subsequent processing step or even manually. Thus, it seems desirable to simplify the reconstruction problem whenever this is sensible from a practical point of view.

Simplifying the reconstruction problem involves two concepts: applying the Manhattan-world-assumption [21] and further *Geometry hypotheses*.

Being formulated by Coughlan et al. [21], the Manhattan-world-assumption states that in an urban context, most objects are aligned along a two-dimensional *Manhattan* grid, i.e. along axes perpendicular to each other. Being a reminiscence to Coughlan's work on determining the user orientation with a Computer Vision algorithm, the term became a synonym for any objects aligned with a 2D perpendicular grid. This notion also applies to most buildings themselves, where most objects are aligned along a set of perpendicular axes. Whenever the axes are parallel to the axes of a Cartesian coordinate system, the data can be considered *axis-aligned*. In this work, not only the Manhattan-world-assumption is used to simplify the reconstruction procedures, as even the input data needs to be axis-aligned. Although this is not the case in real-world scenes, the input point cloud can either be axis-aligned programmatically using Principal Component Analysis (PCA) or manually with little effort.

The idea of *Geometry hypotheses* is to simplify reconstruction problems even more by defining constraints for object classes, which apply for the majority of the objects observed. The constraints can include shape or primitive, planarity, horizontal or vertical alignment, orientation of bounding surfaces, etc.

In the following sections, the methods will be described along the reconstruction sequence of components, starting with walls, continuing with doors being child objects of walls and ending with columns representing typical members of structure skeletons. Although some concepts apply to several or all components to be reconstructed, it seems more natural to explain the process sequentially rather than elucidating on a theoretical level with only a minor relation to the actual reconstruction stack. In a preliminary section, however, the geometry hypotheses that apply for the aforementioned types of components will be explained. Some more sections will be dedicated to storey reconstruction, clutter filtering and outside wall reconstruction. Finally, measures for error handling as well as the implementation of the methods will be described. In Annex 3, links to all repositories and online sources supplementing this chapter can be found.

4.2 Geometry hypotheses

Geometry hypotheses can be formulated for each object category to be reconstructed and are used to inject prior information to the instance segmentation and reconstruction process. Some assumptions are trivial but it is important to describe these thoroughly, to let the reader understand the functionality of the reconstruction algorithms developed.

Horizontal surfaces referred to as **levels** or storeys exist that organize a building vertically. These exist in any building, regardless of even large sloped areas. Levels are the key structure to organize a building, thus this hypothesis will be used in the reconstruction of all objects.

Wall objects are reconstructed as cuboids that consist of vertical or horizontal bounding surfaces. The main vertical (mostly the visible) surfaces are parallel to each other. The lower horizontal boundary surface is equal to the floor surface. The upper horizontal boundary surface ends with the ceiling or slab, or with a dedicated upper horizontal surface in case of parapet walls.

Opening objects are cuboid and generally the child of a parent object such as a wall or slab and may thus not exist solely. Any surface of openings is at least parallel to the surfaces of the opening object, some surfaces may be equal. This concept applies to **doors** that share the same lower horizontal surface as the floor. Door objects are aligned with the parent object. As doors are used to control the access to areas in a building, it must be possible to walk through a door, and building codes can be consulted to determine the dimension constraints. The same concept applies to **windows**, except that the lower horizontal surface is typically higher than the floor surface.

In reconstruction algorithms, two distinct shapes for **columns** are acknowledged: cylindrical (round) and cuboid (rectangular). Any column with a curved surface, such as an ellipse-shaped column, is categorized as round. A bounding box will be reconstructed of column instances with other arbitrary shapes, e.g. of polygonal cross-section. Regardless of their shape, columns maintain a consistent profile throughout their length. Functioning as structural components, columns provide vertical connections between various elements like slabs, beams, and girders. In instances where structural elements are concealed, such as a slab hidden by a suspended ceiling, columns are modelled to extend from the floor to the ceiling surface, reflecting the absence of these hidden layers in the data.

Considering the varied configurations and geometric characteristics of actual buildings, one should anticipate that using these assumptions might result in inaccurate reconstructions. This scenario occurs, for example, when reconstructing a curved wall under the assumption that a cuboid wall was observed. Clearly, such inaccuracies are unacceptable. Therefore, methods will be implemented to detect potential erroneous reconstructions. Preliminary steps are to apply clutter removal and downsampling procedures to allow for an efficient processing.

4.3 Density based clutter removal

In a first attempt to reduce the number of points in the point cloud, a density-based clutter removal algorithm is developed and applied to clean the set of points from those that are not of interest to the reconstruction. Clutter points in this case include scanning artefacts outside the building caused by

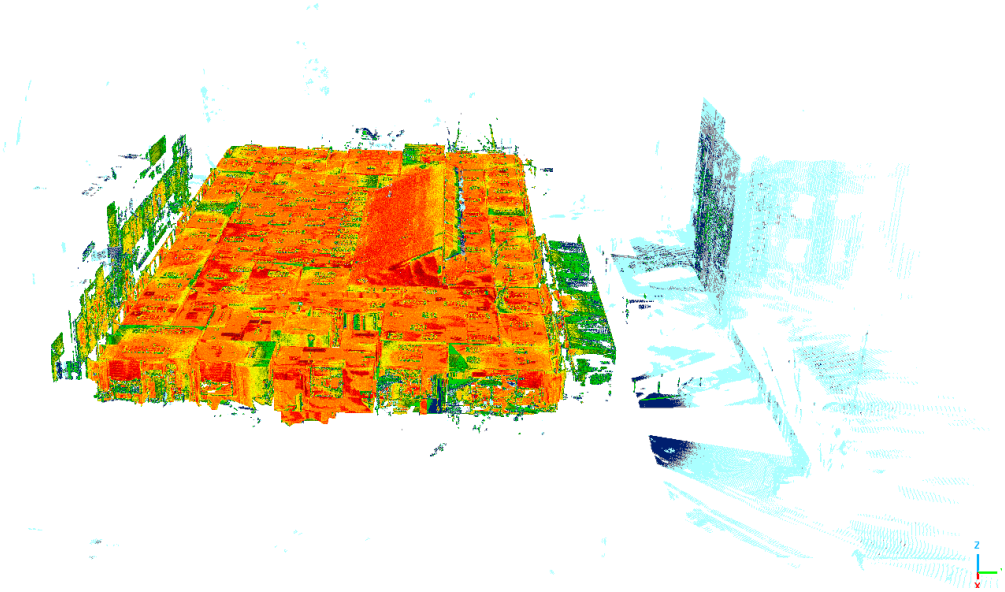


Figure 4.1: Density map of a point cloud of an office building. The point density was calculated by determining the number of neighbours in a given radius of 2.5 cm. The density is visualized using a colour scale spanning from blue (low density) to red (maximum density). The density of clutter objects surrounding the building is considerably lower than the building itself.

reflecting or translucent surface deflections, vegetation, moving vehicles and pedestrians, neighbouring buildings, etc.

The core observation that motivated the development of this density-based clutter removal algorithm is that clutter points, especially those outside of the building, have a lower point density than the object of interest itself (refer to Figure 4.1). As the point density relates to the distance of the object captured, objects of interest related to the building at a closer distance are retrieved with a higher point density and clutter points scanned from a farther distance with a lower point density. This fact can be used to remove clutter based on its lower density compared to building elements of interest.

The principle is visualized in Figure 4.2. Voxels with low occupancy around the perimeter of the building are discarded, whereas dense voxels are filtered.

The density-based clutter removal algorithm used in this work includes the following procedure (refer to Algorithm 4). First, a voxel grid is built on the point cloud. To identify sparsely or not populated voxels, the number of points inside a particular voxel is calculated. All voxels populated with a number of points below the predefined limit will be considered as clutter. The inliers, i.e. points of interest, are then identified by constructing a mask with indices of inlaying points that is applied on the input wall point cloud. The result of this filtering step is presented in Figure 4.3 left. The orange bounding box encloses the points of the building to be reconstructed. Note that inside the bounding box points were considered as clutter (grey colour) by the algorithm.

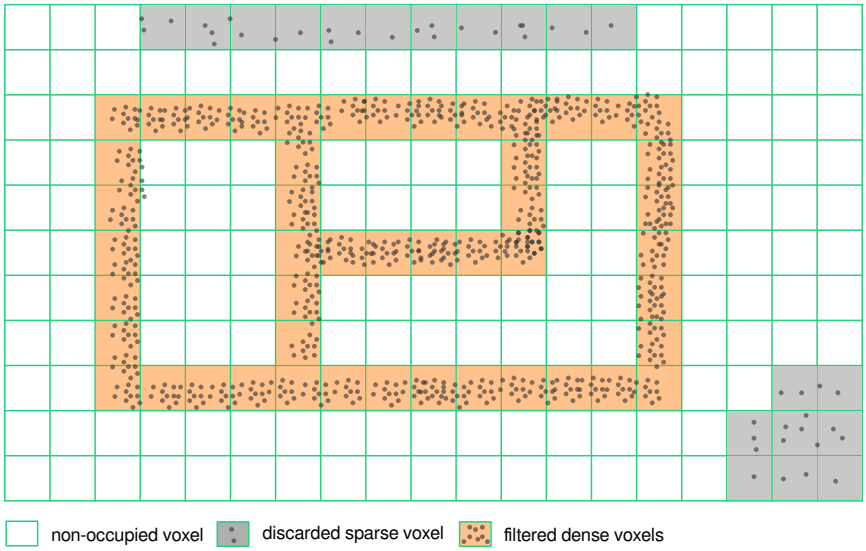


Figure 4.2: Voxel density filtering to remove clutter and noise based on the voxel occupancy.

input : input point cloud
input : voxel size
input : minimum points per voxel `// density threshold`
result : dense point cloud `// only dense voxels are preserved`
output: density filtered point cloud `// noise outside the perimeter of the building removed`

```
   // disjoint the point cloud into discrete voxels
1  voxel grid = voxelize(input point cloud, voxel size)
2  for voxel in voxel grid do
3  |  inliers = find inliers(voxel, input point cloud)
4  |  if number of (inliers) ≥ Minimum points per voxel then
5  |  |  append(dense point cloud, inliers)
6  |  end
7  end
8  bounding box = minimum bounding box(dense point cloud)
   // get inliers from input point cloud to preserve all the details
9  density filtered point cloud = find inliers(bounding box, Input point cloud)
10 return density filtered point cloud
```

Algorithm 4: Density-based clutter removal

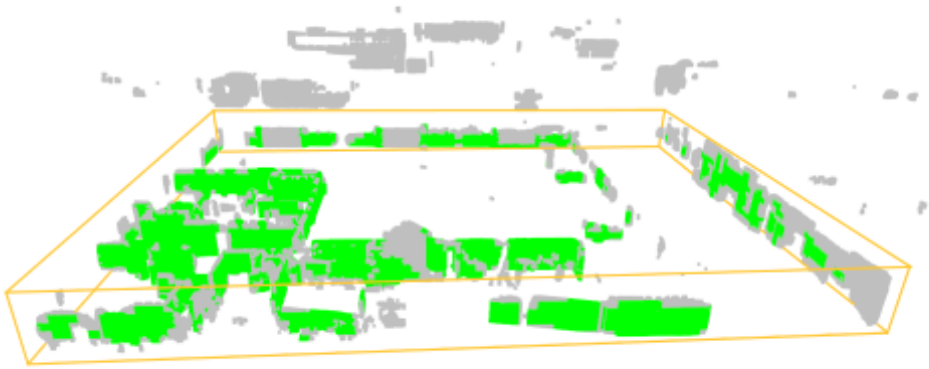


Figure 4.3: Density-based filtering of input point clouds - filtered points green, other points grey, inlier bounding box orange.

To avoid discarding points with a lower density from within the building itself, the minimum oriented bounding box of all inlier points is calculated using Open3D: a modern library for 3D data processing (Open3D)'s *get minimal oriented bounding box* method, which provides the bounding box with the smallest volume [111]. All points inside this bounding box (see Figure 4.3) are considered for further processing thus removing low density clutter outside the perimeter of the building but preserving low dense details.

In the example given in Figure 4.3 a set of wall points was processed. A voxel size 0.65 m was chosen, and the density limit, i.e. minimum number of points per voxel, was set to 2500. Depending on the density of the input point cloud, and downsampling applied before, these parameters need to be adjusted. A disadvantage of the proposed method is that it is computationally expensive and requires huge memory if a small voxel size is chosen. For point clouds of buildings, a large voxel size of more than 0.5 m is acceptable and provides good filtering results at a reasonable computation time.

As this algorithm exploits the point density, it is crucial to apply it to the input point cloud before any other downsampling procedures, which could potentially harmonise the point density.

4.4 Voxel downsampling

Point clouds from building scans can encompass tens of millions of points, dictated by factors such as the scanning resolution, overlap of aligned scans, and preliminary processing steps. However, processing these point clouds at full resolution can require significant computational resources and extend the processing time. Moreover, this might not necessarily enhance the accuracy of the outcome. To avoid unnecessarily lengthy processing and resource-intensive procedures, a common initial step is to reduce

the quantity of points, a process known as downsampling. Still, it is essential that any down sampling method retains the key features that contribute to a detailed BIM model of the point cloud, such as edges and planar surfaces. As such, efficient down sampling strategies are necessary to preserve these crucial geometric details.

input : input point cloud
input : voxel size
output: downsampled point cloud

```
   // disjoint the point cloud into discrete voxels
1  voxel grid = voxelize(input point cloud, voxel size)
2  Function voxel downsample(points, voxel grid):
3      for voxel in voxel grid do
           // find all points inside the voxel
4          inliers = find inliers(voxel, input point cloud)
5          if number of(inliers) ≥ 0 then
               // average coordinates and colours
               // a new point is returned for every voxel with one or more points
6              new XYZ = mean (coordinates)
7              new RGB = mean (colours)
8              append(downsampled array, new XYZ, new RGB)
9          end
10     end
11     return downsampled array
12 downsampled point cloud = voxel downsample(input point cloud, voxel grid)
13 return downsampled point cloud
```

Algorithm 5: Voxel downsampling.

One approach that fulfils the described requirements is *voxel down sampling*, as presented in Algorithm 5. This method builds a voxel grid inside the boundaries of the point cloud. A voxel grid is a discrete 3D structure of cuboids that can be used to work with point clouds in a structured manner and can be applied to tasks such as spatial search and filtering. The size of each voxel in the voxel grid, i.e. the edge length of the voxel cubes, controls the rate of points that will be reduced. The larger the voxel size, the less points will be preserved in the output. This is caused by the fact that in every voxel, all points will be combined to one point by calculating the arithmetic mean of all X, Y and Z coordinates. Likewise, colours and normals are averaged. Along this procedure, any number of points in a voxel will be deflated to one average point of the former voxel points. The algorithm as described here is implemented in Open3D [111]. The main advantage is that regardless of the number of voxel inliers a point representing the former multiple points is returned.

Compared to random down sampling, voxel down sampling preserves the detail in the point cloud (see Figure 4.4). In random down sampling, random points will be deleted from the point cloud. As this method is agnostic of the density of the point cloud, areas along the edges, sparse areas and finer details might be lost. In sparse areas of the point cloud, voxel down sampling still preserves points, as sparsely populated voxels (even voxels with only one point) will still return one down sampled point, so sparse areas will not be lost, and details preserved. Voxel downsampling thus equalizes the point density by discarding more points in dense areas and preserving points in sparse areas.

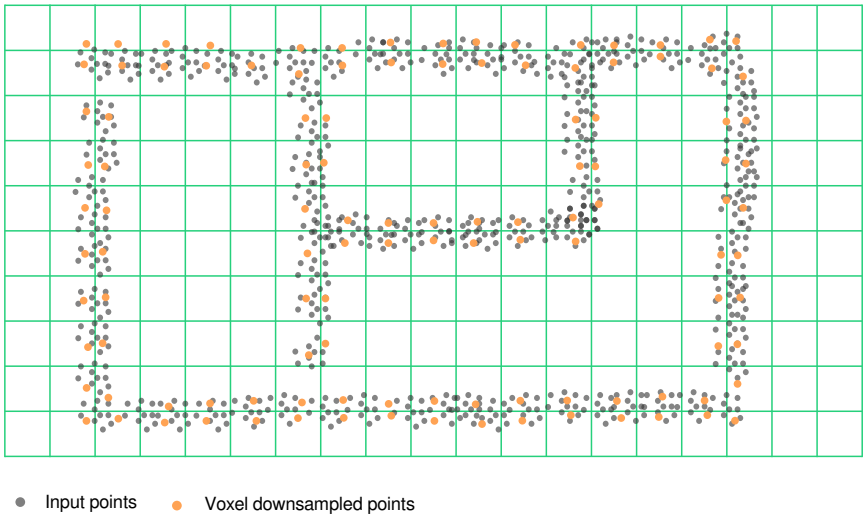


Figure 4.4: Principle of voxel downsampling, 2D projection. All points (grey) in an occupied voxel will be reduced to the average of all points (orange) in the voxel. It can be observed, that details such as edges are preserved well, while significantly reducing the density of the point set. Note that the figure is not scaled, in a real application a smaller voxel size would be chosen.

For practical application, a voxel size between 0.02 m and 0.05 m can be chosen depending on the component category. It was observed that within this range the density of the point clouds can be reduced up to a factor of 10, while the remaining resolution of at least 5 cm still preserves the necessary details. Voxel downsampling is applied to each subset of points of a particular class individually. Dedicated voxel sizes can thus be chosen for every element class depending on the number of points, processing steps and the features of the elements in the point cloud. Generally, larger objects are down-sampled more thoroughly than detailed objects confining less space and thus fewer points.

With the downsampled point cloud obtained, the first reconstruction procedure to find the building's levels can be initiated.

4.5 Levels and storeys

Many elements are bounded by horizontal structural components, such as slabs, or by floor and ceiling when the structural element is concealed. In BIM models, objects are typically associated with levels to organize the model. Identifying these levels is thus a fundamental initial step in facilitating the reconstruction of other elements.

Level reconstruction algorithms can be designed and applied either to the entire point set without prior

semantic segmentation, or specifically to points identified as slab, floor, and ceiling. When semantic segmentation is applied beforehand, all points categorized as floor, ceiling, and slab are grouped horizontally. The average of all z-coordinate values within each horizontal group is then calculated to determine the heights of the floor, ceiling, or slab. The vertical distances allow for the identification of the floor or the top surface of the slab.

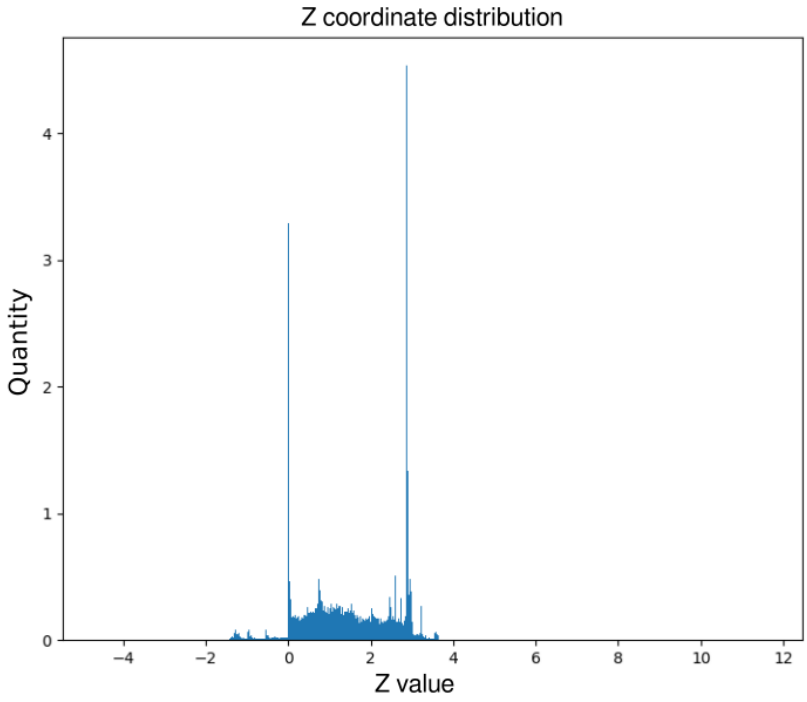


Figure 4.5: Distribution of z coordinates of a one-storey building. As expected, two peaks appear at the level heights.

In case the complete input point cloud is used, peaks in the z-coordinate value distribution are analysed. Although being computationally more expensive, this method offers the benefit of not relying on the accuracy of semantic segmentation that may be incomplete resulting in errors in the level heights computed. Additionally, inaccuracies in determining level heights can significantly impact every object reliant on the horizontal constraint hypothesis. Thus, it is recommended to use the entire input point cloud. In previous research, similar approaches to horizontal segmentation based on a point density histogram have been introduced in [87] and [81]. The algorithm proposed here is based on these approaches. However, [87] use a point density histogram to split the point cloud in between floor and ceiling. Likewise, [81] use the point density along the z-axis to split the point cloud into storeys to then further divide it into subspaces such as rooms. Although the algorithm proposed here uses a similar approach, it is designed to identify the level heights as an input for further object reconstruction.

The basic principle of finding levels is to exploit the point distribution of points in vertical direction (Z-axis). Intuitively, horizontal floor, ceiling or slab surfaces should cause a remarkable concentration of points in vertical direction. This phenomenon can be observed in Figure 4.5, which shows the point distribution along the Z-axis of a one-storey office building. The centre bins of the histogram (from

Input : input point cloud
Input : number of bins `// bins of the histogram`
Result: histogram `// can be plotted to visually check the result`
Output: levels

```
1 Z coordinates = extract Z coordinates(Input point cloud)
   // Build the histogram
2 histogram = build histogram(Z coordinates, number of bins)
   // peaks are local maxima in the histogram
3 peaks = find peaks (Histogram)
4 for peak in peaks do
5 |   min, max = get peak boundaries (peak)
6 |   level height = median (min, max)
7 |   append(levels, level height)
8 end
9 return levels
```

Algorithm 6: Level finding algorithm.

0 to 3) represent most of the objects (structural elements, furniture, other objects, clutter). The two major peaks represent the floor and ceiling, as for horizontal surfaces the number of points with equal z coordinate value will be significantly higher than the relatively homogeneous z distribution in between.

The procedure chosen can be described as follows (refer to Algorithm 6). Given a point cloud, a histogram is defined by partitioning the data set into k bins. In this case, only the Z component of the point coordinates is considered. Identifying peaks can be formulated as finding local maxima in the histogram. A bin is considered a peak if its count is greater than the counts of its neighbours.

Floor and ceiling or upper and lower surface of slabs can hereby be distinguished by the distance of the respective levels and prior defined thresholds such as the minimum room height or the maximum vertical distance of floor and ceiling. By applying these constraints, false positive level detection can be avoided.

Having reconstructed the levels, the object reconstruction can be initiated starting with the walls.

4.6 Walls reconstruction

Beyond the vertical organization of a building by its levels, walls are the main objects that enclose and separate the spaces thus forming the topology of the building. Hence, it is sensible to reconstruct the walls after the levels and storeys to provide these elements for the subsequent reconstruction of child objects of walls, e.g. doors, windows, openings, etc.

The core principle of wall reconstruction is to perform a per-direction reconstruction of the walls, followed by instance segmentation including clustering, plane fitting and parallel plane grouping. A minimum bounding box is then fitted to the wall instances. Special attention is taken to perform outside

walls reconstruction in a dedicated procedure to overcome the issue of incomplete outside wall data frequently encountered as only the inside surface of the outside walls is scanned and the outside surface is sparse. Finally, a topology-aware refinement algorithm is applied to ensure correct intersection of wall instances at corners and combine such near and parallel walls that could be merged but have initially been reconstructed as two or more instances.

These operations are performed on the subset of points assigned with the label wall. Hence, the first procedure is to filter the subset of a given label from the input points. Instead of iterative checking each point's label, a mask of boolean values is created from the labels array, where True indicates all positions with a given label. Applying this mask to the points array is an efficient way to retrieve all points of the given label.

As described in the objective section, only vertical walls with coplanar surfaces and a straight centreline are considered in this reconstruction procedure.

4.6.1 Axis sorting of walls

Reconstructing walls direction-wise yields the main benefit, that intersections at wall corners are eliminated from the per-axis data, thus resulting in a more reliable instance segmentation. Parallel walls are typically separated by at least the distance accessible for persons, typically 1 m or above. This fact can be used to apply density-based algorithms more efficiently and reliably.

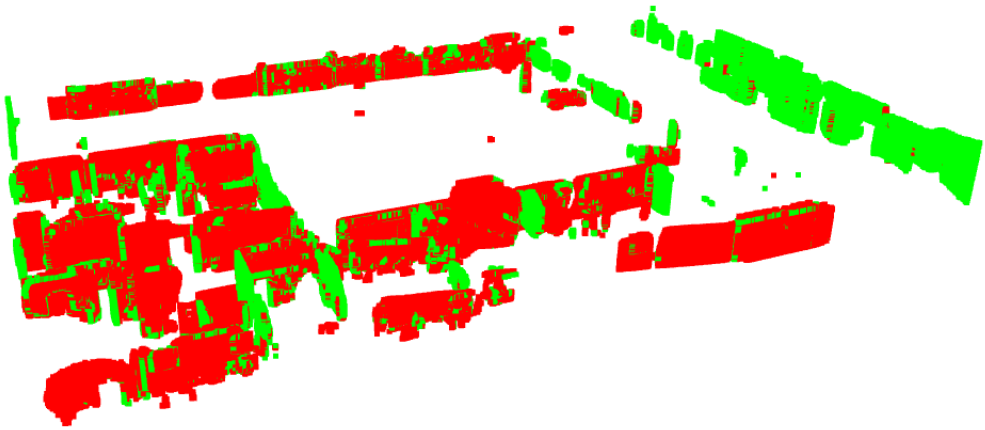


Figure 4.6: Axis sorted wall points. Axis sorted points along the x-direction in red, remaining non-selected points are assigned the y-direction and rendered in green.

Based on the Manhattan-world assumption [21] and the initial axis-alignment of the data, the point normals can be used to sort the walls according to their respective direction, namely walls oriented parallel to the X or Y-axis. An example of wall points sorted in axis direction is presented in Figure 4.6.

Input : wall points
Result: normals
Output: axis sorted points

```
// A k-dimensional tree structure is used for an efficient
 nearest-neighbour search
1
2 Kd tree = build KdTree(Wall points)
  // calculate the point normals in a local neighbourhood
3 Function calculate normals(points, Kd tree, max number of neighbours):
4     for Point ∈ Wall points do
5         neighbours = find nearest neighbour(Kd tree, Point, max number of neighbours)
6         covariance matrix = calculate covariance matrix(neighbours)
7         Eigenvalues λ1, λ2, λ3 = eigen decomposition(covariance Matrix)
           // The last of the Eigenvalues is the point normal
8         append(normals, λ3)                          // store normals in an array
9     end
10 Function axis sorting(points, normals, direction):
       // X or Y direction, use respective normal component
11    mask = where(N_X ≥ 1.0 − threshold in normals)      // True if condition is met
12    axis sorted points = points[mask]
13    return axis sorted points
14 axis sorted points = axis sorting(wall points, normals, direction)      // do for X and Y
   direction
15 return axis sorted points
```

Algorithm 7: Axis sorting.

The axis sorting algorithm (refer to Algorithm 7) includes two steps: (i) calculating the point normals, i.e. the vector normal to a point considering a number of adjacent points and (ii) splitting the set of points into two subsets of such points normal to the X or Y direction, respectively.

First, the normals are calculated within a local neighbourhood of points. The points within the local neighbourhood are identified by a k-nearest-neighbour search in a k-dimensional tree (k-d tree) structure using the Open3D implementation [111]. The normal itself is computed using the Eigenvalues of the covariance matrix, with the third Eigenvalue equivalent to the point normal. This normal value is then assigned to the respective point. The algorithm outputs two normal vectors of opposite direction, which both can be correct. Without any context, the direction is randomly chosen. This behaviour is acceptable, as only the orientation is needed for axis sorting, not the actual direction of the point normal.

To sort all walls that are close or in x-axis direction, all points with normals' Y component equal to one within a predefined limit of $\pm 0.1\%$ are selected. As the normal vectors have previously been normalized, the limit does share the unit / scale as the overall point cloud. The procedure is applied for Y direction, respectively. There is no strict verticality constraint applied but implicitly, points of non-vertical direction are not considered as such arbitrary orientation of the point normal causes the X or Y component of the normal vector to drop below the limit.

Figure 4.7: DBSCAN clustering. A set of wall points with clusters visualized. Points not assigned to a cluster in black colour.

4.6.2 Instance segmentation

From the axis sorted points, wall instances need to be segmented. Based on the analysis of basic geometry processing algorithms (refer to section 2.3), plane fitting and density-based clustering are the alternatives to be considered. Although plane fitting algorithms robust to noise exist, we experienced limitations of applying algorithms such as RANSAC directly to a larger set of points. One of the main limitation is that the input parameters need to be selected carefully, which is difficult on an unseen dataset in an automated scenario. Some insights into the limitations of RANSAC as proposed by [64] will be given in the next section, revealing that such procedures applied to a larger set of points do not yield robust and accurate results. This leads to the preference of density-based clustering algorithms for three reasons. (i) In a scenario with spatially separated clusters as is the case with axis sorted wall points, density based clustering algorithms such as DBSCAN introduced by Ester et al. [68] yield robust and accurate results with only a few, easy-to-set input parameters. Indeed, the parameters could be almost set globally, with tiny tweaks to reduce the processing time. (ii) The algorithm only returns an instance label per point, no further information such as primitive parameters. Although this might sound like a disadvantage, it is in fact irrelevant as such parameters would not be used at this stage. (iii) DBSCAN does not require the number of clusters as an input, which would be impractical to select on unseen data.

An example of a batch of axis-sorted wall points that the DBSCAN algorithm was applied to is shown in Figure 4.7. All clusters are coloured randomly. Note that colours might repeat as the number of colours in the colour map is limited. Points that were not assigned to a cluster are black.

For the details of the DBSCAN algorithm [68], please refer to Algorithm 2 in section 2.3. The algorithm exploits the fact that wall instances are formed by points within a certain minimum distance to the

nearest neighbour points. Hence, for every point in a wall instance one or more points can be found that are within a distance threshold. The distance threshold can be considered as the maximum wall thickness that is set manually. Longer distances to neighbouring points would be encountered in other wall instances, e.g. another wall on the other side of a hallway. As the walls were axis-aligned before processing, neighbouring points at corners are eliminated beforehand.

In this work the Open3D implementation of DBSCAN [111] is used as it is efficient compared to a pure python implementation. The Open3D implementation of the algorithm takes two parameters: epsilon and the minimum number of points per cluster. The first can be interpreted as the maximum wall thickness expected in the data. The minimum number of points per cluster can be selected based on the density of the dataset observed, typically a few hundred points. All tests have proven that the algorithm works robustly on building and construction data both in terms of quality / accuracy and computational robustness. It is also robust to noise in the data. As a side effect, DBSCAN discards areas with a point density too low, or small clusters. These can be considered artefacts (refer to Figure 4.7) not relevant to the reconstruction.

The algorithm is applied to a set of points that have been labelled as 'wall' during the process of semantic segmentation. It is important to note that this set might include FPs. These FPs can be points associated with objects mounted on walls (like pictures), items positioned in front of the wall, and various artefacts. Fitting a bounding box to these instances of walls would likely result in significant inaccuracies. Therefore, additional steps are necessary to discern the inherent geometry of the observed wall instance. The primary objective is to obtain an accurate depiction of the wall surfaces.

4.6.3 Hypothesis based plane fitting and parallel plane grouping

The scope of wall reconstruction has been contained to walls with planar surfaces. As this limitation applies to the vast majority of wall objects observed in building point cloud data, it is sensible to try to reconstruct the planar surfaces of the wall instances observed. To this end, plane fitting algorithms can be utilized. The most common ones are elaborately discussed in chapter 2.3, with their specific advantages and disadvantages when applied on building data. Based on the literature study, the RANSAC algorithm [64] could be identified as the most suitable for this work.

Nevertheless, when applied to building data, some limitations remain. An example of such limitations of the RANSAC algorithm [64] is presented in Figure 4.8. The figure reveals that both the *minimum number of support points per primitive* and *maximum distance to primitive* need to be selected carefully. The *maximum distance to primitive* controls the minimum acceptable distance of a point to the plane, *minimum number of support points per primitive* defines the minimum number of points for a plane to be accepted. Depending on the degree of noise and the characteristics of the data, setting the correct input parameters requires human interaction and experience. This can be challenging due to several factors. (i) It is likely that there is no global parameter setting that yields good results for all wall instances. (ii) Setting the parameters correctly requires a good intuition and experience based on a previous visual inspection of the data. This is impractical in an automated pipeline. This underpins the motivation to develop a more robust derivative of RANSAC [64].

The problem described and shown in Figure 4.8 is mainly caused by the modus operandi of the RANSAC algorithm [64] (refer to Algorithm 3 in section 2.3). As the name indicates, RANSAC [64]

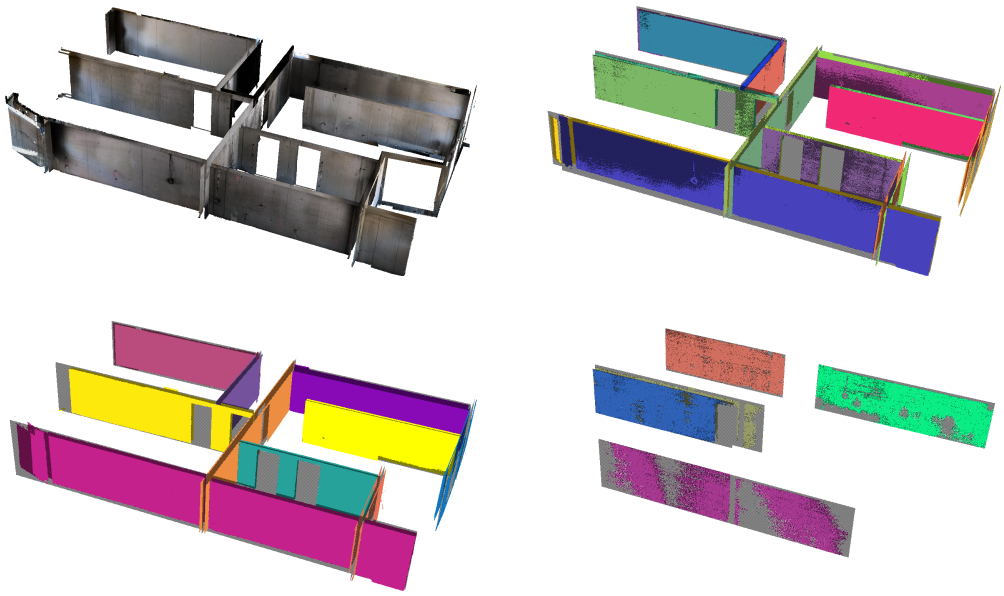


Figure 4.8: Good and bad results of RANSAC plane fitting on wall points. Top left: original input point cloud, manually segmented. Top right: good result, two planes (one per wall surface) were fitted with *minimum number of support points per primitive* = 50.000 and *maximum distance to primitive* = 3 cm. Bottom left: Only one plane was fitted to two surfaces with *minimum number of support points per primitive* = 50.000 and *maximum distance to primitive* = 12 cm. Wall thickness was approx. 20 cm. Bottom right: incomplete segmentation with *minimum number of support points per primitive* = 50.000 and *maximum distance to primitive* = 0.1 cm. CloudCompare [4], which implements the efficient RANSAC as proposed in [64], was used for the experiments.

will first sample a number of seed points randomly from the input points. Schnabel et al. implement a random sampling within a local neighbourhood, as from seed points of a local neighbourhood it is more likely to find a good fitting plane. If the plane constructed encompasses more points than *minimum number of support points per primitive* and the point's distance is below the *maximum distance to primitive*, the plane gets accepted. The number of iterations is limited, and the best fitting plane is returned whenever an accepted plane is found. This can lead to planes accepted by the algorithm that still merely represent the geometry of the data.

The random sampling of seed points for plane fitting can hereby be identified as the main obstacle as, based on random seed points, FPs are likely to be accepted when both input parameters do not restrict the algorithm to well-fitting planes. Besides, many iterations that come at a high computational cost may be needed to find a good fitting plane. If it was possible to identify seed points more likely to approximate the plane surface, both the number of iterations could be declined, and erroneous fitting could be avoided. To find better seed points, some hypotheses can be formulated that leverage a pseudo-random seed finding algorithm:

- Elements and thus elements' planes are either horizontal (floor, ceiling, slabs, ...) or vertical (walls). This hypothesis is supported by the Manhattan-world assumption [21], which is applicable.

- The point cloud cluster of the respective elements comprises one plane, one plane and a sparse plane, or two planes. The first case will be encountered when only a single side of the element was captured. The second case is typical for outside walls, where scans are complete and dense on the inside but comparably sparse on the outside due to occlusions or a lower resolution. The third case should be standard for 'complete' data, e.g. of inside walls.

- Considering the distribution of points perpendicular to the main wall surface, the best fitting plane is most likely to be found in areas with the highest concentration of points. This notion can be supported intuitively by the observation that most points of a wall surface indeed support the main planar surface, and a comparatively smaller portion is slightly offset due to objects attached to the wall, misclassification, false cluster assignment, etc.

The aforementioned shortcomings of RANSAC and the assumptions specific to the proposed pipeline motivate the development of the Hypothesis Based Sampling Consensus (HYSAC) algorithm (refer to Algorithm 8) with a special focus on an advanced seed sampling algorithm based on the distribution of points perpendicular to the main wall surface. First, the histogram of the point components perpendicular to the wall direction is constructed. By finding peaks in the histogram the subset of points subject to forming the centre of a plane can be identified. Identifying peaks can be formulated as finding local maxima in the histogram. A bin i is considered a peak if its count is greater than the counts of its neighbours, i.e. $H(i-1)$ and $H(i+1)$. Therefore, peaks can be defined as $P = \{i : H(i) > H(i-1), H(i) > H(i+1)\}$. From the peaks in the histogram, a random set of points is selected, which is used to fit a plane.

For every set of seed points a plane is fitted using Singular Value Decomposition (SVD), which acts similarly to the previously described PCA and delivers three main vectors representing the data. Technically, three seed points would be sufficient to define a plane equation. However, with only three seed points there is a chance to not fit an optimal plane, resulting in more iterations. Although SVD is designed to handle larger sets of input points, the computation time of SVD is $O(n^3)$ as matrix multiplications are involved. Thus, the computation time increases by a factor of 8 when the number of points double. Considering that a few hundred plane fitting operations can be needed to process an entire scene, it is sensible to minimise the number of seed points. To yield good results at a low number of iterations, a number of 5 to 10 seed points can be recommended, depending on the degree of noise in the data.

The third of the vectors returned by SVD is the plane normal vector. The plane equation in the general form can be formulated as $ax + by + cz + d = 0$, where a, b, c are the x, y, z of the normal vector. The parameter d can be calculated using the mean of all seed points, i.e. the seed centroid which is inserted as x, y, z to solve the equation and retrieve d.

With the plane equation, the distance of all points to the plane is calculated. All points with a distance below the defined distance limit are considered as plane inliers, are returned and will be removed from the initial set of points. The second condition to be met by the algorithm is a minimum inlier ratio. Whenever the error is below the threshold given and the inlier ratio is above the minimum defined, the plane is accepted. If the plane is discarded, the process of finding seeds and SVD plane fitting is repeated until a satisfying plane is found. The number of iterations is limited to avoid endless processing

Input : wall cluster points
Input : number of bins in the histogram
Input : number of seed points
Input : min points per plane
Output: plane equation
Output: plane inliers

Function `histogram seeds(`*wall cluster points, number of seed points*`)`**:**
    `// build the histogram perpendicular to the wall direction`
    histogram = `histogram(`*wall cluster points, number of bins*`)`
    peak = `find peaks(`*histogram*`)`
    `// randomly pick n points from the peaks`
    seed points = `random sample(`*peak, number of seeds*`)`
    return *seed points*
Function `hysac plane(`*points, number of seed points*`)`**:**
    `// loop until all planes are found`
    while *points* $\geq$ *min points per plane* **do**
        seed points = `histogram seeds(`*points, number of seed points*`)`
        plane equation, plane inliers = `fit plane svd(`*seed points*`)`
        if `number of(`*plane inliers*`)` $\geq$ *min points per plane* **then**
            `remove(`*remaining points, plane inliers*`)`
            return *plane equation, plane inliers*
        end
    end
plane equation, plane inliers = `hysac plane(`*wall cluster points, number of seed points*`)`
return *plane equation, plane inliers*

Algorithm 8: HYSAC plane fitting.

in cases where no plane can be fitted to the data.

In Figure 4.9 two examples of HYSAC plane fitting are presented. As described it can be observed that HYSAC is robustly fitting planes to the wall surfaces, even when they are sparse, incomplete or cluttered. Noise and clutter are reliably identified as outliers and not assigned to a plane. Even when the points are not continuous along the wall (see above right in Figure 4.9), based on the peaks, planes can be fitted reliably.

The plane fitting is performed on the initial set of seeds. In case the inlier ratio is below the minimum inlier ratio, and the mean error is above the maximum error, another set of seeds is calculated, and the process is iterated up to the maximum number of iterations defined as an input. If no plane is fitted after the maximum number of iterations, 100 by default, it can be assumed that the data cannot be represented by a plane.

When testing the algorithm and applying it to wall point clusters of building point clouds, multiple advantages compared to the conventional RANSAC could be observed:

- On dense data, only one iteration is needed to find a plane within a 5 mm point-to-primitive error

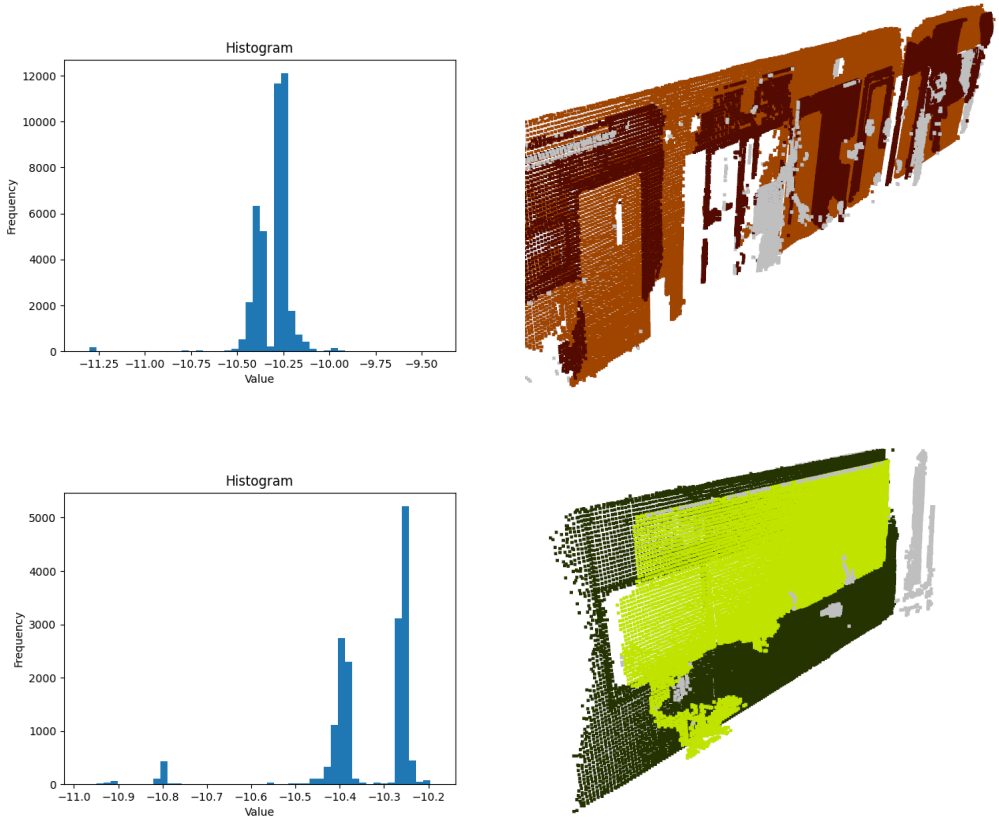


Figure 4.9: HYSAC plane fitting. Result of plane fitting (plane points coloured, outliers grey) with corresponding histogram of point distribution.

margin. Even on cluttered and noisy data, only a few iterations are needed to segment a plane within the given error margin.

- Other than RANSAC, which would fit a plane in between two wall surfaces (refer to Figure 4.8), HYSAC will reliably identify the two planes representing both wall surfaces. This is related to the fact that the seed finding algorithm inherently focuses on potential planes.

- In dense as well as in sparse data planes can be fitted reliably, as peaks can be identified as local maxima agnostic of the absolute number of points in the peaks. This reduces the importance of the minimum number of points per primitive as an input parameter.

It must be noted that the application of HYSAC is currently limited to axis-aligned data. However, there are perspectives to adapt HYSAC to arbitrary oriented objects, e.g. by applying a PCA to determine the main wall direction.

When encountering sparse, half scanned objects, HYSAC deliberately fits only one plane to the data. Without any additional algorithms, this is what can be derived from the input data. However, wall BIM objects with zero thickness are neither realistic nor workable. To handle these exceptions, a *one plane handler* algorithm is used. As some outliers, e.g. from window or door linings, might be present these can be used to infer the direction the wall element expands to. The wall thickness must be given as an input as it cannot be inferred reliably from the data. Effectively, all points from the wall cluster will be clipped in a predefined distance parallel to the plane equation. To ensure that the predefined thickness is represented in the data, one point at the distance equal to the predefined thickness is added to the handled points on the side the most points are present. The *one plane handler* takes the estimated wall thickness as an input, hence walls with only one plane fitted will have a fixed thickness. The points clipped using this procedure with the added thickness point will then be used for further processing.

The planes fitted using HYSAC will then be grouped together based on a parallelism constraint and a maximum distance of parallel planes, i.e. the maximum wall thickness expected. The result of these procedures are sets of points per wall instance. Any points outside the wall surface plane, occurring through objects attached to the wall, clutter close to and assigned to the wall cluster, have been cleaned from the data. Thus, the next processing step, namely bounding box fitting, yields more accurate results.

4.6.4 Bounding box fitting

The key process to retrieving the geometric parameters is bounding box fitting. Although a procedure not too complicated, this step only yields realistic geometric parameters when the data is treated as previously described including instance segmentation, plane fitting and grouping which implicitly cleans the data from noise. Directly applied to wall instances, the noise would result in too big dimensions of the bounding box retrieved.

Obviously, an axis-aligned bounding box[1] does not provide too much information about the set of points encapsulated. Neither does it provide an accurate dimension, nor translation or rotation in any direction. However, axis-aligned bounding boxes are easy to fit to the data, as the minima and maxima per coordinate component need to be found. On the other hand, an oriented bounding box provides accurate dimension, translation and rotation of the data.

One approach to fitting an oriented bounding box to a set of points is to apply PCA. Such a procedure is implemented in Open3D *get oriented bounding box* method. This procedure is based on a PCA of the convex hull of the input points [111]. When testing this implementation, it became obvious that this algorithm does not yield good results. Especially on sets of input points that represent an object with a cuboid shape as typically encountered in buildings, a misalignment of the bounding box (refer to Figure 4.10 left) can be observed. The misalignment is a result of the PCA, which tries to approximate the three main axes of the data, thus encompassing all variations of the point distribution. At the bottom, the result of our approach is visualized that robustly encapsulates the input set of points with a minimum bounding box.

The approach used in this work was guided by two assumptions. As buildings are organized along horizontal levels, it is sensible to reconstruct bounding boxes with horizontal top and bottom surfaces.

[1] Definitions of these terms can be found in the notations, see the *Terms and Definitions*

Figure 4.10: Comparison of bounding box fitting algorithms tested on sets of wall points with a cuboidal bounding shape. Left: oriented bounding box calculation based on PCA as implemented in Open3D [111]. Right: the approach developed in this work.

Along with this hypothesis, the bounding box fitting can be reduced to a 2D problem. This method we refer to as Horizontal Oriented Bounding Box (hobb) fitting. The Manhattan world assumption [21] partially applies to the horizontal constraint but bounding boxes can be fitted to data with any rotation around the vertical Z-axis.

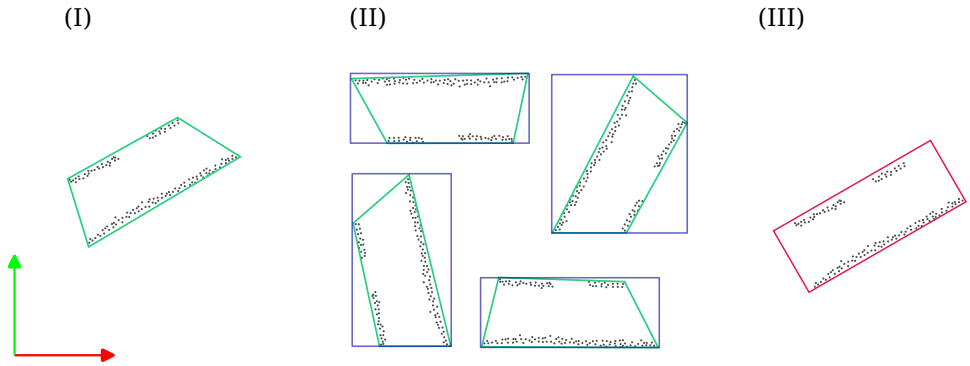


Figure 4.11: hobb fitting procedure on 2D projected data. Left: 2D projected input data with convex hull. Centre: Axis-aligned bounding box fitted to each edge rotated parallel to the X-axis. Right: Inverse rotation to minimal bounding box, final result.

Following the horizontal constraint, the Bounding Box fitting algorithm (refer to Algorithm 9) uses a 2D fitting approach. First, all points are projected onto the horizontal XY-plane by removing the Z coordinates from the data. A convex hull is then fitted to the data. The convex hull is a set of edges with a convex shape enclosing the data. With the convex hull, the hypothesis can be formulated so that one edge of the convex hull exists that is equal to the length of the minimum bounding box, i.e. connecting the first and second corner point (refer to Figure 2 in chapter). To identify this edge, each edge is rotated parallel to the X-axis, and an axis-aligned bounding box is fitted. Fitting axis-aligned bounding boxes comes at a low computational cost, as it only involves finding the X and Y minima and

Input : point cloud
Output: bounding box

```
   // find all candidates for horizontal aligned bounding boxes (hobb)
1  Function hobb candidate(edge, convex hull vertices):
2      angle = angle(Edge)                                  // angle w.r.t.  X axis
3      rotate(convex Hull vertices, angle)                    // parallel to X axis
        // essentially fitting an axis-aligned bounding box
4      min X, max X, min Y, max Y = get extrema(convex Hull vertices) // Get all four
        corner points; requires combinations of min / max X and Y values
5      hobb candidate = get corner points(min X, max X, min Y, max Y, min Z, max Z)
6      rotate(hobb candidate, - angle)                        // inverse rotation
7      return hobb candidate
   // fit the minimum hobb to the points
8  Function fit hobb(point cloud):
9      2d points = project to 2D(point cloud)
10     convex hull = convex hull(2d Points)
11     for edge in convex Hull do
12         hobb candidate = hobb candidate(edge, convex hull vertices)
13         candidate list = append (hobb candidate, candidate list)
14         volume = calculate volume(hobb candidate)
15         volume list = append(volume, volume list)
16     end
17     hobb = candidate list[where(min Volume)]          // find box with minimal volume
18     return hobb
19 bounding box = fit hobb(points)
20 return bounding Box
```

Algorithm 9: Horizontal oriented Bounding Box fitting.

maxima. For each of these axis-aligned bounding boxes, the volume is calculated. After iterating all edges of the convex hull, the bounding box with the minimum volume is chosen, the inverse rotation is applied, and the minimum bounding box is returned. The overall process is illustrated in Figure 4.11. Note that in Figure 4.11 (II), two rotations approximate the minimum bounding box, resulting in the same volume. A common solution in this case is to return the first of the two minima found. As both are rotated $180°$, there is no practical relevance to distinguish this case, which is rather unlikely in real-world data anyway.

It must be noted that a similar algorithm[2] was added to Open3D after our method was developed. The Open3D method applies the same principle in 3D by fitting an axis-aligned bounding box to the data from the frame of each triangle in the 3D convex hull [111]. This makes the algorithm applicable to more general cases and lets it find non-horizontal bounding boxes. However, horizontal aligned bounding boxes as fitted by our method are preferable for building reconstruction.

The time complexity of the Quickhull algorithm used for convex hull fitting is $O(n \log v)$ for both 2D and

[2]The method is named *get minimal oriented bounding box*. The documentation can be found here: `https://www.\ac{Open3D}.org/docs/release/python_api/\ac{Open3D}.geometry.PointCloud.html`

3D data, where n denotes the number of input points and v the number of output vertices [112]. This seems to result in a similar computational cost for both the 3D and our 2D hobb approach. On the same set of data, however, the number of 3D triangles will always be higher than the number of convex hull edges found on the 2D projected data. As an example, reducing the number of edges by the factor of 10 results in half the processing time. This underlines the positive effect on computation time of the hobb approach followed here.

The algorithm for fitting hobbs is meticulously applied to wall segments previously processed by the HYSAC algorithm. This operation is conducted on a per-storey basis. The outcome of this fitting process is a collection of bounding boxes that effectively approximate the geometry of the walls. An example of these wall bounding boxes is presented in Figure 4.12. It can be observed that the bounding boxes reconstructed represent the data well but still suffer from incorrect topology and unreasonable geometry. The key observations include that separate wall instances that share the same centreline are not connected, gaps occur at wall corners and small bounding boxes are present in the data, which should either be merged with larger walls along the same direction or treated as artefacts. Motivated by these observations, topology-aware bounding box correction routines are implemented.

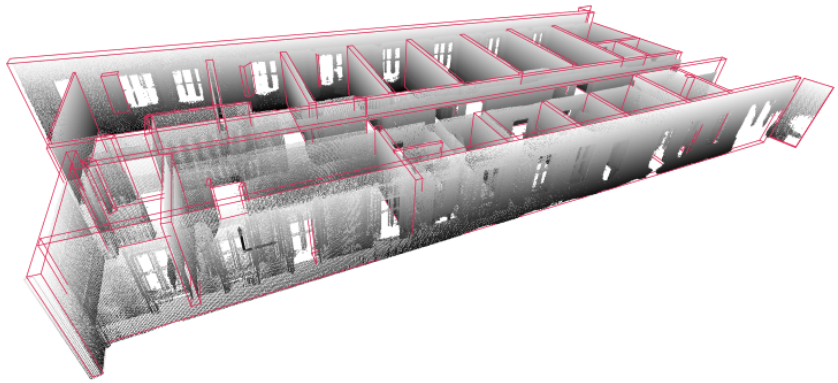


Figure 4.12: Initial wall bounding boxes with wall points.

4.6.5 Outside walls reconstruction

Ideally, outside walls should be reconstructed using the procedure as described. However, when outside walls are sparse, other procedures must be used. The basic concept is to reconstruct four walls around the perimeter of the building. The input source yielding the most accurate results is a density filtered point cloud (refer to Algorithm 4 in section 4.3). The algorithm (refer to Algorithm 10) then finds all endpoints of walls around the perimeter of the building and returns the bounding boxes connecting the respective endpoints with a predefined width.

Input : density filtered point cloud
Output: list[bounding box]

```
Function outside walls(density filtered point cloud):
    bounding box = fit hobb(density filtered point cloud)
     // get the four corner points in horizontal base plane
    end points = get endpoints(bounding box)
    for i in number of(end points) do
        // end points are in counter-clockwise order
        // get eight corner points of wall bounding box
        corner points = get corner points(end points [i], end points [i+1], width)
        bbox = bbox(corner points)
        append(wall bounding boxes, bbox) // append to list of wall bounding boxes
    end
return List[bounding box]
```

Algorithm 10: Outside walls.

Although this procedure can provide an exterior wall reconstruction from a sparse scan, it is susceptible to inaccuracies. Two primary factors contribute to these inaccuracies. First, a predefined wall thickness may not accurately represent the actual wall thickness. Second, the centre axis may not align with the outer perimeter, leading to misalignment in the reconstructed exterior walls. Both scenarios are likely to result in reduced reconstruction accuracy.

4.6.6 Topology-aware bounding box correction

One of the goals of this work is to provide a framework for topologically correct scan-to-BIM reconstruction. In the context of buildings, a topology describes the relationships of building elements and spaces and their respective sub-elements and sub-spaces. Applied to wall reconstruction, a topology-aware bounding box correction method ensures geometrically correct connections of walls. To develop an algorithm for topology correction, the following relevant configurations of wall instances are identified:

- Perpendicular bounding boxes intersect each other. Based on the configuration, the bounding box needs to be clipped or extended to form an edge or T-connection. See Figure 4.13 (I).
- Boxes oriented in the same direction need to be merged into one longer bounding box. See Figure 4.13 (II).
- Boxes intersecting each other, with one box comprising most of or the entire smaller bounding box, see Figure 4.13 (I). In this case, the smaller bounding box needs to be removed as it can be expected that it is an artefact.

In the development of the correction algorithms, two pivotal concepts apply. Firstly, the identification of pertinent configurations is achieved through an analysis of the geometric properties of the bounding boxes, which were previously adapted to fit the data. This approach utilizes the spatial characteristics

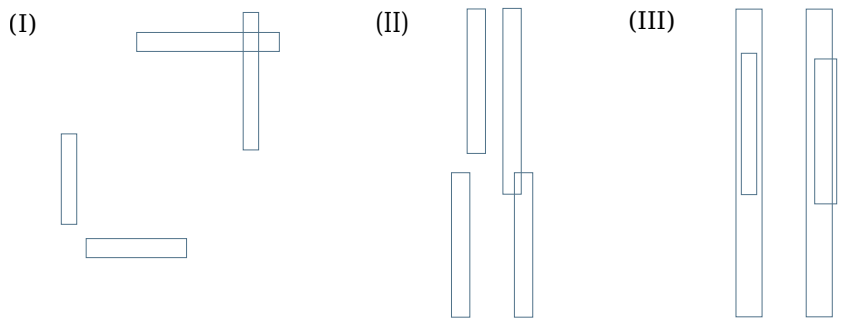


Figure 4.13: Configurations relevant to Bounding Box refinement.

of these boxes to discern structural details. Secondly, the algorithms are augmented with pre-existing knowledge about architectural norms, serving as guiding parameters in the reconstruction process. This knowledge encompasses certain dimensional constraints typical in building structures. For instance, walls that are either parallel or perpendicular to each other and separated by a distance exceeding 1.0 m should not be conflated; such a criterion is crucial in accurately defining narrow spaces, individual rooms and passageways. Additional dimensional thresholds, pertinent to these algorithms, will be elucidated in conjunction with their respective processing routines.

4.6.7 Clipping and extending perpendicular bounding boxes

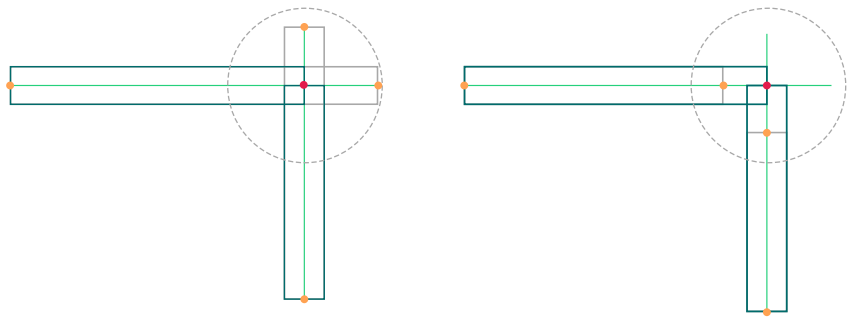


Figure 4.14: Extending and clipping bounding boxes.

For clipping bounding boxes two cases can occur (see Figure 4.14). Either the perpendicular bounding boxes intersect each other but expand further than the intersection point; or both adjacent perpendicular bounding boxes do not reach the intersection point but could be extended to form a wall corner. In both cases, bounding boxes and the respective endpoints for clipping or extending need to be identified. To achieve this, the baselines of all bounding boxes are derived. Baselines are the centre line of the lower four points spanning along the length of the bounding box.

Input : list[bounding box]
Input : distance threshold
Output: list[bounding box]

```
1  Function check and close(Bbox 1, Bbox 2, distance threshold, angle margin):
       // only perpendicular bounding boxes are candidates for closing
2      if is perpendicular(Bbox 1, Bbox 2) then
           // calculate intersecting point using the wall centre lines
3          intersection = line intersection(Bbox 1, Bbox 2)
4          if intersection then
5              if distance(endpoint, intersection point) ≤ distance threshold then
                   // extend / clip bounding boxes to form a corner
6                  new bbox 1 = create new bounding box(Bbox 1, intersection)
7                  new bbox 2 = create new bounding box(Bbox 2, intersection)
8                  return new bbox 1, new bbox 2
9              end
10         end
11     end
   // apply the algorithm on all combinations of bounding boxes
12 for bbox 1, bbox 2 in combinations(list[bounding box]) do
13     check and close(bbox 1, bbox 2, distance threshold)
14 end
15 return List[Bounding Box]
```

Algorithm 11: Close bounding boxes.

The algorithm to close bounding boxes is illustrated in Algorithm 11. From all combinations of baselines, the perpendicular pairs are identified. For these, the intersection points are calculated. From the intersection points, the closest endpoints of the bounding boxes are identified. If the distance of the closest endpoint of the bounding box to the intersection point is below a distance threshold, both bounding boxes' endpoints are extended onto the intersection point. For walls, the distance threshold should be in a range of 0.5 m to 1.5 m, depending on the configuration of the building and the smallest width of rooms to be expected. However, the distance threshold may only be chosen so that passages connecting spaces are not closed unintentionally.

During clipping and extending, the existing bounding boxes are modified, and no bounding boxes will be generated or deleted.

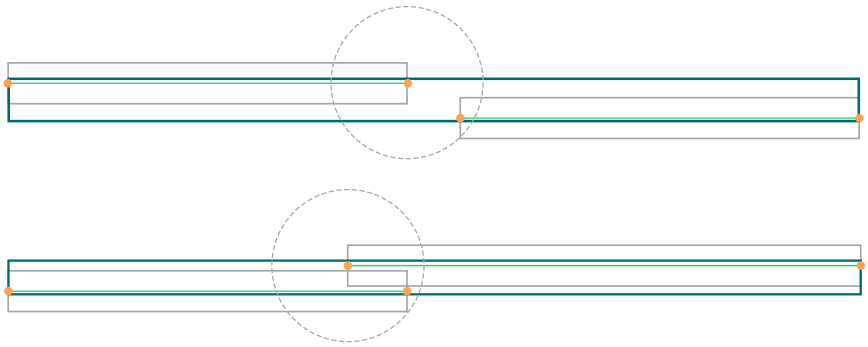


Figure 4.15: Merging bounding boxes.

4.6.8 Merging

Merging is applied to bounding boxes that share parallel, adjacent baselines. To identify these the endpoints of all bounding boxes are calculated. Typically, the bounding boxes relevant for merging are close to each other or intersect, with near endpoints below the distance threshold (see Figure 4.15). In the process of merging, however, the former two entities are combined into one bounding box.

The merging algorithm is illustrated in Algorithm 12 and includes the following procedure. First, in all combinations parallel bounding boxes within an angle margin are identified. For parallel bounding boxes, the endpoints are retrieved. Any distance between the endpoints is then calculated. If one of the distances is below the distance threshold, the wall is considered for merging. The distance threshold for merging can be chosen based on previous knowledge about the building topology and should be set so that narrow hallways or passages are not blocked by merged walls. The endpoints the furthest apart are then used to deliver a new bounding box that represents both bounding boxes. The Z value of both new endpoints is set to the mean of the previous ones. This is required to ensure horizontality according to the hobb definition. The new endpoints, mean width and height of the combined bounding boxes are then used to calculate the eight corner points of a new bounding box.

Other than for the clipping and extending, in the process of merging the former two entities are combined into one bounding box.

It is recommended to apply the closing and merging procedure iteratively to the bounding boxes. Each routine changes the set of bounding boxes, and the altered bounding boxes might be within the thresholds for closing and merging again. The number of bounding boxes changed decreases after several iterations. To the experience of the author, the number of bounding boxes reliably converges to 0 when applying both closing and merging four times.

Input : list[bounding box]
Input : distance threshold
Output: list[bounding box]

```
Function merge(bbox 1, bbox 2, distance threshold):
    // find parallel bounding boxes
    if is parallel(bbox 1, bbox 2) then
        end points 1 = get endpoints(bbox 1)
        end points 2 = get endpoints(bbox 1)
        if distance(combinations(end points 1, end points 2) ≤ distance threshold) then
            // new endpoints are the main input for the new bounding box
            new endpoints = new endpoints(end points 1, end points 2)
            // the original bounding boxes are needed to calculate the mean
             width and height for the new one
            new bbox = create new bounding box(new endpoints, bbox 1, bbox 2)
            return new bbox
        end
    end
for bbox 1, bbox 2 in combinations(list[bounding box]) do
    merge(bbox 1, bbox 2, distance threshold)
end
return list[bounding box]
```

Algorithm 12: Merge bounding boxes.

The primary approach applied is to eliminate wall artefacts, i.e. small wall bounding boxes at positions not related to actual walls, by closing and merging. This procedure covers most of the situations when wall artefacts are reconstructed based on incomplete input data, incomplete segmentation and other errors in the processing. Ultimately, wall artefacts can be removed from the data if they could not be incorporated into topologically correct wall instances before.

4.6.9 Removing wall artefacts

Although the merging and closing algorithms consolidate the walls reconstructed by combining smaller walls into larger objects, some wall artefacts can still remain. In this case, wall artefacts can be characterized as small bounding boxes intersecting larger ones. Such wall artefacts can be present in case of sparse walls resulting in the reconstruction of numerous small wall objects. Typically, these small wall objects intersect either the outside walls reconstructed separately, or larger merged and closed wall objects. In any case, wall artefacts need to be removed from the data. A typical example of a wall artefact removal is presented in Figure 4.16.

Algorithm 13 describes the procedure. Intersecting bounding boxes are identified by checking if any corner points of one bounding box are inside the other bounding box and vice versa. If intersecting bounding boxes are found, the volume ratio is then calculated. Whenever the volume ratio is below a predefined limit, i.e. whenever one bounding box is considerably smaller than the other, this bounding box is considered as a wall artefact and removed from the list. The volume ratio limit should be set to

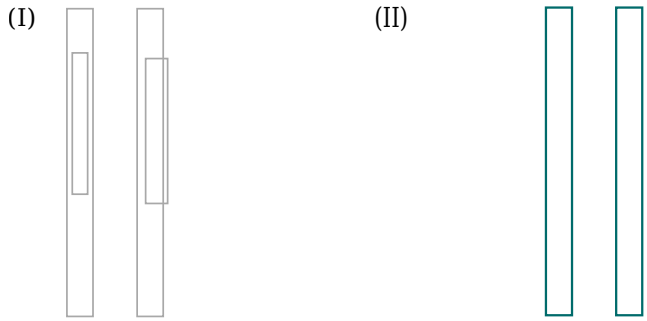


Figure 4.16: Removing intersecting bounding boxes. Left: original bounding boxes. Right: Remaining, bounding box artefacts removed.

less than 40%, depending on the characteristics of the data.

Input : list[bbox]
Output: list[bbox]
Function `intersects` *(bbox 1, bbox 2)*:
  if `any(get corner points(`*bbox 1*`))` in *bbox 2* **then**
    return *True*
  end
for *bbox 1, bbox 2 in* `combinations(`*list[bbox]*`)` **do**
  if `intersects(`*bbox 1, bbox 2*`)` **then**
    volume ratio = `volume(`*bbox 1*`)`/`volume(`*bbox 2*`)`
    if *volume ratio* $\leq$ *max volume ratio* **then**
      `remove(`*list[bbox], bbox 1*`)` `// and vice versa if condition is met`
    end
  end
end
return *list[bounding box]*

Algorithm 13: Wall artefact removing.

To identify intersecting bounding boxes, it is not sufficient to check only for corner points inside the other bounding box. A plain intersection, where neither corner is inside the other box, also needs to be considered. However, when the algorithms developed here are applied sequentially, the merging and closing algorithms handle these intersecting bounding boxes effectively. Additionally, the wall artefact removal algorithm, as described, reliably eliminates most other artefacts. Nevertheless, in cases of sparse data with low quality and erroneous segmentation, some wall artefacts may remain that need to be identified. This is a challenging problem, as a wide range of configurations need to be considered and these cannot be easily distinguished from a TP. Such examples are given in Figure 4.17. In these cases, the segmentation should be improved, and a sufficient data quality should be ensured in the first place to guarantee an accurate reconstruction.

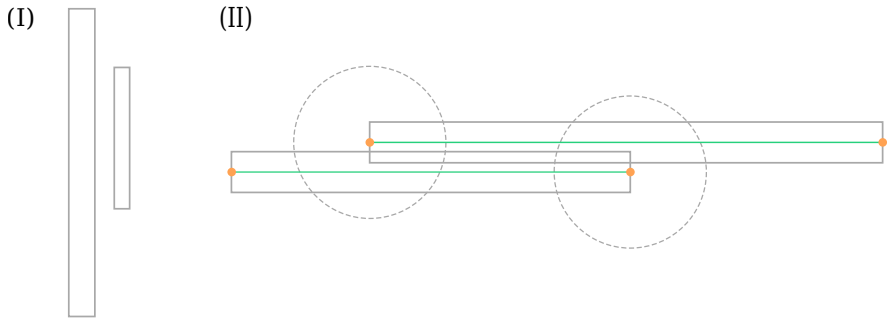


Figure 4.17: Bounding boxes not removed using the proposed wall artefact removal method. Left: Bounding box artefact not intersecting, thus not removed. Right: Bounding boxes overlapping beyond the merging threshold.

4.7 Child objects: the case of doors

In BIM, opening objects are represented as child objects of enclosing objects. This is the case for doors, which typically exist as the child object of a wall. The core principle to reconstructing such child objects is to use the parent object's geometry, identify points assigned to the child object and reconstruct the child object. In the following, the method is described for doors only, as it can be adopted to process windows easily. The main limiting factor here is that the points representing the frames of either doors or windows must be distinguished from wall points accurately for a precise reconstruction. Methods to mitigate incomplete segmentation of door frames will be introduced here, but remain to be developed for windows reconstruction.

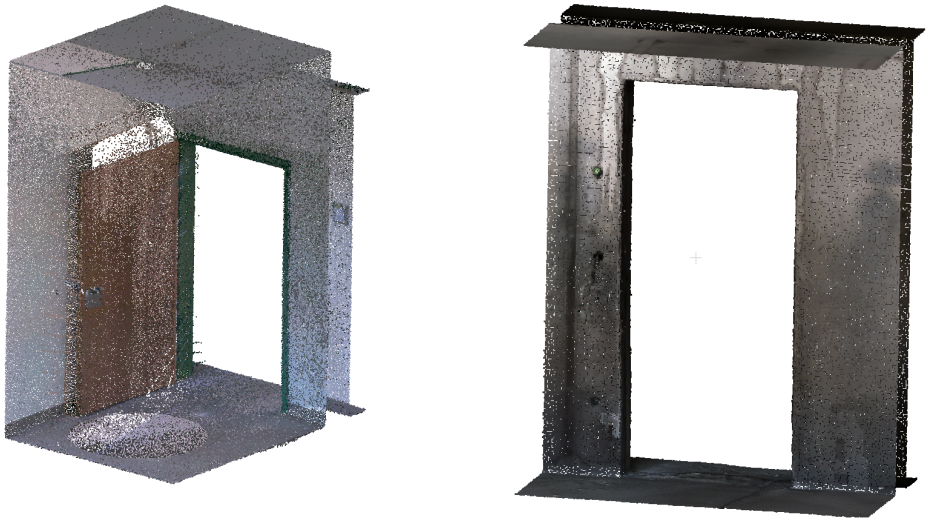


Figure 4.18: Sections of scanned scenes. Left: Door with frame, lining and door leaf points that can be segmented. Right: Plain opening, no dedicated objects to be segmented.

A door incorporates a dedicated object inside an opening comprising a door frame, lining and one or more door leafs, panels, etc. These objects are represented by points in the input data and can thus be segmented and reconstructed. On the other hand, plain openings of walls can fulfil the same purpose as a door, i.e. to connect two spaces, but an opening as such does not contain any dedicated objects and will thus be represented by a void space in the data. Such voids in objects are notoriously hard to reconstruct and require a completely different approach than the one described here. One example of each is given in Figure 4.18.

An overview of the door reconstruction algorithm is given in Algorithm 14. It is initiated by searching for door points in and close to every potential parent object, in this case in each previously reconstructed wall instance. Any door points assigned to the parent object will then be clustered to separate the individual door instances. In cases of under segmentation, it is beneficial to find the points representing the

Input : wall bounding boxes
Input : IfcWall

```
1 Function door reconstruction(wall bounding boxes, IfcWall):
      // iterate both wall bounding boxes and IfcWall
      // we need the IfcWall to assign parent ID to IfcDoor
2     for wall in wall bounding boxes, IfcWall do
          // Use wall bounding box, expand search space, cluster points
3         door clusters = assign door points(wall bounding box, door points)
          // Segment door leafs, rotate into door frame
4         closed door clusters = close doors(door clusters)
          // fit bounding box, project into parent
5         door bounding boxes = fit door bounding box(closed door clusters)
6     end
7     return IfcDoor
8 ifc door(s) = door reconstruction(wall bounding boxes, IfcWall)
```

Algorithm 14: Door reconstruction.

door leaf to then literally close the door. Such closed door instances can help to accurately reconstruct the door dimensions. After these procedures, a bounding box is fitted to the data, which is projected into the parent geometry to account for flush surfaces of the reconstructed doors and parent walls. In contrast to wall and column reconstruction, the IFC door object needs to be created while iterating all parent objects. This is related to the fact that the parent object needs to be assigned to each door object. The most practical solution is to directly create IFC door objects to avoid the error-prone later assignment of parent IDs.

Each of the methods including door points assignment, door closing and door bounding box fitting will be explained in a detailed fashion in the next sections. As illustrated in Algorithm 14, the methods need to be applied iteratively to all wall instances to reconstruct all doors of the scene.

4.7.1 Door points assignment and clustering

As doors are child objects of walls, the first step in the process is to find door points associated to the respective wall object. As doors are kept open during data acquisition to ensure a good overlap of multiple scan positions or in mobile scanning procedures, a two-step approach is followed to ensure that all door points are assigned to the wall. First, any door points inside the wall bounding box are identified. Naturally, these represent parts of the door frame, lining and door leaf when closed during scanning. Only, if such points are present, the wall bounding box is expanded to find all points of the door instances, mostly covering the door leafs if the door was kept open.

By following this procedure, it can be assured that doors are assigned to the correct walls with the opening and door frames. When directly applying the expanded search space, errors might occur as open doors adjacent to perpendicular walls (see Figure 4.19, right side) might be assigned to the wrong wall.

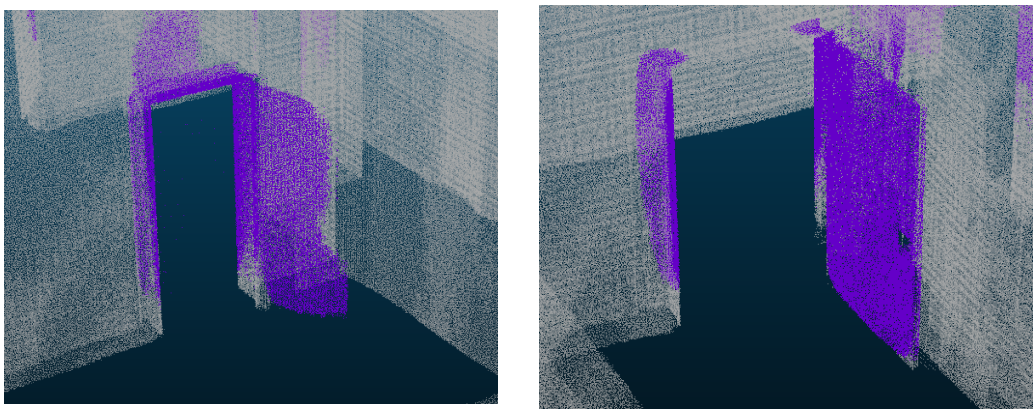


Figure 4.19: Examples of segmented doors. Left: good segmentation with door leaf and door frame. Right: under segmentation of door frame and door leaf. Wall points are displayed transparently.

The result of this procedure is that a number of door points are assigned to the wall object. However, these points might still contain several door instances. To find these, DBSCAN [68] is applied, which identifies point clusters forming the door instances. The algorithm can be found in Algorithm 15.

Input : wall bounding box
Input : door points
Output: door clusters
Function `assign door points(`*wall bounding box, door points*`):`
    door frame points = `points in bounding box(`*wall bounding box, door points*`)`
    if `any(`*points*`)` **then**
        `expand bounding box(`*wall bounding box, search offset*`)`
        all door points = `points in bounding box(`*wall bounding box, door points*`)`
        return *all door points*
    end
    `// reset wall bounding box to original size`
    `expand bounding box(`*wall bounding box, - search offset*`)`
all door points = `assign door points(`*wall bounding box, door points*`)`
door clusters = `cluster DBSCAN(`*all door points*`)`
return *door clusters*

Algorithm 15: Door points assignment and clustering.

However, the segmentation of door frame and linings may often be incomplete and is likely to fail resulting in under or over segmentation (see Figure 4.19). Among others, factors such as similar features of door frame and incorporating wall, various types of door frames not recognized by the segmentation model, and a lack of generalization in the segmentation model can contribute to such segmentation errors. As it can be observed in Figure 4.19, the segmentation of door leafs might be more accurate. In any case, the door leaf points can provide valuable information for the door reconstruction. Thus, the door leaf points should be integrated into the reconstruction procedures by closing the doors.

4.7.2 Closing the doors

As doors are typically open during scanning, and present as such in the data, the door leaf needs to be assigned to the respective door frame and wall. This process is referred to as *closing the doors*, as the physical turn of the door leaf must also be accomplished within the point cloud thus facilitating an accurate reconstruction of the consolidated door points. The algorithm includes segmenting the door leaf, calculating the rotation angle, performing the rotation, and a voxel point cloud filtering to eliminate noise. An overview is given in Algorithm 16.

Input : door cluster
Input : wall normal
Output: closed door points
Function `close doors(`*door cluster, wall normal*`)`**:**
    `// in this case, plane inliers are door leaf points`
    plane, door leaf points = `fit ransac plane(`*door cluster, min points per plane,* ε *= 5 cm*`)`
    if `any(`*door leaf points*`)` **then**
        θ = `angle` (plane normal, wall normal)
        door leaf points = `rotate(`*door leaf points,* θ`)` `// check sense of rotation`
    end
    closed door points = `voxel pcd filtering(`*door leaf points, voxel size, min number of points*`)`
    return *closed door points* `// door leaf now rotated into rest of door points`
closed door points = `close doors(`*door cluster, wall normal*`)`

Algorithm 16: Closing the doors.

First, door leaf points must be distinguished from the door frame and lining. To this end, we assume that door frame and lining are typically located inside the wall bounding box. Thus, on any door points outside the wall bounding box, a RANSAC [64] plane segmentation is performed to approximate the surface of the door leaf. The ε parameter which defines the maximum distance of a point to the reconstructed plane is set to 5 cm, anticipating a maximum thickness of 10 cm for the door leaf. Segmented planes are discarded when the number of points is below a threshold, and only vertical planes are accepted. This procedure retrieves all door leaf points and the respective plane equation, if those are present in the data.

To rotate the door leaf into the door frame, both the rotation centre and axis as well as the rotation angle need to be calculated. The rotation centre is found by intersecting the door leaf plane with the adjacent wall surface. Following the Manhattan world assumption [21], this problem can be solved in 2D, thus intersecting the door leaf centre line with the closest wall boundary. The intersection point is used as the rotation centre, with the z-axis as the rotation axis.

The normal vectors of the door leaf plane and of the side plane of the wall bounding box are used to calculate the angle θ of the door leaf and wall, i.e. the rotation angle to close the door. θ is calculated using the $\arccos$, thus resulting in the smallest angle. The sense of the rotation cannot be determined either. Thus, rotations of $\theta \pm 180$ are applied to the door leaf plane. On each rotation, the hobb is fitted, and its volume calculated to identify the smallest closed door bounding box.

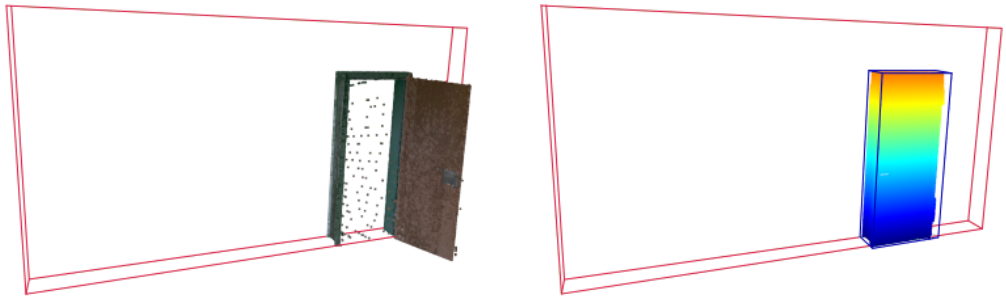


Figure 4.20: Close doors algorithm. Left: Wall bounding box and original door points. Right: Wall bounding box, closed and filtered wall points and door bounding box.

During the procedure described above, it is likely that some artefacts of the door leaf and other clutter remain in the closed door points. To remove these, the voxel density filtering algorithm as described in section 4.3 is applied. Thus, the later reconstruction can deliver accurate results without deviations caused by clutter or artefacts present in the data, as illustrated in Figure 4.20.

4.7.3 Bounding box fitting and refinement

Per closed door cluster, a bounding box is fitted to the points. Depending on the previous processes, two inconsistencies may occur. On the one hand, the bounding box might exceed the wall width, as door frame, linings and even the door leaf might be on or outside the wall surfaces. On the other hand, under segmentation may lead to an incomplete bounding box reconstruction with the resulting bounding box inside the wall bounding box. In both cases, the door bounding box needs to be projected into the wall bounding box thus ensuring that both the wall and door bounding box vertical intersecting surfaces are flush.

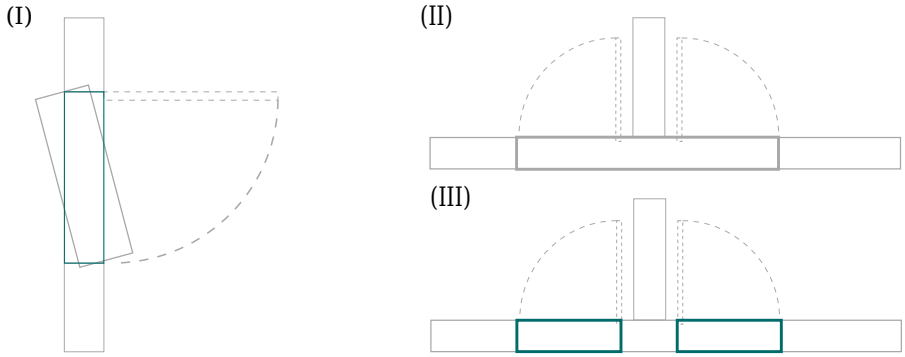


Figure 4.21: Bounding box refinement. Left: projecting the bounding box into the parent geometry. Right: splitting bounding boxes too wide.

One configuration and the result of projecting the door bounding box into the parent wall are presented

Input : door bounding box
Input : wall bounding box
Output: door bounding box

Function `project into parent(`*door bounding box, wall bounding box*`):`
    `// angle between door and wall bounding box`
    `// normal of the larger vertical surface is used`
    angle = `angle(`*wall bounding box normal, door bounding box normal*`)`
    `rotate(`*door bounding box, angle*`)`
    `// parent plane, use two points along the width of the parent`
    normal vector = `norm(`*point[3] - point[0]*`)`
    normal vector = a, b, c
    d = $-$`sum(`$a * x + b * y + c * z$`)`
    plane equation = $a * x + b * y + c * z + d = 0$
    `// transform each corner point`
    for *corner point in bounding box* **do**
        distance = $|a * x + b * y + c * z + d|$
        translation = $normal\ vector * distance$
        corner point += translation
    end
    return *door bounding box*

Algorithm 17: Project into parent.

in Figure 4.21 (I). The *Project into parent* algorithm (refer to Algorithm 17) involves the following steps: First, the door bounding box is rotated onto the wall bounding box, i.e. parallel to the wall. Then, the plane equation of the parent plane is calculated using a normed vector along the width of the bounding box and one of the corner points. The distance of all door bounding box corner points to the parent plane is then calculated. From the distances, the translation vector is calculated by multiplying the normal vector and the distances. The translation is then applied to the bounding box to retrieve the projected door bounding box. Along with the definition of hobb, this algorithm requires vertical surfaces of both the door and the parent bounding box.

Depending on the parameters used to apply the DBSCAN algorithm [68], two separate door instances might be combined to one door (see Figure 4.21 (II)). This applies to doors adjacent to each other, e.g. 50 cm or less distance of the closest door points between both instances. Depending on the topology of the building, such configurations can be encountered frequently, e.g. in office buildings with mirrored rooms and respective doors beside the same interior wall. Ultimately, a reconstructed door might intersect a wall. Clearly, the erroneous bounding box needs to be split.

To identify candidates for splitting, i.e. door bounding boxes too long to represent one door entity, a maximum width of doors is defined based on the topology of the building. All bounding boxes longer than the threshold will be split into two separate door instances. Along the split surface, an offset can be applied to ensure a realistic distance in between the doors (see Figure 4.21 (III)).

4.7.4 Inverse opening reconstruction

Similar to outside wall reconstruction, Algorithm 18 describes an alternative method that has been developed to reconstruct doors in cases of under segmentation. The core principle is not to reconstruct doors based on door points but by finding openings in the parent object. A general notion is that openings are typically encountered in a certain vertical corridor, e.g. in between 0 m and 2 m above the floor level. Thus, a slice of the wall points is used for further processing. On such slices, DBSCAN [68] (refer to Algorithm 2) is applied to segment clusters while discarding miscellaneous clutter points. Voids in between clusters are considered as opening candidates. To distinguish door or window openings from miscellaneous other openings, dimension limits can be applied automatically. Openings wider than 0.8 m with a height that humans can traverse would be considered as doors, likewise windows with an additional height constraint for the parapets.

```
   Input  : wall points
   Output: list[bbox]
 1 sliced points = z slice(wall points, min Z, max Z)
 2 clusters = dbscan(sliced points)
 3 opening candidates = find voids(clusters) // find voids in between point clusters
 4 for opening in opening candidates do
 5 |   if width(opening) ≥ minimum door width then
 6 |   |   bbox(opening)
 7 |   |   for door in door list do
 8 |   |   |   if door not intersects opening then
 9 |   |   |   |   append(door list, opening)     // If there is a door, discard opening
10 |   |   |   end
11 |   |   end
12 |   end
13 end
14 return List[bounding box]
```

Algorithm 18: Inverse door reconstruction.

4.8 Other structural components: columns reconstruction

The layout of a building is primarily determined by walls and elements attached to them, like doors. Walls may bear loads, but other structural components like columns, beams and trusses also contribute to the building structure. Unlike walls, these components do not have sub-elements but they can be interconnected in a way that forms the building's framework, such as the combination of columns and beams in a skeleton structure. Similar to how walls are divided across different floors of a building, these load-bearing components are also typically separated by levels. This discussion will focus specifically on columns to illustrate how such structural elements can be reconstructed.

4.8.1 Clustering

In alignment with prior reconstruction methods, the process begins with a collection of points classified as 'column' during semantic segmentation. The segmentation of these points into distinct columns is accomplished using DBSCAN [68] (refer to Algorithm 2 in section 2.3). This approach is effective and dependable, as columns are usually spaced apart, facilitating straightforward clustering. Artefactual points in smaller groups are filtered out by imposing a minimum points per cluster limit. This limit is determined based on the point cloud's density, typically ranging from around 100 to 5000 points.

4.8.2 Distinguishing round and square columns

Columns are typically found in round, square, or rectangular shapes. For a detailed reconstruction, these shapes need to be distinguished using the following two-step procedure. First, the curvature is calculated. This involves building the k-d tree for an efficient nearest neighbour search, calculating the covariance matrix and respective eigenvalues. The curvature is retrieved by dividing the third Eigen value by the sum of the first, second and third Eigen values:

$$curvature = \frac{\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3} \tag{4.1}$$

The curvature values are modelled using a normal distribution, with the Maximum Likelihood Estimation (MLE) algorithm applied for fitting. This process utilizes the standard deviation to differentiate between round and square columns. The rationale is that round columns tend to exhibit consistent curvature, leading to a lower standard deviation, whereas square or rectangular columns display a broader spectrum of curvature values, culminating in a higher standard deviation. In Figure 4.22, the curvature of a sample of round and square columns is displayed as a heat map.

4.8.3 Bounding box and cylinder reconstruction

Depending on the shape of the column instance, either bounding box fitting or RANSAC cylinder fitting can be employed to retrieve the geometry of the object.

The bounding box fitting algorithm is detailed in previous chapters, specifically in Algorithm 9 in subsection 4.6.4. Following a similar method, this process involves projecting all points onto the XY base plane, fitting a convex hull encompassing all data fit a bounding box for each convex hull edge thus minimizing the volume of the bounding box created. It is important to note that in cases of over segmentation, floor and ceiling points adjacent to the column may be included in the cluster, leading to bounding boxes that are excessively large and not accurately representative of the column itself. To address this, further refinement steps as outlined in section 4.9 might be necessary.

Alternatively, parts of the column affected by potential over segmentation can be omitted from the reconstruction by horizontally clipping the column just above the floor and below the ceiling. The remaining

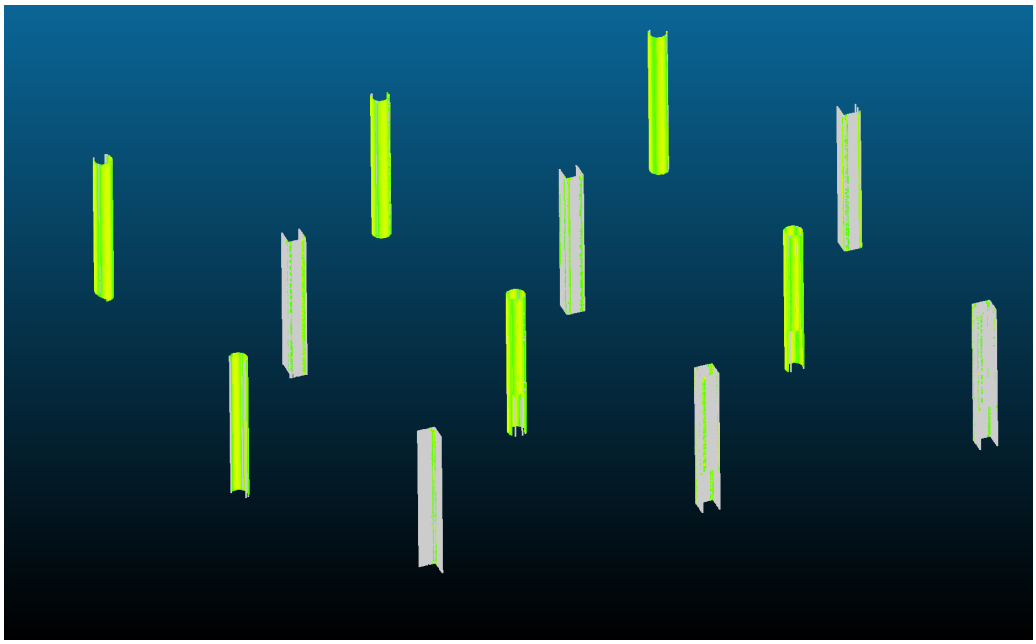


Figure 4.22: Distinguishing round and square columns by estimating the curvature. The curvature is displayed as a heat map, with light grey representing low curvature, and green representing high curvature.

section of the column can then be used for a more precise bounding box reconstruction. This is illustrated in Figure 4.23. On the left side, a horizontal slice is displayed resulting in a reconstructed bounding box with approx. 0.5 x 0.7 m cross section. On the left side, however, the over segmented column instance is shown with a cross section of 1.1 m x 1.1 m.

Adjustments to the column height are made either to match the original points or align with the building levels identified earlier, as mentioned in section 4.5.

In a manner akin to the HYSAC method previously described, certain constraints can be implemented in the process of fitting a cylinder to a column. A key constraint is the assumption that columns are vertical. By applying this, the cylinder fitting task simplifies to a 2D problem, achievable by projecting all points onto the XY base plane within a Euclidean space. The column's vertical extent is determined by the point set's global minimum and maximum z-values. To fit a circle to the projected points, a random selection of three points is used, and a circle is then fitted to these. With the defined circle and the known minimum and maximum z-values, a cylinder can be constructed. The optimal cylinder fit is determined by iteratively minimizing the distance from the points to the cylinder. Similar to standard RANSAC methodologies, the number of iterations is an adjustable parameter of the function, which ultimately yields the cylinder's centre point, radius, and axis.

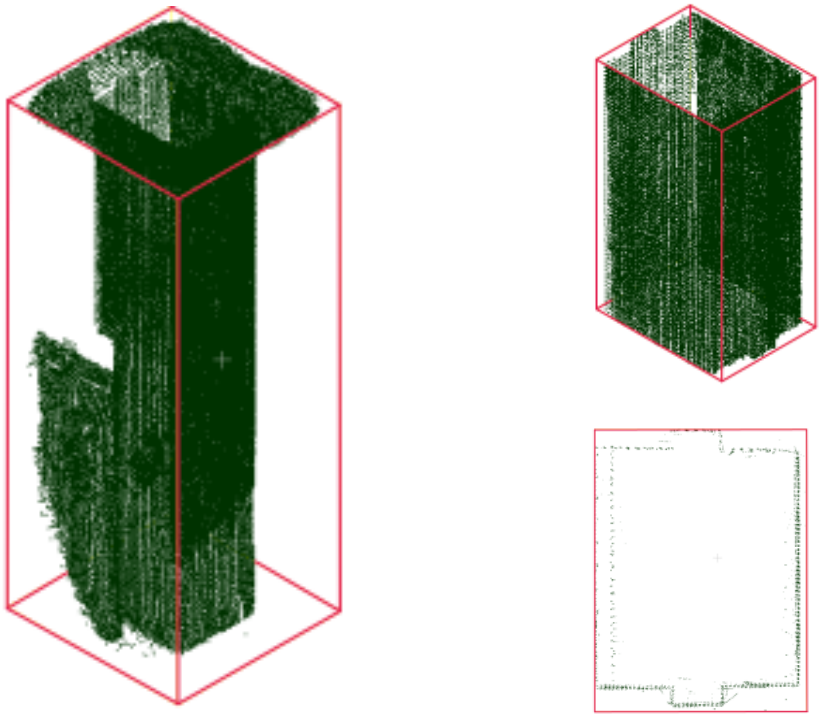


Figure 4.23: Column over segmentation and horizontal slicing for an accurate reconstruction. Left: segmentation of column instance. Right: horizontal slice providing an accurate reconstruction.

4.9 Enhanced methods for further filtering and refinement

By applying the reconstruction procedures as outlined in the previous sections, an accurate reconstruction can be retrieved. However, wrong or preposterous dimensions of objects can still occur, mainly caused by incomplete input data, sparse segmentation or faulty behaviour of the reconstruction algorithms. To deliver a correct BIM model, further filtering and refinement procedures need to be developed and applied. Dimension and volume limits as well as topology-aware dimension adjustment are the two relevant approaches. While the first tackles the problem of insufficient clearance and elements too diminutive, the latter applies basic topology rules to correct the dimensions of the objects previously reconstructed.

The approaches are explained on a theoretical basis here. The specific parameters along with the application examples can be found in chapter 6 and Annex 7.

4.9.1 Dimension and volume limits

During the previous reconstruction procedures, several dimension and volume limits have been applied implicitly, e.g. splitting doors too wide, removing wall artefacts, etc. However, it is sensible to apply a final filtering using dimension and volume limits to eliminate any remainders of the previous processes. The main approach is to identify elements too small or too big and to either adjust the dimension or remove such instances from the set of bounding boxes. This includes:

- Wall minimum volume: Walls with a small volume should be excluded from the reconstruction. Most small walls initially reconstructed would be merged or closed so that remainders with a small volume can be considered as artefacts. A wall minimum volume in the range of $0.5\ m^3$ to $1.5\ m^3$ seems appropriate. Setting the volume limit depends on the quality of the data and segmentation and the type of the building. A volume limit is sufficient here as mostly small wall artefacts of arbitrary shape remain after the previous reconstruction procedures that can be identified just by their small size.

- Doors require a minimum clearance, hence bounding boxes with a length below the clearance limit can be extended to a predefined length. A door minimum clearance of $0.8\ m$ seems appropriate for office style buildings as in the CV4AEC dataset. For residential buildings, a minimum cleareance of $0.6\ m$ can be applied, too. Doors below the limit can be discarded. However, the more practical solution is to extend the width of the door bounding box to the predefined minimum width.

- Likewise, the height of objects that humans traverse, e.g. doors, require a minimum height to fulfil their purpose. The minimum height depends on several parameters such as the age of the building, regulations and standards. Reconstructed doors too low can be set to the minimum height to ensure a consistent reconstruction.

- On the columns reconstructed, both a maximum volume limit and minimum or maximum dimensions can be applied. Considering both is necessary here to ensure that only slim elements are considered in the columns reconstruction. Regardless of the material, one would expect that the column cross-section exceeds $10\ cm\ x\ 10\ cm$ at minimum for any material used for load-bearing columns. On the other hand, columns too large would rather be considered as walls, thus a maximum volume threshold can be applied. Likewise, a maximum cross-section limit can be applied.

To set these limits, standards and building codes can be consulted. For concrete columns, the national annex to Eurocode 2 sets a minimum dimension of $200\ mm$ in either direction [113]. Likewise, other building codes define similar limits. On the other hand, including some civil engineering knowledge can help to set the limits. In any case, there is no guarantee that the building complies with the standards.

Clearly, applying these dimension and volume limits to remove artefacts cannot replace a more complete scan or better segmentation results. With high quality input data it is even likely that these geometric correction rules seldom apply as the errors do not occur or will be caught by the reconstruction algorithms themselves.

However, in cases of poor input data quality and under segmentation, applying dimension and volume

limits can improve the results. Applying these refinement measures in the context of other elements, e.g. to remove a smaller wall intersecting a larger one, yields better results than a stand-alone. Beyond these considerations, incorporating the topology of the building into correction methods can be beneficial for the reconstruction.

4.9.2 Topology-aware geometry adjustment

Depending on the completeness of the data and the segmentation accuracy, another option to increase the quality of the reconstruction is to set the height or the respective horizontal boundaries to the levels calculated as described in section 4.5. Among others, the following principles and hypotheses can be used to determine which objects' horizontal boundaries should be set in relation to the levels:

- Load bearing objects, e.g. structural walls and columns, need to be connected to the horizontal structural members to ensure structural integrity. Although the surfaces of the load bearing elements might be not visible depending on the state of the building scanned, these objects can at least be expanded to the floor and ceiling levels. Even an offset can be considered, if the distance from the floor and suspended ceiling to the structural horizontal member can be estimated by single manual measurements.

- The bottom surface of objects that humans traverse, e.g. doors, should be aligned with the floor level height. Hence, the lower horizontal door boundary can be set to the floor level.

- Topologically, any building consists of confined spaces. These need to be enclosed by building objects. Although our approach does not include a spatial reconstruction, it could be an option to identify confined spaces based on the building objects previously reconstructed. Based on these spatial representations, further topology checks could be applied including checking for accessibility, i.e. each space needs to be accessible by an opening or door. On a higher level, the topology could be checked for patterns including hallways to access each separate room, the connection towards the entrance and stairways of the building, etc. However, such topology checks have not been developed and applied yet.

The above refinement procedures can also be applied to other types of elements not covered in this study depending on their topology and specifications, e.g., a height minimum clearance below beams.

Beyond the reconstruction procedures, refinement and topology adjustment, it can be beneficial to implement a feedback loop to the input point cloud, to assess the geometric confidence and handle reconstruction errors.

4.10 Geometric confidence and error handling

When introducing the reconstruction algorithms, it has been noted that measures should be implemented to assess the confidence and accuracy of the reconstruction. This is essential to catch errors, handle them, and ultimately issue warnings while documenting the overall reconstruction accuracy. This

task encompasses two major fields: (i) evaluating the overall accuracy of the reconstruction, which is discussed in section 6.2, and (ii) identifying errors within the individual processing steps.

Although these two fields might seem similar, it is crucial not only to assess the overall reconstruction accuracy but also to identify the specific elements that might contain errors. Such errors include faulty shapes and the general fit of the reconstruction to the data. Several procedures can be implemented to identify these respective errors:

- Errors related to the Manhattan-world assumption [21] include any non-axis-aligned objects, such as oblique walls.
- Assuming horizontal surfaces might result in erroneous reconstructions when sloped floors and ceilings, including ramps, are encountered.
- If a cylindrical column has been reconstructed as a cuboid, this indicates an error related to primitive shape fitting.
- Generally, the input points should cover the surfaces of the reconstructed geometry in a manner similar to how they would cover the real-world physical object. When the input points are either inside or outside the reconstructed geometry with a noticeable distance to the primitive surfaces, this indicates an error related to geometric approximation.

The following strategies can be implemented as mitigation measures along with the errors identified:

- To determine if the Manhattan-world assumption [21] is applicable at the level of building component reconstruction, a PCA can be applied to identify the main vectors characterizing the data. If the first two principal components are neither horizontal nor parallel to the X or Y axis, further routines should be applied to reconstruct the more complex geometry.
- Similarly, investigating the third principal component can reveal sloped objects. In such cases, more detailed investigation and elaborate reconstruction procedures should be applied.
- To identify shapes and primitives inappropriately fit, the error metrics of the fitting algorithms themselves can be investigated. To this end, carefully chosen error rates for accepting a primitive can primarily avoid inappropriate fitting. This strategy will primarily be followed in this work, but it increases the risk of discarding geometry based only on a bad fit of the primitive shape. On the other hand, even accepted primitives can fit the data badly. To identify these, metrics such as the point-to-primitive distance can be investigated and ultimately, other primitive shapes could be fit that approximate the data better. The surface curvature (refer to section 4.8) can be used to distinguish round and square shapes.
- Generally, the geometry reconstructed should fit the input data. One approach to quantify the *fit* of the geometry is to calculate the point-to-primitive distance. The mean of all distances is illustrative of how well the geometry fits the data. Depending on the noise in the data, a mean distance of less than 5 cm can be accepted as a good fit and in the case of worse distances, further investigations should be implemented.

Although some mitigation measures can be applied automatically, it is recommended to at least raise

an exception along with the error parameters. Ultimately, human intervention might be appropriate to correct the reconstruction or discard the reconstructed object.

The described error mitigation measures are only partially implemented in this work. Most erroneous reconstructions are avoided by carefully selecting the parameters for the reconstruction algorithms. Before presenting the results of this approach, the implementation of both the geometric reconstruction and open BIM authoring algorithms and routines will be discussed.

4.11 *Pystruct3d*: efficient geometry reconstruction algorithms

In conjunction with this research, we have created and released two open source Python libraries: *openbimxd* and *pystruct3d*. The *openbimxd* package, which includes techniques for creating BIM objects from the reconstructed data, will be explored in the following chapter. Meanwhile, *pystruct3d* is presented in this section, encompassing implementations of all the algorithms discussed in this chapter, complete with an extensive API.

Pystruct3d is composed of three distinct modules: (i) The *bbox* module, which includes a class for bounding box representation along with methods for their fitting, manipulation, and combination. (ii) The *metrics* module, which is equipped with methods to assess the precision of bounding box reconstructions against established sets of reference bounding boxes. (iii) The *visualization* module, featuring a visualizer class designed for the display of bounding boxes, individual points, and point clouds, serving as a tool for testing and diagnostic purposes. The *pystruct3d* module leverages high-performance libraries like NumPy [108] and SciPy [114] to optimize computational efficiency. In the realm of point cloud handling, Open3D [111] is utilized, providing a plethora of methods and algorithms applied in this study. The visualizer class also employs Open3D [111]. The package is available at `https://github.com/humantecheu/pystruct3d`. The three modules along with their classes and methods will be described in the following sections.

The subsequent descriptions maintain consistency with the Python module's nomenclature, with the class and method names written in *italics*.

bbox: Fitting and manipulating bounding boxes

The bounding box module comprises the *BBox* class, which represents a bounding box by eight corner points stored in an 8 x 3 NumPy [108] array. During development, it became obvious that a convention on the order of the points in the *BBox* object would be beneficial. This convention is presented in Figure 2 in the *Terms and Definitions*.

To ensure a consistent assortment, the bounding box corner points are ordered using Algorithm 19, which is implemented in the *order points* method. More specifically, the points are ordered in a counter-clockwise direction from bottom to top. Additionally, two conditions are enforced. (i) The edge connecting the first and second point must be longer than the one connecting the second and third point, i.e. the edge of the first and second point is along the length of the object. (ii) The first point's X and Y coordinate must be smaller than the second point's X and Y coordinate.

Input : bounding box
Output: bounding box

```
1 Function order points(bounding box):
2     centroid = centroid(points)                          // mean of all corner points
      // point sorting based on angle of each point to centroid vector
3     sorted points = sort(arctan((points Y - centroid Y) / (points X - centroid X)))
      // edge connecting the first and second point should be the longest
4     if norm(point 2 - point 1) ≤ norm(point 3 - point 2) then
5         roll(points)                  // Each point index += 1, last becomes first
6     end
7     return bounding box
```

Algorithm 19: Order points.

Basic methods for transformation include rotation and translation. On a higher level, the *split bounding box* method can be used to split the bounding box in two. This procedure modifies the box it is applied to and returns another one which represents the other half of the split bounding box. The *axis align* method is used to rotate the bounding box along the coordinate frame, whereas the *project into parent* method is applied to modify a bounding box flush to the vertical surfaces of another one (refer to Algorithm 17 in subsection 4.7.3).

To calculate and return basic dimensions of the bounding box, methods including *length*, *width*, *height*, *angle* and *volume* have been implemented. Similarly, other methods calculate and return the endpoints, center plane, and side planes. The *points in bbox* method can be used to retrieve all points inside a bounding box. This method either applies a convex hull or a particle density function to retrieve the points inside the bounding box from an array of N × 3 input points.

The *fit ...* methods are used to fit either an axis-aligned or horizontally aligned minimum bounding box to a set of input points. As the fit methods are the most crucial for scan-to-BIM pipelines, this field is still under development, seeking more effective, robust, and efficient algorithms. However, the hobb fitting algorithm is implemented and used in this work.

visualization: visualize bounding boxes and input data

The visualization class is essentially a wrapper around some of the Open3D visualization functions and classes. The core idea is to instantiate an object of the visualizer class, add geometry to it and invoke the window with the rendered geometries using the *visualize* method. The methods to add geometry to the visualizer object encompass:

- *point cloud geometry* to visualize a set of points provides as a NumPy [108] array.
- *o3d point cloud* to add an Open3D [111] point cloud to the visualizer. Essentially a shorthand of the previous one, as point sets are both stored as NumPy arrays and Open3D point clouds.
- *bbox geometry* to render a bounding box as a set of lines representing the box edges

- *points geometry* to render single points as mesh spheres, which is mostly used to visualize geometry corner points. Not suitable for a large number of points.

- The *clear* method can be used to remove all geometry from the visualizer object.

metrics: evaluating the reconstruction accuracy

In section 6.2 several approaches to calculating the accuracy of the reconstruction will be proposed. Any of these are implemented in the *metrics* module. Applying these and evaluating the accuracy of the reconstruction is part of the pipeline integration chapter 6.

Likewise the introduction of the geometric reconstruction algorithms and their implementation, the procedures, tools and implementation used for open source BIM authoring will be introduced in the next chapter.

5 Open source BIM authoring

5.1 Introduction

To generate BIMs from the previously reconstructed information, there is a wide range of potential software solutions and ever evolving BIM ecosystems. On a high level, these can be distinguished as closed and open BIM environments. The former involve the use of proprietary software and data formats, the latter use open source software, or at least open data standards.

The major advantage of closed BIM is the higher degree of stability in more user-friendly environments and a seamless integration of other applications within the proprietary ecosystem. Although this promise seems to hold up, there are cases when proprietary systems fail to be stable and data-loss-avoiding. The major limitation, however, is the lack of flexibility, as data can only be accessed using licensed software.

Open BIM involves the use of the open data exchange standard IFC. Technically, this allows data to be exchanged among different ecosystems, thus allowing for a higher rate of flexibility in projects. However, the data exchange may be associated with the loss of information, and specifications are needed to ensure consistency. Various quality of exported IFC data can be observed, potentially resulting in information losses. One of the reasons is that the user cannot gain full control of the generation of the IFC data, especially regarding the mapping of software-specific properties and attributes to the respective IFC specifications.

However, there is a promising initiative for an open source BIM ecosystem to overcome the previously described issues. Initiated by Thomas Krijnen, the IfcOpenShell project [23] has become the nucleus of an emerging ecosystem allowing the user to gain full control of the IFC model creation. The development has recently been connected to Blender to deliver BlenderBIM[1] (refer to `https://bonsaibim.org/`, former `https://blenderbim.org/` for details), with Blender serving as a flexible user interface and powerful 3D modelling engine augmented with advanced functions to interact with IFC files.

IfcOpenShell's python API is used in this work to build the tools for IFC model creation. The API allows for convenient access to high-level modelling procedures, while maintaining full control over the process, i.e. whenever the high-level implementation does not yield the expected result, the modelling procedure can be controlled down to the lowest level. Blender BIM as the respective GUI application is used for visual model checking.

Besides the technical advantages, the IFC data format perfectly serves the main use cases of as-is BIMs including delivering models for re-design of existing structures, quantity take-offs, circularity

[1] Recently, BlenderBIM rebranded to Bonsai. As it is still common, the term 'BlenderBIM' will be used in this work.

assessments, life-cycle-assessments, etc. Both in the early stages of projects as well as during the operation phase of buildings, open data formats allow multiple consultants and service providers to access the data, while proprietary closed source environments would limit the usability.

One key step is to transfer the reconstructed geometry into IFC geometry representations to provide a fully functional BIM model. Note that, technically, even point clouds or mesh representations could be used in IFC. Both are heavy and inefficient, thus high-level parametric representations are used. Beyond creating geometric representations, the assignment of material and property sets as well as the correct spatial assignment of the reconstructed object will be discussed.

5.2 From bounding box to BIM geometry

BIM representations comprise both geometric and semantic information. For the former, the previously reconstructed geometry (see chapter 4) needs to be transferred into BIM geometric representations which are derived from the geometric reconstruction, e.g. bounding boxes.

The purpose of this work is not to discuss the overall structure of the IFC standard. However, the basic concept of geometry is vital for further considerations. An IFC element, if the standard allows it to incorporate geometry, will be assigned an IfcRepresentation, which is assigned an IfcShapeRepresentation. Among basic geometries such as points or curves, an IfcShapeRepresentation can include parametric solid geometries such as Swept solid, Boundary representation and Constructive solid geometry [3]. All mentioned concepts have particular advantages. However, for cuboid-like geometry as reconstructed previously, the IfcSweptSolid is the primary choice. The basic principle is to use a profile and extrude it along a curve. Thus, to create a box as an IfcSweptSolid, a rectangle is used as a profile, which is then extruded along the normal of the rectangle with a given dimension.

Technically, such parametric IFC geometry could be created using global coordinates, i.e. the corner points of the bounding boxes previously reconstructed. However, IFC models follow the concept of local placements. A local placement is a local coordinate frame defined by its axes as vectors and the coordinates of the origin. The concept of local placement applies to the overall structure of the IFC model, including IfcProject, IfcSite, IfcBuilding and finally, the building components. According to the standard, all IfcElements (IfcWall, IfcColumn, ...) should be placed relative to the local placement of its container, e.g. IfcBuilding [3].

The same principle applies to child elements such as openings that are associated with a wall, as illustrated in Figure 5.1. The origin of the IfcLocalPlacement is defined with respect to the parent IfcLocalPlacement as defined in the IFC documentation. Mainly, the length of the object should expand in the x direction of the IfcLocalPlacement, and the main edges of a cuboid-style geometry should be aligned with the coordinate frame axes, with Z as the upright. The origin of the IfcLocalPlacement should be set at the 'lowest-left corner' (see Figure 5.1) depending on the rotation. The length of the object, however, should expand in x-direction.

The described concept requires the global bounding box coordinates to be transformed into a local coordinate frame, or a local coordinate frame relative to the parent element, respectively.

As the points in the bounding box are presented in a specific order, the IfcLocalPlacement origin can be

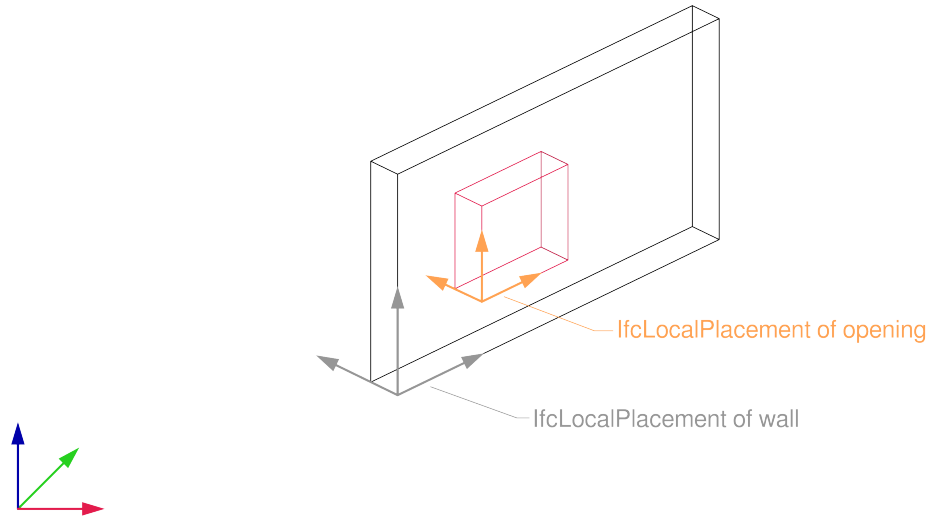


Figure 5.1: IfcLocalPlacement of IfcWall and IfcOpeningElement [3].

set to the first bounding box point, the X axis is the norm of the vector connecting the first and second point, the Y axis can be set as the vector connecting the first and fourth point, the Z-axis is set vertical. To ensure that all axes are perpendicular, only the origin and rotation are passed to IfcOpenShell's [23] util.placement.rotation function.

For the IfcLocalPlacement of any opening, a copy of the wall local placement that has been transformed to the opening position can be used. As the door has been projected into the wall bounding box, the transformation vector is the one connecting the first points of the wall and door bounding boxes. The points of both bounding boxes need to be ordered before using Algorithm 19.

The described concept and procedure can be used for any object with cuboid geometry (walls, slabs, square columns, ...) and respective child elements (doors, shafts, openings, ...). The IfcLocalPlacement of round objects, e.g. round columns is placed at the centre of the lower circle and the orientation is aligned with the global X and Y axes, the Z axis with the vertical column centre axis.

To demonstrate the behaviour, Figure 5.2 shows wall and door bounding boxes with the IFC geometry. The blue points represent the origin of the IfcLocalPlacement, which aligns with the first corner point of the bounding boxes. In the lower part of the figure, the resulting geometry solids are displayed. Note that some elements have been hidden to allow for a better view.

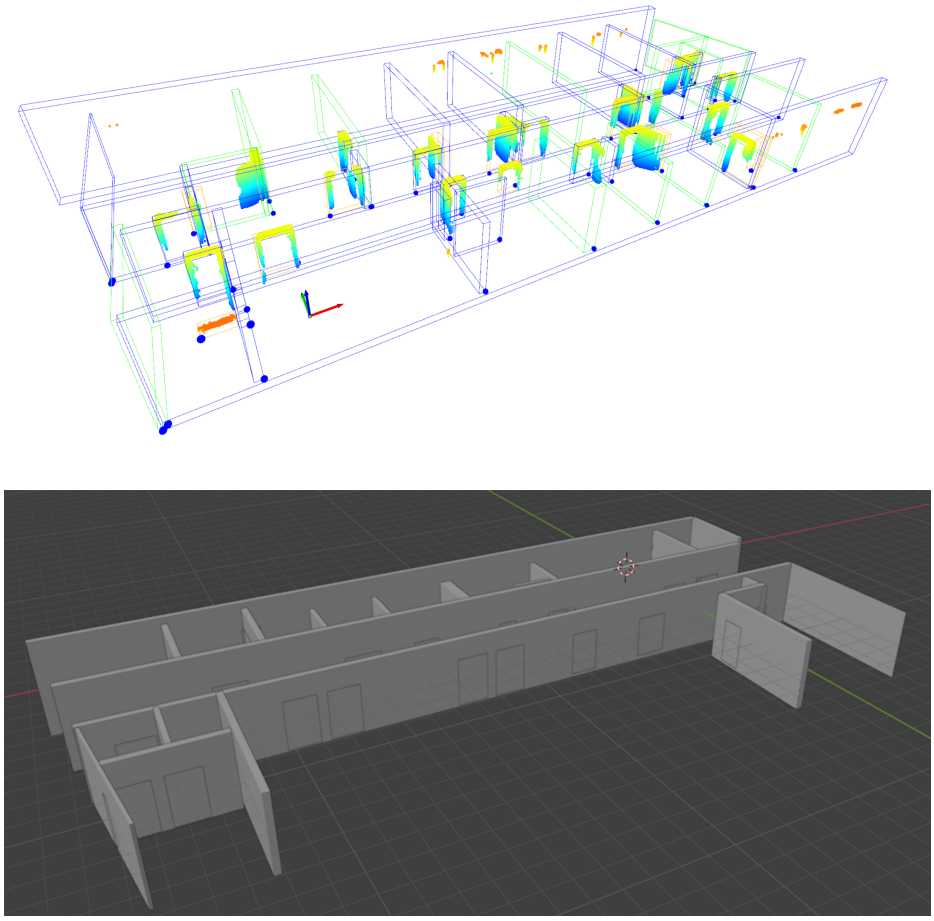


Figure 5.2: Wall and door IFC elements. Top: door geometry with door points. Points mark the origin of the IFC local placement. Bottom: Resulting IFC objects.

5.3 Semantic IFC object creation

In the context of this work, three types of IFC concepts for the enrichment of models with semantic information apply: IfcMaterial creation and assignment, property sets and spatial relationships. Note that the scope of this work is limited to visual features and mainly the geometric reconstruction. Thus, any information on material and other properties are either captured visually or set as generic values to demonstrate the assignment of both material information and other properties.

Material information

In the course of the IFC model creation, the IfcMaterial definitions can be set up as one of the first steps, as these can be used for all objects created. Besides the plain IfcMaterials creation, other concepts such as IfcMaterialLayerSets apply for elements such as walls. The IfcMaterialLayerSet entity consists of a set of layers with specified thickness and assigned material. Although an IfcMaterialLayerSet would be the right choice for elements such as walls, in this work we only assign an IfcMaterial to the elements reconstructed, as the layer composition is unknown.

Property sets

Beyond material assignment, the concept of property sets is relevant to most building objects in the IFC schema. These property sets act as templates for assigning semantic information to specific elements. Specifically, the standard property set for walls, denoted as *PSet_WallCommon*, includes properties like *Reference, AcousticRating, FireRating, Combustible, SurfaceSpreadOfFlame, ThermalTransmittance, IsExternal, ExtendToStructure, LoadBearing, Compartmentation and Status*[2]. Even without detailed explanations, it is evident that these properties are intended for the standard occurrences of walls, aligning with the primary objective of property sets. In addition to the standard ones, specific property sets can be created to meet the particular requirements of projects and organisations.

Spatial relationships

In IFC, objects are assigned to spatial containers. In buildings, storeys and levels are the common type of spatial containers. The level elevations are one of the first information inferred in the proposed pipeline, and can be used to set the elevation of *IfcBuildingStorey* objects. Any object of the storey is then assigned to the respective spatial container using an *IfcRelContainedInSpatialStructure* object. Note that the information on the spatial relationship of an object is not contained within the object itself, but in a separate class object.

[2] Names are written as in the IFC schema, ignoring correct spelling

5.4 *Openbimxd*: open source BIM authoring

All functionalities described above have been implemented in the *openbimxd* python package, which is available at `https://github.com/humantecheu/openbimxd`. *Openbimxd* mostly relies on IfcOpenShell [23], which offers a comprehensive library to create and interact with IFC data.

Openbimxd comprises five modules: (i) *ifcfile* to set up an IFC file and its basic structure, (ii) *elements*: to create IfcWall, IfcDoor and IfcColumn from bounding boxes, (iii) *ifcmaterial* to define a set of IFC materials to be assigned to objects, (iv) *filtering* to find and extract single objects or groups of objects of a common class or with common properties, (v) *ifctolabel* to convert the geometry representation of IFC objects into bounding boxes and assign the IfcClass to all points inside the respective bounding box and (vi) *ifcupdate* to update attributes, properties and geometry of IFC objects.

ifcfile: basic definitions and file structure

This module contains the class *IfcModelBuilder*, which creates and populates an IFC file object. The IFC file object is passed to every other process to either extract data or create new objects in it. Beyond the constructor, it contains the *write* method that saves the changes made to a file with a given path and filename.

elements: creating semantic objects

The *elements* module is used to create representations of different types of building objects. At the current state, three sub modules are available: *ifccolumn, ifcdoor* and *ifcwall*. All three sub modules contain a *create* method that creates the respective object into a given *IfcModelBuilder* object. The input to the *create* method is a bounding box that has previously been reconstructed. For creating columns, the shape needs to be given as a parameter along with the bounding box or the parameters of a cylinder for round shapes. Similar to the procedures for geometric feature extraction (refer to chapter 4), the three methods can be adopted to other types of objects. Based on the *ifcdoor* module, a procedure to create *IfcWindow* objects could be implemented trivially, the same applies for other objects.

ifcmaterial: create material representations

At this stage, the module contains the *IfcMaterials* dummy class, which creates two materials. As the material information cannot be retrieved from the previous reconstruction process, the purpose is to demonstrate how material information can be added to the IFC model. The concept of the class is to provide a comprehensive set of materials to be used by multiple objects, as a particular material is present once in the IFC and the material is assigned to the objects.

filtering: extracting objects from IFC models

This module contains the *objectFilter* class. The main method for filtering is the *filter_objects* method, which is a wrapper of the IfcOpenShell *util.selector.filter_elements* method and accepts search strings to apply one or several criteria to retrieve the filtered objects. Other methods are implemented to reassign materials, spatial containers, openings and property sets back to the filtered objects to avoid data loss. Finally, the *export_model* method writes the filtered objects to a newly created file. A common application of this procedure is to filter a subset of a large input model to serve a specific use case, e.g. semantic labels assignment.

ifctolabel: assign semantic labels from IFC to a point cloud

Given a BIM model and a point cloud aligned in a global or local coordinate frame, the *filtering* module can be used to retrieve all elements of a specific class. The *ifctolabel* module in turn can be utilized to assign semantic labels from the previously filtered objects to a point cloud. The *IfcToLabel* class includes a method to retrieve the inliers inside a bounding box. The point cloud labels can be edited using the *edit_labels* method. Through the API, high level functions can be called to parse the labels of IfcDoor, IfcWindow, IfcSlab and IfcWall to the point cloud. Finally, the *visualize* method visualizes the result of the label assignment. Both retrieving bounding box inliers and visualization are functions of the *pystruct3d* module, which is a dependency of this module.

This module is not relevant to scan-to-BIM, but relevant to this work as it is useful to partially automate the annotation of training data for semantic segmentation if a BIM model is available.

ifcupdate: updating IFC objects

As with the previous module, the *ifcupdate* module is not directly used in the proposed scan-to-BIM pipeline. However, it can be useful in a slightly different case when an already existing BIM model should be updated based on reconstructed information. The *UpdateIfcObject* class encompasses methods to update the *location, material and property* of an IFC object along with a *write* method to save the updated objects into a new file.

With the development and implementation of methods for semantic segmentation, geometric reconstruction and open BIM authoring, the foundation is now set to establish a comprehensive scan-to-BIM pipeline.

6 Pipeline integration and results

6.1 ScaleBIM: a comprehensive scan-to-BIM pipeline

In the previous sections, a wide range of algorithms and methods for the topologically correct reconstruction and BIM object creation have been discussed. These methods have been integrated into a comprehensive, automated pipeline that converts the input point cloud into a BIM model. This pipeline comprises all steps necessary and, beyond the mere algorithms that have been tested individually, all the methods are integrated. Although this seems trivial, a considerable research effort had to be spent to find the best performing processing sequence and to eliminate algorithm side effects that can influence subsequent procedures. On a high level, the pipeline is illustrated in the pseudocode provided in Algorithm 20. Note that the granularity has been chosen on purpose to give a coarse overview of the overall pipeline. All operations needed for intermediate pre- and post-processing to connect the algorithms, provide interfaces, perform checks and error handling are not included. The pipeline is implemented in Python in an object-oriented manner, which is not reflected by the pseudocode either. The project comprises almost 9,000 lines of code in the pipeline itself and in the *pystruct3d* and *openbimxd* modules.

The pipeline takes both the input point cloud and an array of labels inferred from the segmentation model as an input. The behaviour of the algorithms is controlled by a set of parameters managed through a configuration file. These parameters control both the global behaviour and the specific application of the algorithms. An explanation of the parameters can be found in Annex 7.

The pipeline is structured into three primary processing stages, specifically designed to reconstruct walls, doors and columns in that sequence. Initially, preprocessing procedures are implemented to filter out clutter (see Algorithm 4) and to reconstruct the levels (see Algorithm 6). If the building's exterior is only sparsely covered, a specialized method for reconstructing the exterior walls can be utilized (see Algorithm 10).

As parent objects for elements such as doors, walls are the first to be reconstructed. The detailed procedure is outlined in section 4.6; here, a brief overview is provided to describe the sequence used. Initially, points categorized as walls are sorted along either the X or Y axis using Algorithm 7. Next, DBSCAN clustering (see Algorithm 2) is applied. Planes are then fitted using the HYSAC algorithm (Algorithm 8) and subsequently grouped by wall instance. Initial bounding boxes are generated using Algorithm 9, followed by topology correction: Algorithm 11 corrects wall corners and Algorithm 12 merges walls along a common direction. The latter two algorithms are applied several times, as each iteration might produce new wall instances that could be candidates to be merged or closed. The result is a list of wall bounding boxes for use in subsequent procedures, with IfcWall objects created for each bounding box. Similarly, exterior bounding boxes will be reconstructed and IfcWall objects will be created for exterior walls reconstructed separately.

```
   Input  : input point cloud
   Input  : label array
 1 Function preprocess(input point cloud):
 2     filtered point cloud = density based clutter removal(input point cloud)
 3     levels = level finding(input point cloud)
 4 Function process walls(input point cloud, label array):
 5     wall points = filter labels(input point cloud, label array, wall ID)
 6     for X, Y direction do
 7         axis sorted = axis sorting(wall points, direction)
 8         clusters = dbscan(axis sorted)
 9         for cluster in clusters do
10             grouped points = hysac(cluster)
11             if any(grouped points) then
12                 bounding box = hobb fitting(grouped points)
13                 append(wall list, bounding box)
14             end
15         end
16     end
17     return wall list                                    // list of bounding boxes
18     merge bounding box(wall list)
19     close bounding box(wall list)
20     ...                                                   // Apply multiple times
21     outside walls = outside walls(filtered point cloud)
22     append(wall list, outside walls)
23     for bounding box in wall list do
24         create IfcWall(IFC model, bounding box)        // add IfcWall to the model
25     end
26 Function process doors:
27     for parent bbox in wall list do
28         door points = filter labels(input point cloud, label array, door)
29         opening points = find opening(parent bbox, door points)
30         bounding box = door reconstruction(opening points)
31         append(door list, bounding box)
32     end
33     return door list for bounding box in door list do
34         create IfcDoor(IFC model, bounding box)        // add IfcDoor to the model
35     end
36 Function process columns:
37     column points = filter labels(input point cloud, label array, column)
38     clusters = dbscan(column points)
39     for cluster in clusters do
40         bounding box = hobb fitting(clusters)
41         append(column list, bounding box)
42     end
43     for bounding box in column list do
44         create IfcColumn(IFC model, bounding box)    // add IfcColumn to the model
45     end
46 return IFC model
```

Algorithm 20: scan-to-BIM pipeline.

Each of the reconstructed wall bounding boxes are then used to apply Algorithm 15 to identify door openings and to reconstruct these using Algorithm 14, resulting in both a list of door bounding boxes and IfcDoor objects.

Columns are reconstructed by applying DBSCAN clustering (see Algorithm 2) to identify column instances in the overall set of column points. For each cluster, a bounding box is fitted using Algorithm 9. To enhance the reconstruction results, additional measures such as clipping the clusters along the Z-axis can be implemented to avoid over segmented points protruding at the top and bottom of columns.

Finally, all IFC objects are stored in the IFC output file. Beyond simple reconstruction, an evaluation can be initiated to assess the quality of the generated output. For this purpose, various evaluation metrics are employed.

6.2 Evaluation Metrics

An appropriate solution to quantitatively evaluate the reconstruction's quality is to use previously created ground truth BIMs. These could already exist or can be created from various sources. In this work, both the 14NH dataset, which is a contribution of this work, as well as the publicly available CV4AEC dataset are used as a benchmark to evaluate the reconstruction quality. Before presenting the results, it is helpful to discuss various sources of ground truth BIMs.

In projects incorporating BIM-based design and execution, various BIMs with differing levels of detail and content serve as benchmarks for the reconstructed BIMs. These models may represent the design stage and can evolve into as-built models, enriched with semantic and/or geometric information about the finished product. If a geometrically detailed as-built model exists, it functions as a benchmark. Commonly, as-built models are updated predominantly in semantic aspects, e.g. adding data on production, manufacturer, product types, environmental conditions and material properties. These models geometrically depict the project's final design stage. It is typical for the execution to deviate somewhat from the design stage. For existing buildings, it also cannot be ensured that the as-built models contain all changes during the operations phase such as alteration of MEP components, room layout, etc. However, such models are not suitable benchmarks for evaluating scan-to-BIM output, as such an evaluation would include deviations between the design and actual construction, rather than solely the discrepancies between the as-is building scan and the reconstructed BIM model derived from the point cloud.

If the as-built model does not comply with the as-is state of the building or is not available at all since the project did not involve BIM, other ways have to be explored to deliver benchmark BIMs. Thus, the following procedure will be employed in this work. The starting point for both the automated generation and creation of the benchmark model is the point cloud captured from the object. From this point cloud, the benchmark BIM model is created manually using Autodesk Revit as a modelling tool and Autodesk Recap to visualize the point cloud inside Revit. This process involves civil engineering knowledge to provide a topologically correct BIM model. To this end, the structural system is analysed and correctly modelled, which involves aligning the centre axes of vertical structural components, aligning horizontal objects on the same level, etc. Inherently, the manually created model is a simplified representation of the point cloud. Typical simplifications include applying the Manhattan-world assumption, i.e. objects are axis-aligned; slabs, floor, ceiling are aligned horizontally; walls and columns are aligned vertically;

opening objects are aligned with the parent objects. The degree of simplification can be distinguished by calculating the cloud-to-BIM distance, with a larger distance meaning a higher degree of simplification.

The primary method to assess the reconstruction accuracy is to compare the reconstructed and the reference BIM model. To this end, bounding boxes are derived from any geometry representation. Obviously, cuboid representations could be used directly, but walls and doors could hold more complex and detailed representations potentially hard to compare. To quantify the deviation of the obtained benchmark and reconstructed bounding boxes, different methods exist. In this work, the IoU is considered, which has some limitations. Thus, the Volumetric Intersection over Union (vIoU) was developed and implemented, which is calculated on a voxel representation of the bounding boxes and overcomes some limitations of the conventional IoU. As supportive metrics, the centroid deviation is used. Finally, the point-to-BIM distance can be applied to describe the deviation of the scanned data and reconstructed BIM model explicitly.

Intersection over Union

As the name states, the IoU is calculated by dividing the intersection of the reconstructed geometry and ground truth by the union of both. The ratio quantifies the overlap of the reference and the reconstructed geometry in reference to the combined volume of both. Thus, this metric considers both the size of the reconstructed and the reference geometry.

$$IoU = \frac{R \cap GT}{R \cup GT} \tag{6.1}$$

Where R denotes the reconstructed geometry and GT denotes the ground truth geometry.

The IoU can be applied on points to evaluate the semantic segmentation of point clouds, in 2D on areas and in 3D on volumes. In the context of this work, the 3D IoU is the relevant option. More specifically, the IoU calculation is limited to bounding boxes, and the volume will be used to calculate the IoU. To this end, it is crucial to find the intersecting volume of the reconstructed and ground truth bounding box.

Before calculating the intersection volume, it is necessary to first identify the two bounding boxes with the maximum overlap. One approach is to algorithmically determine the most overlapping bounding boxes. In the CV4AEC challenge evaluation procedure, the Hungarian Algorithm [115] is employed to find the bounding boxes with the closest endpoints [1]. The cost is calculated as the distance between the respective endpoints, which subsequently generates a confusion matrix. The bounding box with the maximum overlap is then identified by matching the endpoints at the lowest cost, i.e. the smallest distance. However, this method has certain limitations. First, solving the problem in 2D by projecting all points onto the XY base plane can introduce significant errors. Second, the closest endpoints may not necessarily correspond to the most overlapping bounding boxes. To address these issues, we adopt a

[1] There is no paper to refer to. Thus, the open source evaluation code base (see `https://github.com/GradientSpaces/cv4aec-challenge`) was used to examine the IoU calculation.

brute-force approach, calculating the IoU for all possible combinations of bounding boxes and selecting the one with the highest score.

What remains, is to calculate the intersection volume. This problem can be significantly simplified by assuming that all boxes are axis-aligned. Under this assumption, only the vertices of both boxes need to be compared, and the intersection points can be easily calculated by subtracting the corresponding coordinate components. However, this approach does not apply to the data and implementations previously described. Therefore, a more general method, based on the following assumptions, needs to be implemented:

- Vertical upright edges, i.e. no rotation around the X or Y axes.
- Horizontal lower and upper faces.
- Arbitrary rotation around the vertical Z axis and arbitrary translation in X, Y and Z direction.

Based on these assumptions, the IoU calculation can be simplified to a 2.5D problem. The implementation follows the method used in PyTorch3D [116]. For each edge of the first bounding box, its intersection with the other box is checked. If an edge does not intersect, it is discarded; otherwise, the intersection point is computed. The collection of all intersections and edges forms a convex n-gon, representing the intersection solid. The volume of this intersection solid is then determined. To find the volume of the union of both bounding boxes, the volumes of the individual bounding boxes are summed, and the intersection volume is subtracted to avoid double-counting. The IoU is subsequently obtained by dividing the intersection volume by the union volume of the two bounding boxes:

$$IoU = \frac{intersection\ volume}{volume\ box\ 1 + volume\ box\ 2 - intersection\ volume}$$

Calculating the volumes of the individual bounding boxes is straightforward. However, directly calculating the volume of the complex union of two bounding boxes is avoided. Instead, the focus is on determining the intersection volume, which is more challenging due to the potential polygonal shape of the intersection solid. This solid has vertical edges and horizontal upper and lower faces, allowing its volume to be calculated by multiplying the area of the polygonal base by the height of the intersection solid. To determine the 2D area of the polygon, Gauss's area formula is applied, as it is effective for polygons of arbitrary shapes.

Although being a commonly used metric for the evaluation of reconstructed geometry, the IoU does not develop linearly, as can be observed in the examples in Figure 6.1. The left-most configuration of bounding boxes results in an IoU of zero, as the boxes do not intersect. Towards the right, the IoU increases along with the overlap of the bounding boxes up to 0.62 representing the best achievable overlap of both boxes with slightly different dimensions. Only a full overlap of two boxes with equal dimensions results in an IoU of 1.0, with a remarkable difference of 0.38 to the previous configuration. Note that adding the third dimension will increase the effect described.

When calculating the IoU, the first step is to identify the two bounding boxes with the most overlap and then compute the IoU. However, this process can be challenging, particularly when one object is

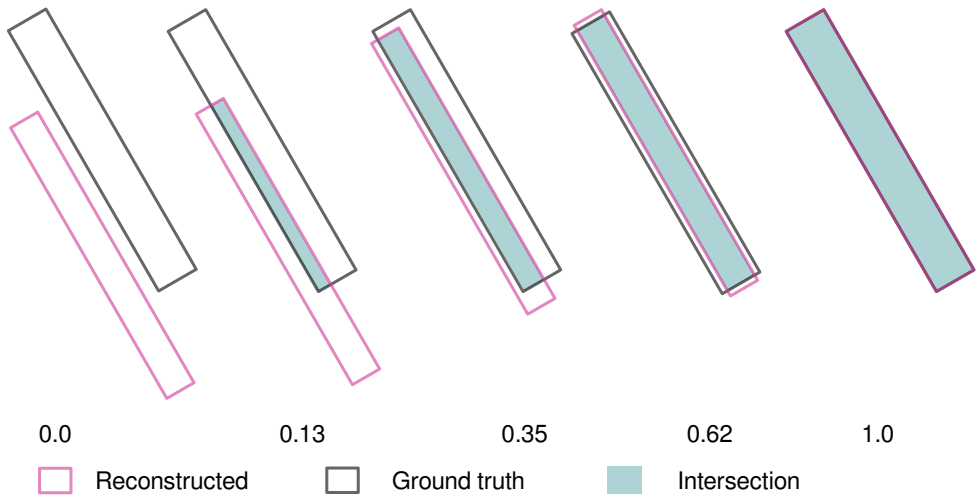


Figure 6.1: 2D quantitative calculation of the IoU metric on different rectangle configurations. Except for the right configuration, reconstructed and ground truth do not share the same dimension. The IoU values given are calculated in 2D based on the areas.

mistakenly reconstructed as two instances of similar size. In such cases, only one bounding box — half the size of the ground truth — would be used to calculate the IoU, leading to a poor IoU score. Although obtaining two instances for a single ground truth object, or vice versa, is not inherently problematic, it is penalized when the IoU is used as the evaluation metric.

This underpins the observation that the IoU metric experiences a rapid and significant decline even with minor misalignments between two boxes. This phenomenon is attributed to the sensitivity of IoU to variations in size, position, or orientation, as any such discrepancy leads to a decrease in the IoU value. Notably, the IoU penalizes even slight positional shifts between two identically sized boxes. This aspect highlights another limitation of IoU: its lack of descriptiveness. Whether the reduction in IoU is due to changes in translation, rotation, or dimension, the metric drops similarly, making it challenging to ascertain the specific cause of the reduced IoU from the metric alone. This is because the IoU fundamentally relies on the extent of overlap between two bounding boxes, and any alteration in translation, rotation, or dimension impacts this overlap. Consequently, there is an acknowledged necessity to develop and adopt more informative metrics that can more accurately identify the reasons behind poor results.

Volumetric Intersection over Union

As the name indicates, the vIoU is calculated based on a voxel grid. Thus, the space encompassing both the ground truth and reconstructed bounding boxes is discretized into a set of voxels. Any voxel intersecting with or being part of a bounding box is then classified as populated by determining whether the centroid of the voxel is inside the bounding box. This method returns a set of populated voxels. As the voxel grid is a structured spatial representation, the indices of all populated voxels for both the ground truth and reconstructed bounding boxes are sufficient to obtain the vIoU, which is calculated by dividing the intersection by the union of both sets of indices. The principle is illustrated in Figure 6.2.

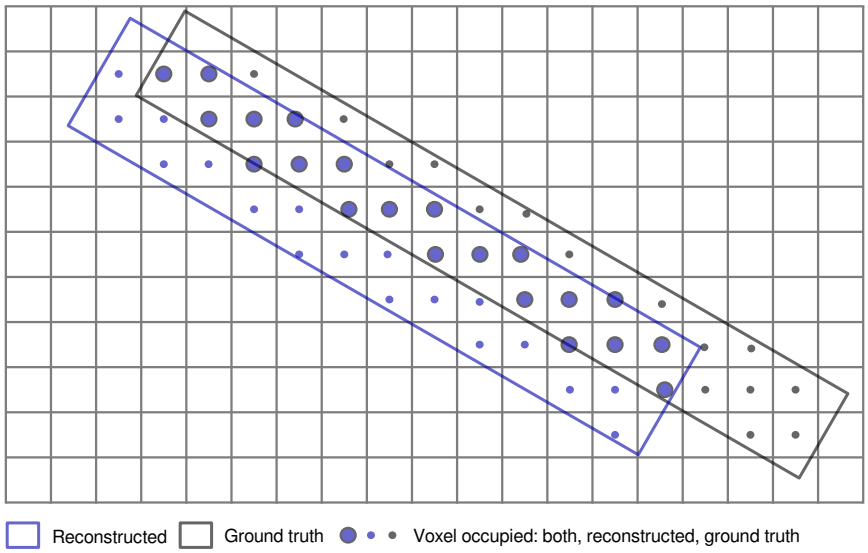


Figure 6.2: Volumetric IoU calculation. For each voxel it is determined whether it is occupied by a reconstructed or ground truth bounding box based on the centroid.

This approach yields certain benefits compared to the conventional IoU. First, the conventional IoU requires complex calculations to match the best overlapping bounding boxes and to then calculate the overlap of both bounding boxes. Secondly, the vIoU behaves more linearly in the case of small deviations than the conventional IoU. Qualitatively, it behaves more naturally as the results do not drop as quickly as with the conventional IoU. This is related to the fact that small deviations (depending on the voxel size) are ignored as we consider the voxel centre. Such small deviations (e.g., 2 cm) do not have any relevance in practice. Thirdly, the vIoU avoids a double consideration of boxes, as the voxel grid asserts a one-to-one matching and calculation without uncertainty of matching algorithms.

Beyond that, the conventional IoU is applied on exactly one ground truth and reconstructed bounding box. These could be matched using different strategies such as applying the Hungarian Algorithm on the bounding box centroids. During the reconstruction it is likely that one wall instance is reconstructed as two or vice versa. From a practical point of view, this does not affect the quality of the reconstructed

model. Even in manually created BIMs both cases occur, and no strict topological rules exist, as the split of instances can be motivated by several reasons such as different material of adjacent walls, construction sequencing based on sections of the building, etc. The vIoU overcomes this issue as the sets of populated voxels are compared, which is agnostic of the number of bounding boxes in each set. Indeed, this results in a practically more relevant metric. Moreover, any occurrence of one object being reconstructed as one will not be penalized by the vIoU, thus yielding a more realistic score.

Geometric deviations

Although the vIoU is a practically more relevant metric, it still does not describe the geometric deviation well. Especially during the development stage of scan-to-BIM methods, a more specific description of the deviations can be beneficial to identify methodological problems. In the following, a list of such deviation metrics is presented:

- **Centroid deviation**: The centroid deviation is the length of the vector connecting the centroids of both bounding boxes and describes the translation of both bounding boxes. Note that the calculated translation is most explicit for boxes with nearly the same dimensions. In other cases, a portion of the deviations might be related to the differences in dimension. The centroid deviation is associated with a length unit and can thus not be combined with IoU directly.

- **Angle deviation**: The angle deviation can be calculated by subtracting the angle of the ground truth bounding box from the angle of the reconstructed bounding box. Hereby, the angle is defined as one of the longest horizontal edges to the x-axis, where counter-clockwise is positive. By this metric, the rotation can be evaluated. Note that this metric is in degrees, thus it cannot be directly combined with the IoU metrics.

- **Dimension accuracy**: As the bounding box implementation developed and used in this work provides methods to calculate the length, width and height of a bounding box, it is straightforward to calculate the dimensional deviation. Subtracting the reconstructed by the ground truth dimension and dividing by the ground truth dimension, we obtain the accuracy in the respective dimension.

- **Volume accuracy**: Likewise, the volumes of both the ground truth and reconstructed bounding box can be calculated.

Typically, bounding boxes deviation is calculated per object class separately. The first step in the procedure is to identify the adjacent bounding boxes. This is achieved by calculating all bounding box centroids then finding the nearest neighbour bounding box in the benchmark set for the respective reconstructed bounding box. A k Nearest Neighbour (kNN) search is performed using the SciPy implementation [114] to identify the closest bounding box ($k = 1$). For each pair of bounding boxes, the respective accuracy measure can then be calculated.

Note that in this work primarily the centroid deviation is used to evaluate the translation. This supplements the IoU or vIoU metric. Other geometric deviations such as the dimension accuracy are implicitly part of IoU and vIoU but could be used additionally for further examination of the results.

Point-to-BIM distance

Any metrics previously described rely on a ground truth BIM model. As discussed, such manually created models can be biased due to varying experience of the modellers and subjective interpretation of incomplete data as well as simplifications based on the expertise of the modeller. Thus, it is considered helpful to quantify the accuracy only based on the input data and the reconstructed BIM model. This can be achieved by directly comparing the reconstructed BIM model with the point cloud from which it was reconstructed. Technically, the Euclidean distance of each point of the point cloud to the BIM geometry objects is calculated. The mean of all distances can then be used as a comprehensive metric for the reconstruction accuracy.

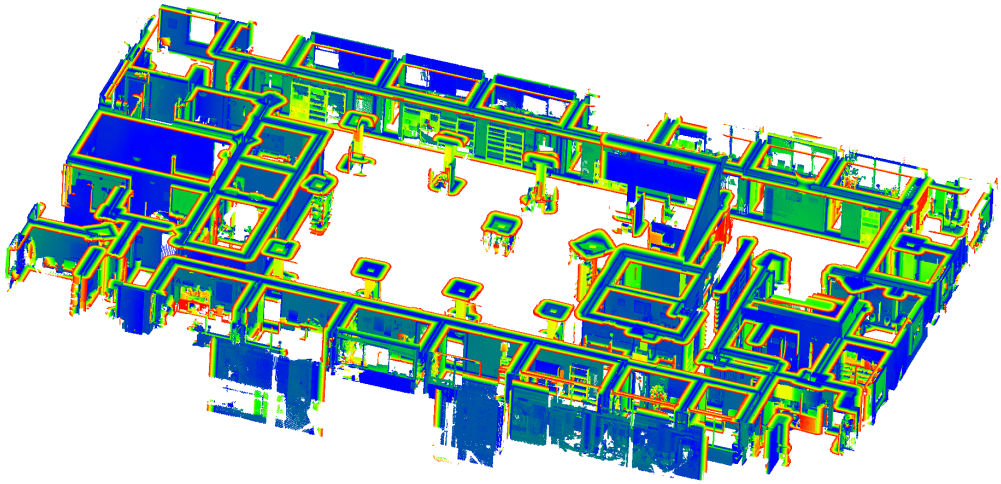


Figure 6.3: Cloud-to-BIM distance on the CV4AEC 08_ShortOffice_01_F1 scene. The distance was limited to 40 cm for calculation. The mean deviation is approx. 11 cm. The colour map represents the distances from 0 cm (blue) to 40 cm (red).

To calculate the distance, two options can be identified: (i) A point cloud with semantic annotations can be used, thus calculating the distance per class. This excludes any points not related to the reconstructed objects. However, annotation errors can affect the distance calculation. (ii) The original point cloud can be used, but any distances beyond a distance threshold can be excluded. Some experiments prove that even with a high distance threshold of 40 cm, only a small portion of non-related points are incorporated in the calculation, resulting in reasonable distances (see Figure 6.3). On the one hand, severe misalignments could be excluded from the calculation. On the other hand, points far from the object are likely related to other objects potentially not reconstructed at all, and can therefore be excluded. Hence, using the original cloud is the preferable option, as it guarantees that no human-related errors are incorporated.

The evaluation metrics introduced earlier will be applied in the following experiments. By comparing these metrics and analyzing their behaviour in various scenarios, a deeper understanding of their specific limitations can be achieved.

6.3 Experiments

Experiments were performed on both the 14 Nothelfer Hospital, Weingarten, Germany (14NH) and on the Computer Vision in the Built Environment (CV4AEC) dataset. Both datasets were chosen as they provide distinct characteristics. While the first barely contains interior objects, the latter is littered with interior objects leading to occlusions. Hence, the performance on occluded scenes can be assessed as well. The experiments will clearly reveal the potential of the proposed framework. Nevertheless, the limitations will be examined and demonstrated as well.

Beyond the mere results, an ablation study examining the advances of the proposed reconstruction pipeline and assessing the effect of the semantic segmentation accuracy will be presented. One of the scenes of the 14NH dataset is used for the ablation study.

14 Nothelfer hospital, Weingarten, Germany

The 14NH dataset comprises four scenes from three different buildings. The data was captured as part of the HumanTech `https://humantech-horizon.eu/` research project using a Leica BLK 360 1st generation laser scanner on 10th and 11th October 2022 in an unused hospital building in Weingarten, Germany. The first two scenes were captured in Building A, which features office rooms on the first floor and patient rooms on the second floor. This building, approximately 100 years old, has a classic layout with structural walls, a central hallway, and small terraces on the second floor. Both floors in Building A were largely unfurnished. Building B includes an area from a former Intensive Care Unit (ICU), where the captured rooms were mostly empty except for integrated furniture and clinical equipment, such as terminals with power and gas outlets mounted on movable arms attached to the ceiling. Compared to Building A, the doors and hallways in Building B are wider, meeting the requirements for an ICU section in a hospital. Finally, scenes from surgery rooms in Building C were captured. Similar to the ICU in Building B, the surgery rooms were empty except for gas and power outlets on movable arms and integrated furniture. The doors and hallways in Building C are similarly wide. For Building A, an aligned scan of the building's exterior is available and was used for processing; for the other buildings, only the inner surface of the exterior walls is included in the dataset.

All four scenes were reconstructed using the pipeline as specified in section 6.1. The results are presented in Table 6.1. No columns were present in the A first floor and C surgery scene, thus '-' was reported for the column metrics. A rendering of the reconstructed wall, door and column bounding boxes can be found in Annex 8.

In the second-floor scene of Building A, a notable decrease in both the walls' and doors' IoU and vIoU compared to the first floor is observed. The primary issue stems from the wall configuration, which forms small compartments, likely used as lavatories, on the backside of the hallway walls. A visual comparison indicates that the pipeline struggles to accurately reconstruct these small wall segments (see Figure 6.4). This difficulty is linked to the wall merging and closing procedures (refer to Algorithm 11 and Algorithm 12), which tend to merge these small segments into larger wall instances. Beyond that, the difference in wall IoU and vIoU of 0.244 suggests that numerous small misalignment errors are occurring. Given the vIoU voxel size of 5 cm, any errors within this range will negatively impact the IoU but not the vIoU. As previously discussed, errors within the 5 cm range are practically insignificant for

Metrics	A 1st floor	A 2nd floor	B ICU	C surgery
Wall centroid deviation [m]	0.334	0.309	0.183	0.156
Wall bounding box IoU [-]	0.395	0.219	0.270	0.328
Wall vIoU 5 cm [-]	0.553	0.463	0.388	0.451
Door centroid deviation [m]	0.158	0.355	0.214	0.184
Door bounding box IoU [-]	0.417	0.104	0.193	0.223
Door vIoU 5 cm [-]	0.402	0.103	0.219	0.236
Column centroid deviation [m]	-	0.191	0.200	-
Column bounding box IoU [-]	-	0.037	0.095	-
Column vIoU 5 cm [-]	-	0.007	0.097	-

Table 6.1: Results of the evaluation on the 14NH dataset.

delivering an architectural BIM model for general applications as architectural design or quantity take-offs. Other use cases, e.g. prefabrication, might require a higher accuracy, but would in turn require different reconstruction and evaluation approaches.

Given the poor wall reconstruction, the door reconstruction is similarly unsuccessful. The sparse door segmentation further exacerbates the quality of the door reconstruction. While there is a difference between the wall IoU and vIoU, the door IoU and vIoU are both low and nearly identical, indicating that the door reconstruction is not only misaligned but also significantly flawed.

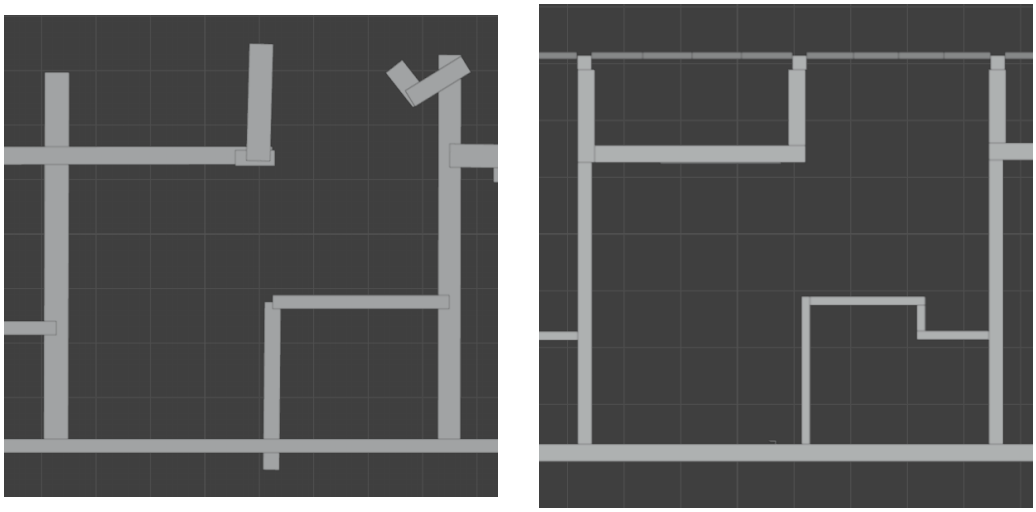


Figure 6.4: 14NH A building second floor. Left: inaccurate reconstruction of lavatories enclosed by small wall instances. Right: Reference BIM model. Only one room of the entire scene is shown.

The reconstruction of the B ICU scene is more accurate, though not as precise as that of the A first floor scene. The differences between the IoU and vIoU once again point to small misalignment errors in the reconstructed objects, as well as the overall absence of certain objects in the reconstructed scene. Although the door reconstruction is generally accurate, aside from some missing instances (see Annex 8), the door IoU suggests lower accuracy. This discrepancy arises because the reference BIM model's geometric representation of doors consists only of a thin door leaf, rather than a more comprehensive

representation that includes the door frame and lining. Consequently, the IoU appears poor, despite the accurate reconstruction.

As previously mentioned, the data includes integrated furniture. As a result, some walls in the C surgery scene are inaccurately reconstructed because the surfaces of integrated cabinets were segmented as walls and subsequently reconstructed as such. However, other walls are accurately reconstructed, though some are split into multiple instances, despite the reference BIM containing only a single wall instance. Another issue observed in this scene is the double reconstruction of doors, occurring both in their parent wall and in the perpendicular adjacent wall, particularly when the door was fully open during scanning (see Figure 6.5).

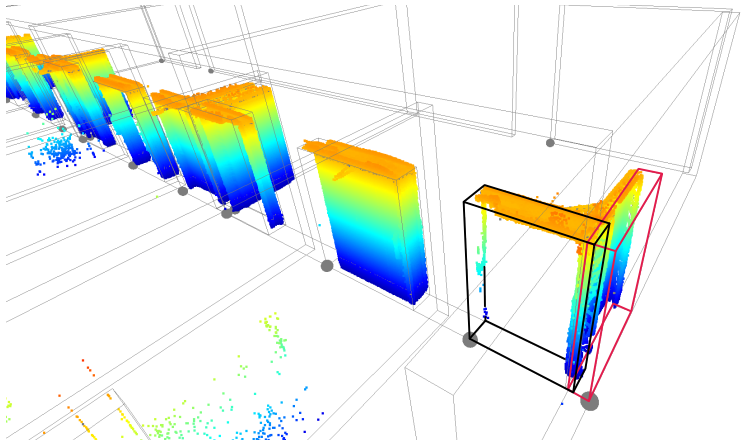


Figure 6.5: In the 14NH C surgery scene, duplicate door reconstructions are present. The correct reconstruction is shown in black, while the erroneous one is highlighted in red.

Regarding the column reconstruction in all scenes, it is important to note that the segmentation model struggles with accurately segmenting columns that protrude from or are directly adjacent to a wall.

Beyond this result overview, an ablation study can support the understanding of benefits that specific methods yield. Thus, three different configurations were examined on the A building first floor scene of the 14NH dataset.

- The **baseline** method includes the full reconstruction pipeline starting with the semantic segmentation, but does neither apply any topology refinement including height correction, wall merging and closing and door splitting, nor any dimension threshold or correction. Instead of HYSAC, RANSAC is used for primitive fitting.

- The **scaleBIM** method includes all algorithms and procedures as described in the previous sections and in Algorithm 20 encompassing semantic segmentation, topology correction and HYSAC plane fitting.

- To assess the effect of the semantic segmentation quality on the overall process, **manual labels** are used instead of the labels inferred by the transformer-based segmentation model.

Metrics	baseline	*ScaleBIM*	manual labels
Wall centroid deviation [m]	0.402	**0.334**	0.330
Wall bounding box IoU [-]	0.371	**0.395**	0.346
Wall vIoU 5 cm [-]	0.466	0.553	**0.571**
Door centroid deviation [m]	0.218	**0.158**	0.173
Door bounding box IoU [-]	0.322	0.417	**0.519**
Door vIoU 5 cm [-]	0.391	0.402	**0.521**

Table 6.2: Ablation study on the scene *A building first floor* of the 14NH dataset. No columns were scanned thus no results can be reported on this class. The vIoU was calculated at a voxel size of 5 cm.

Note, that the inverse opening reconstruction algorithm was neither applied in the *ScaleBIM* method nor was it applied in the manual labels method to demonstrate the impact of the sparse door segmentation.

Considering the numbers presented in Table 6.2 it remains to be noted, that the *ScaleBIM* approach yields considerable advantage regarding both wall and door reconstruction accuracy. This demonstrates the potential of the *ScaleBIM* approach leveraging topology refinement, height correction, wall merging and closing, door splitting and the more efficient HYSAC plane segmentation approach. However, this particular dataset is one of the cleanest used in this work, has very low occlusions and no interior objects. Especially the wall segmentation is very accurate, as it can be seen in Figure 6.6. Remarkably, the manual labels do not outperform the wall reconstruction of the *ScaleBIM* approach, that yields a 4.9% higher bounding box IoU, likewise the wall volumetric IoU is only marginally higher with the manual labels.

To further investigate the impact of the quality of the semantic segmentation on the overall reconstruction, a test was carried out using the manual annotations. This reveals, that there is some positive effect on the wall reconstruction accuracy (particularly vIoU, refer to Table 6.2), but a major positive effect could be observed for the reconstruction quality of doors, with the door IoU increased by 10.2% and the vIoU enhanced by 11.9%. The major conclusion is that although the *ScaleBIM* approach can cope with incomplete segmentation, as observed for doors in this dataset, a better segmentation accuracy boosts the results.

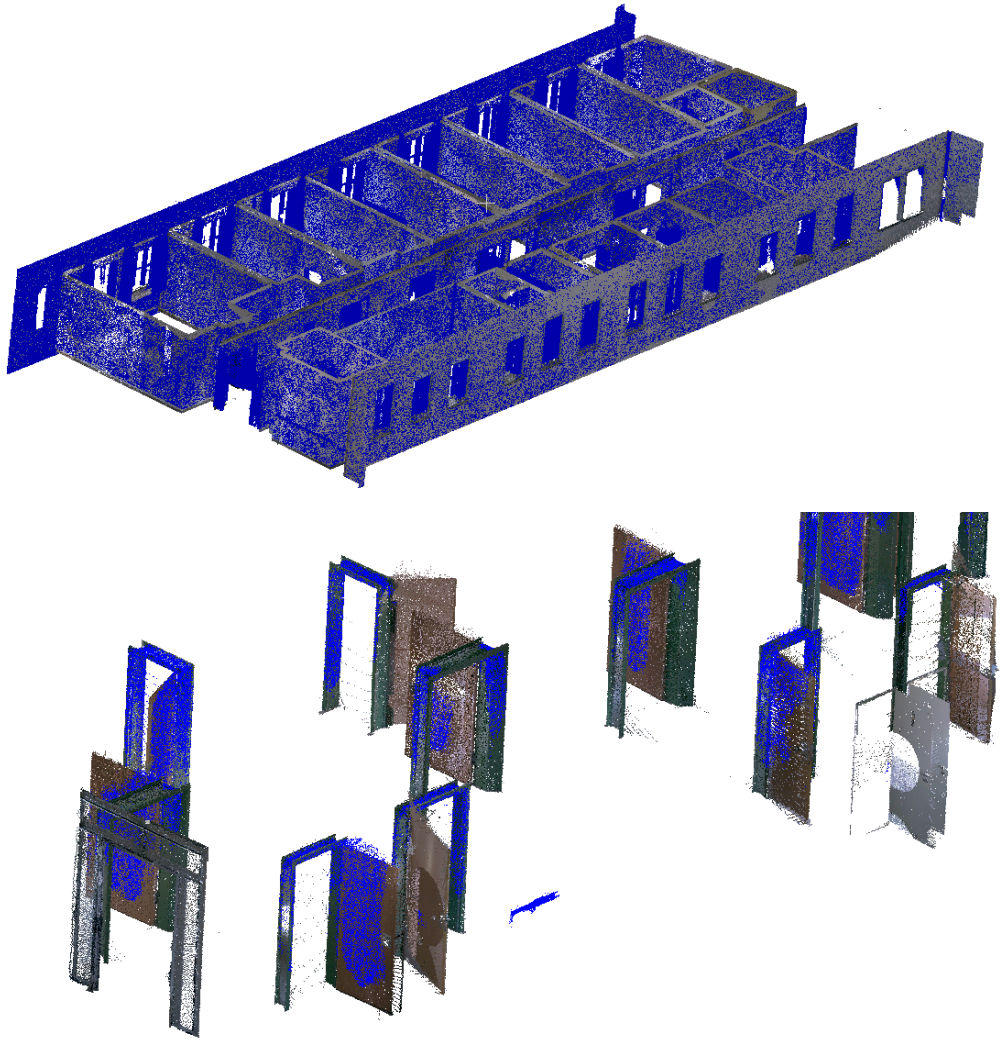


Figure 6.6: Top: 14NH A-Building 1st floor overlay of ground truth labels (natural colours) and segmentation (blue). Bottom: doors, respectively. Door segmentation is incomplete with missing points along the frames and some doors not segmented at all.

CV4AEC

The second dataset used is the Computer Vision in the Built Environment (CV4AEC) `https://cv4aec.github.io/` dataset. Unfortunately, no additional documentation on the type of building and sensor and circumstances of scanning is provided. Compared to the high quality of the 14NH dataset, the CV4AEC dataset is characterized by significant noise and clutter, as well as the presence of interiour objects (refer to section 3.3 for more details). This dataset presents an excellent opportunity to demonstrate the potential of the developed algorithms when applied to noisy data with occlusions. For the experiments, six scenes from the test data were used. In the challenge, no ground truth BIMs were provided for the test scenes. A considerable misalignment of the ground truth was observed in the training scenes. Discussions with the challenge organizers revealed that the provided ground truth was derived from 2D design drawings, which presents two major issues: (i) The actual building naturally differs from the design due to inaccuracies, tolerances, later additions, or changes. (ii) Design drawings may include details that even humans cannot infer from the scanned data, such as wall types, joints, and object classifications. To address these issues, the ground truth BIMs were created manually, ensuring that only objects identifiable from the data were included. To avoid any bias from the ground truth available in the training scenes, with which the author is familiar, the test scenes were selected for evaluation.

Metrics	08_ShortOffice_01_F1	08_ShortOffice_01_F2	11_MedOffice_05_F2	11_MedOffice_05_F4	25_Parking_01_F1	25_Parking_01_F2
Wall centroid deviation [m]	0.185	0.133	0.199	0.211	0.207	0.257
Wall bounding box IoU [-]	0.347	0.358	0.261	0.297	0.280	0.318
Wall vIoU 5 cm [-]	0.437	0.383	0.259	0.302	0.386	0.420
Door centroid deviation [m]	0.099	0.100	0.186	0.141	0.125	0.000
Door bounding box IoU [-]	0.388	0.452	0.157	0.182	0.192	0.000
Door vIoU 5 cm [-]	0.367	0.407	0.150	0.149	0.247	0.000
Column centroid deviation [m]	0.162	0.089	0.171	0.204	0.292	0.190
Column bounding box IoU [-]	0.382	0.400	0.011	0.033	0.197	0.105
Column vIoU 5 cm [-]	0.404	0.547	0.025	0.057	0.250	0.125
Mean bbox IoU [-]	0.372	0.403	0.143	0.171	0.223	0.141
Mean vIoU [-]	0.403	0.446	0.145	0.169	0.294	0.182

Table 6.3: Results of the evaluation on the CV4AEC dataset test scenes.

The results achieved using the *ScaleBIM* pipeline are reported in Table 6.3. Supplementary renderings of the reconstructed bounding boxes can be found in Annex 9.

The six scenes can be characterized as follows. The first two scenes (08_ShortOffice_...) were scanned

from an office building with a classic layout, including interiour walls that separate office rooms and rooms with other functions. This type of layout and topology aligns well with the design of our method, resulting in the best reconstruction outcomes for these scenes. The next two scenes (11_MedOffice_...) feature a large open space area with small booths, cubicles, and various types of office furniture and equipment. The inner courtyard and exterior facade consist of columns and a facade system. While the reconstruction pipeline successfully detects columns in the first two scenes, it struggles in the latter two, where there is minimal successful detection. This issue is related to the fact that columns intersecting with walls are rarely segmented and thus cannot be reconstructed (refer to Figure 6.7). This highlights the need for a more accurate segmentation algorithm capable of handling columns that interfere with other objects. Beyond the column segmentation, it is important to note that the segmentation algorithm seems to assume that the exteriors of rooms are enclosed by walls. However, Figure 6.7 shows that in these scenes, a pattern of windows, columns, or a facade system encloses the rooms along the building's outer perimeter. To address these cases, facade systems need to be incorporated into the segmentation training.

The last two scenes (25_Parking_...) were scanned in a car park. Since few vehicles are present in the data, the scenes are not occluded, allowing for a clear representation of the structure, including columns and perimeter walls. Compared to the previous two scenes, the column IoU and vIoU are significantly higher, although they still reach only about half the accuracy seen in the 08_ShortOffice scenes. In the final scene (25_Parking_01_F2), no doors were detected at all. This issue is again linked to the fact that no doors were segmented, further emphasizing the need for a more accurate segmentation model. Although the inverse opening detection algorithm was applied to the data, no successful reconstruction could be achieved, either because the doors were closed or the openings in the data were smaller than the threshold set by the algorithm. An alternative door detection algorithm could indeed be beneficial for identifying doors that were not segmented in the 3D point cloud and were closed during scanning.

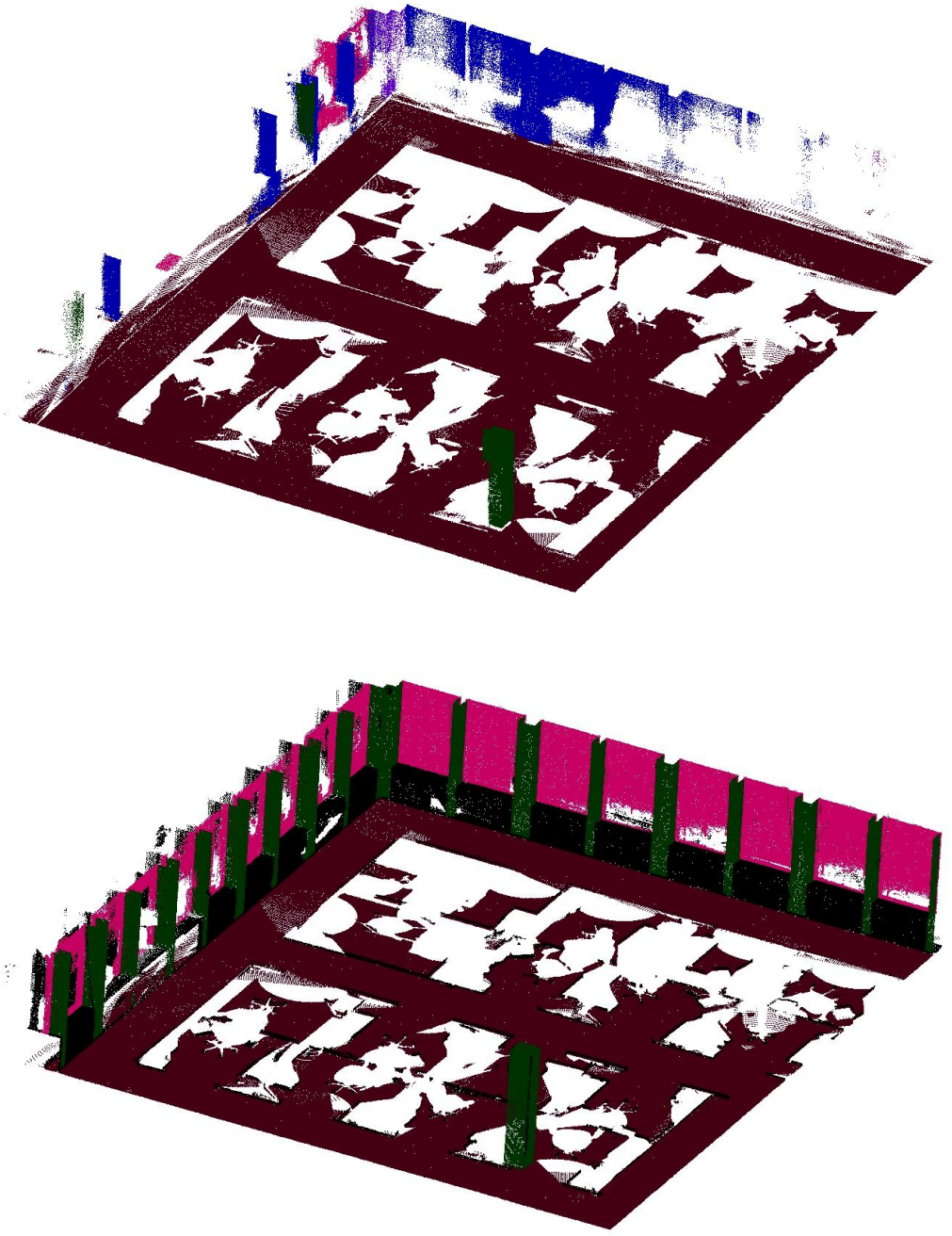


Figure 6.7: Column segmentation in 11_MedOffice_05_F4 scene. Top: Result of the point transformer segmentation. Bottom: Ground truth labels.

7 Conclusions

7.1 Discussion

Based on the results provided, it can be concluded that the *ScaleBIM* pipeline is a robust approach to deliver BIMs from buildings based on point clouds. One of the main limitations seems to be that only three object classes including wall, door and column are reconstructed in the current implementation. Obviously, this is not sufficient to deliver functional BIMs. In this context, we argue that the algorithms introduced can be easily applied to other elements.

Bassier et al. [79] report that the accuracy of the automatically created walls even outperforms the manual modelling. Hereby, the accuracy is evaluated by the minimum Euclidean distance of reconstructed IfcWallStandardCase and the ground truth mesh. Our experiments underpin this finding, as a comparison of the cloud-to-BIM distance of the ground truth BIM and the reconstructed BIM model reveals. For the 14NH A-Building 1st floor, the mean distance to the reconstructed BIM was 0.02 m, but for the ground truth BIM the distance is 0.052 m. The manually classified wall points and the reconstructed IfcWall objects were used to calculate the distance.

Putting the cloud-to-BIM distance of 0.02 m in context to the wall vIoU of 0.553 results in two conclusions. On the one hand, the pipeline approximates the input point cloud accurately with only a minimum distance of the reconstructed objects and the point cloud. On the other hand, the human-created ground truth BIM model contains all the inference and intelligence of the human modeller, who is capable of visually interpreting the configurations of the objects and thus applies corrections that the pipeline proposed in this work does not yet incorporate.

The topology refinement procedures applied are highly prone to improper setting of their input parameters. As the reconstruction of the 14NH A building second floor reveals, the wall correction applied rather deteriorates the reconstruction than improves it, as small wall instances are incorrectly merged into larger ones. Similar effects occur in almost all scenes.

Although a two-step door points search procedure was introduced (refer to subsection 4.7.1) to avoid duplicate door reconstruction, these are still present in the reconstructed data. A possible mitigation measure could be to introduce a separate semantic label for the door leaf and exclude accordingly labelled points from the door reconstruction. In any case, more advanced algorithms need to be developed to avoid such duplicate door reconstruction.

Regarding data availability, two key aspects should be noted. First, the data annotation guidelines, procedures, and datasets significantly influence semantic segmentation accuracy. However, only a fine-tuned model trained on the specific features of the data can achieve the level of accuracy required for the framework proposed here. Consequently, a fine-tuning would have to be performed before processing

any data with features differing from the dataset incorporated into the training. Second, despite not being a primary research focus, the availability of training data with high-quality semantic annotations remains a challenge. Initiating a dataset crowd-sourcing effort, similar to the ScanNet dataset [31], or deriving data from a commercially available service, could be valuable approaches, especially in cases where corresponding BIMs are available to enhance the efficiency of the data annotation process.

Theoretically, achieving 100% accurate semantic segmentation should result in 100% reconstruction accuracy. However, the ablation study revealed that this is not the case, with a maximum wall vIoU of only 0.571 being obtained with manually annotated data (see Table 6.2). This finding leads to the major conclusion of this study: enhanced reasoning algorithms are necessary to infer more geometric information about the observed objects. It is likely that point clouds, even with associated color values as processed here, may be insufficient for this task, necessitating the incorporation of additional data modalities. On an initial level, instance segmentation and image-based segmentation of sub-elements could be explored. Ultimately, a comprehensive BIM generation model could be envisioned, capable of processing various data modalities — including 2D design drawings as pixels or vectors, images, text, and 3D data — and directly inferring a BIM model in the IFC format.

Although not being part of this work, the topology correction algorithms is focussed on the objects only and thus faces some limitations.

Overall, the presented *ScaleBIM* framework in conjunction with the published algorithms and Python implementations offers the potential to further enhance the reconstruction pipelines towards other objects, to incorporate more semantic information and to enhance the reconstruction quality.

Regarding the potential use cases and benefits towards reducing GWP emissions as outlined in the Motivation section, it needs to be discussed whether the BIMs reconstructed using the proposed framework are applicable.

To assess the potential for Reuse, the primary information a BIM model could deliver is the basic configuration of building components and spaces. Reflecting the results presented, the reconstructed BIMs contain this information, although there are still limitations in the reconstruction quality. Going beyond the architectural layout planning stage into a more technical assessment it becomes obvious that the BIMs reconstructed lack vital information including material and its properties, and embedded objects such as installations are not being reconstructed by the framework proposed.

The key problem of Rebuild, i.e. extracting components from existing buildings, preserving them as a whole and reassembling a new structure with them, is to identify seems and joints thus assessing procedures for deconstruction. To this end, BIMs of the *ScaleBIM* framework are applicable given that the objects and joints are visible. However, a detailed inference towards the type of joints encountered cannot be delivered yet.

To assess the potential for Recycling materials from an existing structure, the proposed framework could deliver BIMs that can be used for quantity take-offs. However, information on the material and its properties would be critical, as this is the main information needed to determine whether a component could be recycled.

Considering the use cases and the concluding remarks, it remains to outline perspectives for future research in the next section.

7.2 Perspectives for future research

For future research, two main directions should be followed to leverage the reconstruction accuracy. The first is to enhance the *ScaleBIM* approach with more advanced reconstruction and topology correction procedures. The second direction would be to traverse towards a learning based end-to-end pipeline leveraging large models to infer the BIM semantics and parameters of the object directly from the input point cloud.

In the field of advancing the reconstruction algorithms, several perspectives for future research can be identified.

- When reconstructing opening elements such as windows and doors, the primary focus is typically on the points representing these elements. However, the parent object often contains a void corresponding to the dimensions of the opening. Combining these aspects could enhance the reconstruction accuracy of openings, particularly when the frame and lining are sparsely represented in the data. This approach has been tested in this work revealing that more advanced procedures could be beneficial, as our approach only performs well when no points inside the opening are present. However, artefacts are likely to occur, e.g. when glass panes inserted into parent objects without a dedicated frame. Such situations require more advanced and specialized reconstruction approaches.

- The reconstruction of facades has not been extensively researched in this study. The main challenge lies in accurately reconstructing and representing subcomponents like posts and transoms, which are essential for characterizing the facade, even though the overall facade can be easily represented as a generic object. Additionally, different types of cladding and exterior surfaces need to be considered during reconstruction.

- Although doors have been reconstructed in this work, no efforts have been made to characterize them more explicitly. Future work could focus on developing algorithms to reconstruct subcomponents such as door/window frames and linings, or to semantically characterize doors, e.g. identifying fire protection doors.

- Incorporating additional data modalities into the door reconstruction process could help prevent duplicate door reconstructions. Essentially, the door parent object needs to be unambiguously identified.

- Throughout this work, and with the increasing availability of training data, segmentation accuracy has improved significantly. However, objects that are adjacent to or protrude from other objects, such as columns and walls, still lack sufficient reconstruction accuracy. Developing additional segmentation procedures and applying them after semantic segmentation and the reconstruction pipeline could be beneficial.

- Similarly, combining semantic segmentation with a geometric segmentation post-processing step could improve accuracy. For example, after performing plane segmentation of wall points, any inliers of a different class could be reassigned the wall label.

Looking more broadly, Schönfelder et al. [28] have suggested incorporating multiple data sources

in the reconstruction process such as construction drawings and documents, along with respective AI methods. This certainly is a gap in the current research, which primarily focusses on one data domain. In this context, and in respect to the proposition of end-to-end learning-based methods, current trends in AI research yield promising results of cross-domain learning or inference, e.g. deriving textual descriptions of scenes [117]. Indeed, the trend towards large AI *foundation* models will influence scan-to-BIM research. As such models can be applied within their domain in a general manner on various use cases, there is no alternative to integrate these into scan-to-BIM pipelines and provide the algorithms around the inference of these models.

From a civil engineer's point of view, the question remains how objects embedded inside others can be scanned and reconstructed. The main groups of objects relevant are installations and obscured structural members. Among the last, concrete reinforcement is quite common and the main factor to assess the structural behaviour of concrete components. Ground penetrating radar (GPR) can be used to capture data of concrete components, recent scanning devices and processing software are even capable of capturing larger areas and deliver 3D data of the reinforcement steel bars. Although not included into *ScaleBIM*, some experiments have been conducted to use GPR data to deliver a BIM reconstruction of steel reinforcement and an example is presented in Figure 7.1. An area of 1.2 x 1.2 m was scanned with a Proceq GPR 800 device. An approach similar to reconstructing round columns (refer to section 4.8) was used to reconstruct the geometry of the reinforcement steel bars.

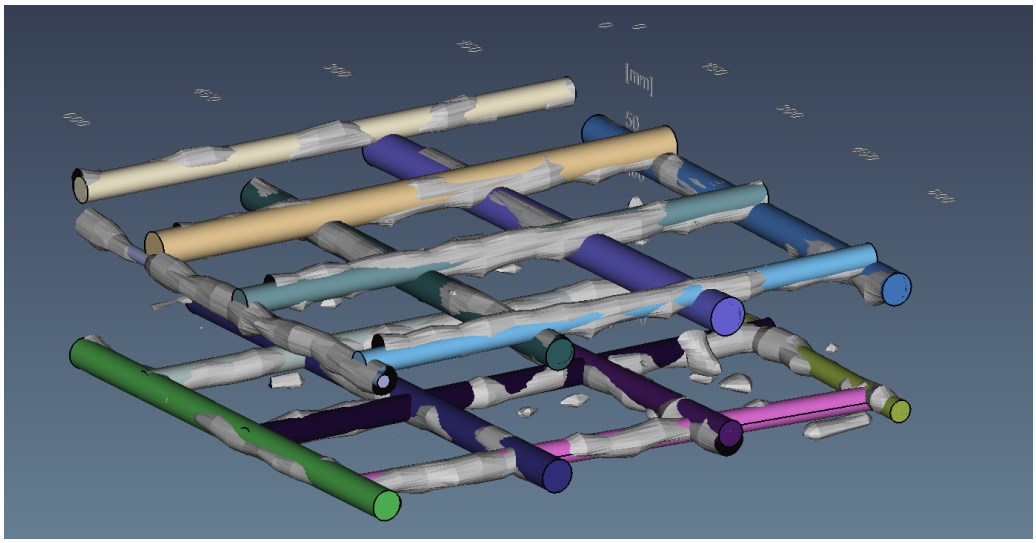


Figure 7.1: BIM reconstruction of steel reinforcement bars. The reconstruction is based on a GPR scan of the concrete piece.

For a complete integration into *ScaleBIM*, the GPR data still needs to be aligned with the building point cloud. As the scanning is based on a grid that needs to be followed by the GPR scanner, markers on the grid paper could be used to align both domains of data. Among that, two more limitations apply: (i) Although theoretically larger areas could be scanned, it is a time-consuming process, thus the same coverage of the objects within the building point cloud cannot be achieved. (ii) GPR cannot be used to retrieve the diameter of the reinforcement bars, which is crucial for structural calculations. Yet, additional

non-destructive testing can be applied to estimate the reinforcement diameter, which in turn complicates the overall data acquisition process.

Similar to GPR data, other data modalities, e.g. thermal and hyperspectral could be incorporated into the reconstruction. The premise is that the data can be aligned with the point cloud.

7.3 Lessons learned

What remains to be noted are key observations and learnings that could be helpful to future researchers.

During the experiments on semantic segmentation of point clouds, a considerable effort was spent to generate a synthetic dataset which would make manual data annotation obsolete. During the course of this work, this approach was neglected as a decent prediction accuracy on real-world data could not be achieved. In further studies, the potential of synthetic training data should be assessed based on a literature study before concentrating on such an approach.

Constantly using the publicly available benchmark datasets both in the fields of semantic segmentation and scan-to-BIM will lead us to a dead end. Although some researchers might question the value of the scientific contribution of a benchmark dataset, it should be considered to incorporate these in future studies. Clearly, the available ones do not reflect the wide range of features and configurations of the built environment and are indeed limited to a very similar style of interior office and residential scenes. The 14NH and CV4AEC dataset providing reference BIM models along with point-level annotations used in this work will be published and others should follow to provide more options to evaluate their approaches to researchers.

The numerous open research questions in this field, combined with the potential of large AI models, should inspire future researchers to explore this area further. Given the complexity of both the available AI models and the demands of the civil engineering field, an interdisciplinary research approach is essential for success. While AI and computer vision research can yield exceptional algorithms, they are unlikely to be effectively adapted to address practically relevant problems without collaboration. On the other hand, the civil engineering domain may struggle to achieve the same level of understanding and expertise in the other necessary fields on its own.

Acknowledgements

This work is part of the HumanTech project and has received funding from the European Union under Grant Agreement No. 101058236. HumanTech is a three-year project that started on the 1st June 2022 and terminates on the 31st May 2025.

Prior to HumanTech, this research was funded by the ERDF (European Regional Development Fund) and the German federal state of Rhineland-Palatinate in the research programme InnoProm. The research was co-funded and accompanied by the construction enterprise P.A. Budau GmbH `https://www.budau.com/` situated in Idar-Oberstein, Rhineland Palatinate, Germany.

Bibliography

Bibliography

[1] Chuck Eastman et al. *BIM handbook: A guide to building information modeling for owners, managers, designers, engineers, and contractors*. 2. ed. Hoboken, NJ: Wiley, 2011. ISBN: 978-0470541371.

[2] Kerstin Hausknecht and Thomas Liebich. *BIM-Kompendium: Building Information Modeling als neue Planungsmethode*. Stuttgart: Fraunhofer IRB Verlag, 2016. ISBN: 978-3-8167-9490-5.

[3] Buildingsmart. *Industry Foundation Classes: Version 4.2 bSI Draft Standard IFC Bridge proposed extension*. Ed. by Buildingsmart. 2019. URL: https://standards.buildingsmart.org/IFC/DEV/IFC4_2/FINAL/HTML/ (visited on 10/01/2022).

[4] Daniel Girardeau-Montaut. *CloudCompare: 3D point cloud and mesh processing software*. 2020. URL: http://www.cloudcompare.org/ (visited on 10/06/2020).

[5] Hoesung Lee et al. *IPCC, 2023: Climate Change 2023: Synthesis Report. Contribution of Working Groups I, II and III to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change [Core Writing Team, H. Lee and J. Romero (eds.)]. IPCC, Geneva, Switzerland*. 2023. DOI: 10.59327/IPCC/AR6-9789291691647.

[6] Roland Bechmann and Stefanie Weidner. "Graue Emissionen im Bauwesen: Bestandsaufnahme und Optimierungsstrategien". In: *DBV-Heft 50 Nachhaltiges Bauen mit Beton: Band 1*. Ed. by Lars Meyer. Berlin, 2022.

[7] European Commission. *Regulation (EU) 2021/1119 of the European Parliament and of the Council of 30 June 2021 establishing the framework for achieving climate neutrality and amending Regulations (EC) No 401/2009 and (EU) 2018/1999 ('European Climate Law')*. 2021. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32021R1119.

[8] Christian Glock et al. "Massivbau in Zeiten von Klimawandel und Ressourcenverknappung – Herausforderungen und Lösungsansätze/Concrete construction in times of climate change and resource shortage – challenges and solutions". In: *Bauingenieur* 97.01-02 (2022), pp. 1–12. ISSN: 0005-6650. DOI: 10.37544/0005-6650-2022-01-02-33.

[9] WWF Deutschland, ed. *Klimaschutz in der Beton- und Zementindustrie: Hintergrund und Handlungsoptionen*. Berlin, 2019.

[10] Christian Glock et al. "Treibhausgas- und ressourcenreduzierter (Beton)Bau: Herausforderungen, Lösungsansätze, Anreizsysteme". In: *DBV-Heft 50 Nachhaltiges Bauen mit Beton: Band 1*. Ed. by Lars Meyer. Berlin, 2022.

[11] Bundesministerium für Verkehr und digitale Infrastruktur, ed. *Stufenplan Digitales Planen und Bauen: Einführung moderner, IT-gestützter Prozesse und Technologien bei Planung, Bau und Betrieb von Bauwerken*. 2015.

[12] DB Station&Service AG, ed. *Vorgaben zur Anwendung der BIM-Methodik: Digitales Planen und Bauen*. Berlin, 2020. URL: https://www1.deutschebahn.com/resource/blob/1786332/1c0d47f32e6d4a8e221a7019f5fdb4ce/Vorgaben-zur-Anwendung-der-BIM-Methodik-data.pdf.

[13] Markus König et al. *Wissenschaftliche Begleitung der BMVI Pilotprojekte zur Anwendung von BIM im Infrastrukturbau: Materialsammlung*. 2016. URL: https://www.bmvi.de/SharedDocs/DE/Anlage/Digitales/bim-materialsammlung.html.

[14] Markus König et al. *Wissenschaftliche Begleitung der BMVI Pilotprojekte zur Anwendung von BIM im Infrastrukturbau: Endbericht*. 2018. URL: https://www.bmvi.de/SharedDocs/DE/Anlage/DG/wissenschaftliche-begleitung-anwendung-bim-infrastrukturbau-2018.pdf.

[15] Christian Glock et al. "Klima– und ressourcenschonendes Bauen mit Beton". In: *BetonKalender 2024*. Ed. by Konrad Bergmeister, Frank Fingerloos, and Johann–Dietrich Wörner. Wiley, 2023, pp. 177–265. ISBN: 9783433034064. DOI: 10.1002/9783433611494.ch2.

[16] Fabian Kaufmann, Christian Glock, and Thomas Tschickardt. "ScaleBIM: Introducing a scalable modular framework to transfer point clouds into semantically rich building information models". In: *Proceedings of the 2022 European Conference on Computing in Construction*. Computing in Construction. University of Turin, 2022. DOI: 10.35490/EC3.2022.194.

[17] Fabian Kaufmann et al. "Ontology-based semantic labeling for RGB-D and point cloud datasets". In: *Proceedings of the 2023 European Conference on Computing in Construction and the 40th International CIB W78 Conference*. Computing in Construction. European Council for Computing in Construction, 2023. DOI: 10.35490/EC3.2023.241.

[18] Marius Schellen et al. "Annotation rules and classes for semantic segmentation of point clouds for digitalization of existing bridge structures". In: *Proceedings of the 2023 European Conference on Computing in Construction and the 40th International CIB W78 Conference*. Computing in Construction. European Council for Computing in Construction, 2023. DOI: 10.35490/EC3.2023.228.

[19] F. Kaufmann, C. Glock, and T. Tschickardt. "Drone-based acquisition of as-built models for the automation of processes within the digital management of bridge assets". In: *The 8th International Symposium on Reliability Engineering and Risk Management* (2022), pp. 91–98. URL: https://rpsonline.com.sg/rps2prod/isrerm2022/epro/html/toc.html.

[20] Thomas Tschickardt, Fabian Kaufmann, and Christian Glock. "Lean and BIM based flight planning for automated data acquisition of bridge structures with LiDAR UAV during construction phase". In: *Proceedings of the 2022 European Conference on Computing in Construction*. Computing in Construction. University of Turin, 2022. DOI: 10.35490/EC3.2022.146.

[21] James M. Coughlan and Alan L. Yuille. “Manhattan world: Compass direction from a single image by bayesian inference”. In: *Proceedings of the seventh IEEE international conference on computer vision*. Vol. 2. 1999, pp. 941–947.

[22] Mahdi Chamseddine, Jason Rambach, and Fabian Kaufmann. “BIMStruct3D: A fully automated hybrid learning Scan-to-BIM pipeline with watertight topology reconstruction: [Manuscript in preparation]”. In: (2024).

[23] Thomas Krijnen. *IfcOpenShell*. 2021. URL: http://ifcopenshell.org/.

[24] E. Kwak, M. Al-Durgham, and A. Habib. “AUTOMATIC 3D BUILDING MODEL GENERATION FROM LIDAR AND IMAGE DATA USING SEQUENTIAL MINIMUM BOUNDING RECTANGLE”. In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XXXIX-B3 (2012), pp. 285–290. DOI: 10.5194/isprsarchives-XXXIX-B3-285-2012.

[25] Zhixin Li and Jie Shan. “RANSAC-based multi primitive building reconstruction from 3D point clouds”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 185 (2022), pp. 247–260. ISSN: 09242716. DOI: 10.1016/j.isprsjprs.2021.12.012.

[26] Ruodan Lu, Ioannis Brilakis, and Campbell R. Middleton. “Detection of Structural Components in Point Clouds of Existing RC Bridges”. In: *Computer-Aided Civil and Infrastructure Engineering* 34.3 (2019), pp. 191–212. ISSN: 10939687. DOI: 10.1111/mice.12407.

[27] M. Saeed Mafipour, Simon Vilgertshofer, and André Borrmann. “Automated geometric digital twinning of bridges from segmented point clouds by parametric prototype models”. In: *Automation in Construction* 156 (2023), p. 105101. ISSN: 09265805. DOI: 10.1016/j.autcon.2023.105101.

[28] Phillip Schönfelder et al. “Automating the retrospective generation of As-is BIM models using machine learning”. In: *Automation in Construction* 152 (2023), p. 104937. ISSN: 09265805. DOI: 10.1016/j.autcon.2023.104937.

[29] Jong Won Ma, Thomas Czerniawski, and Fernanda Leite. “Semantic segmentation of point clouds of building interiors with deep learning: Augmenting training datasets with synthetic BIM-based point clouds”. In: *Automation in Construction* 113 (2020), p. 103144. ISSN: 09265805. DOI: 10.1016/j.autcon.2020.103144.

[30] Florian Noichl, Alexander Braun, and André Borrmann. “BIM-TO-SCAN FOR SCAN-TO-BIM: GENERATING REALISTIC SYNTHETIC GROUND TRUTH POINT CLOUDS BASED ON INDUSTRIAL 3D MODELS”. In: *2021 European Conference on Computing in Construction Ixia, Rhodes, Greece* (2021).

[31] Angela Dai et al. “ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes”. In: (2017). URL: http://arxiv.org/pdf/1702.04405v2 (visited on 01/28/2022).

[32] Iro Armeni et al. “3D Semantic Parsing of Large-Scale Indoor Spaces”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 1534–1543. ISBN: 978-1-4673-8851-1.

[33] Maarten Bassier, Bjorn van Genechten, and Maarten Vergauwen. “Classification of sensor independent point cloud data of building objects using random forests”. In: *Journal of Building Engineering* 21 (2019), pp. 468–477. ISSN: 23527102. DOI: 10.1016/j.jobe.2018.04.027.

[34] Angela Dai et al. “ScanComplete: Large-Scale Scene Completion and Semantic Segmentation for 3D Scans”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 2018, pp. 4578–4587. ISBN: 978-1-5386-6420-9. DOI: 10.1109/CVPR.2018.00481.

[35] Bonsang Koo, Raekyu Jung, and Youngsu Yu. “Automatic classification of wall and door BIM element subtypes using 3D geometric deep neural networks”. In: *Advanced Engineering Informatics* 47 (2021), p. 101200. ISSN: 1474-0346. DOI: 10.1016/j.aei.2020.101200.

[36] Yeritza Perez-Perez, Mani Golparvar-Fard, and Khaled El-Rayes. “Segmentation of point clouds via joint semantic and geometric features for 3D modeling of the built environment”. In: *Automation in Construction* 125 (2021), p. 103584. ISSN: 09265805. DOI: 10.1016/j.autcon.2021.103584.

[37] Andrey Dimitrov and Mani Golparvar-Fard. “Segmentation of building point cloud models including detailed architectural/structural features and MEP systems”. In: *Automation in Construction* 51.Part B (2015), pp. 32–45. ISSN: 09265805. DOI: 10.1016/J.AUTCON.2014.12.015.

[38] Yeritza Perez-Perez, Mani Golparvar-Fard, and Khaled El-Rayes. “Scan2BIM-NET: Deep Learning Method for Segmentation of Point Clouds for Scan-to-BIM”. In: *Journal of Construction Engineering and Management* 147.9 (2021). ISSN: 0733-9364. DOI: 10.1061/(ASCE)CO.1943-7862.0002132.

[39] L. S. Runceanu and N. Haala. “INDOOR MESH CLASSIFICATION FOR BIM”. In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLII-4 (2018), pp. 535–539. DOI: 10.5194/isprs-archives-XLII-4-535-2018.

[40] Chao Yin et al. “Automated semantic segmentation of industrial point clouds using ResPointNet++”. In: *Automation in Construction* 130 (2021), p. 103874. ISSN: 09265805. DOI: 10.1016/j.autcon.2021.103874.

[41] Charles R. Qi et al. “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space”. In: *arXiv:1706.02413v1* (2017). URL: http://arxiv.org/pdf/1706.02413v1.

[42] Xiaoyang Wu et al. “Towards Large-scale 3D Representation Learning with Multi-dataset Point Prompt Training”. In: *arXiv:2308.09718* (2023).

[43] Xiaoyang Wu et al. “Point Transformer V2: Grouped Vector Attention and Partition-based Pooling”. In: *NeurIPS*. 2022.

[44] Hengshuang Zhao et al. “Point Transformer”. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 2021, pp. 16239–16248. ISBN: 978-1-6654-2812-5. DOI: 10.1109/ICCV48922.2021.01595.

[45] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. “3D Semantic Segmentation with Submanifold Sparse Convolutional Networks”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 2018, pp. 9224–9232. ISBN: 978-1-5386-6420-9. DOI: 10.1109/CVPR.2018.00961.

[46] Christopher Choy, JunYoung Gwak, and Silvio Savarese. “4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, pp. 3070–3079. ISBN: 978-1-7281-3293-8. DOI: 10.1109/CVPR.2019.00319.

[47] Hugues Thomas et al. “KPConv: Flexible and Deformable Convolution for Point Clouds”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 2019, pp. 6410–6419. ISBN: 978-1-7281-4803-8. DOI: 10.1109/ICCV.2019.00651.

[48] Qingyong Hu et al. “RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020, pp. 11105–11114. ISBN: 978-1-7281-7168-5. DOI: 10.1109/CVPR42600.2020.01112.

[49] Dario Rethage et al. *Fully-Convolutional Point Networks for Large-Scale Point Clouds*. 2018. URL: https://arxiv.org/pdf/1808.06840.pdf.

[50] Hang Su et al. “SPLATNet: Sparse Lattice Networks for Point Cloud Processing”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 2018, pp. 2530–2539. ISBN: 978-1-5386-6420-9. DOI: 10.1109/CVPR.2018.00268.

[51] Lyne Tchapmi et al. “SEGCloud: Semantic Segmentation of 3D Point Clouds”. In: *2017 International Conference on 3D Vision (3DV)*. IEEE, 2017, pp. 537–547. ISBN: 978-1-5386-2610-8. DOI: 10.1109/3DV.2017.00067.

[52] Charles R. Qi. et al. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 77–85. ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.16.

[53] Zhisheng Zhong et al. “Understanding Imbalanced Semantic Segmentation Through Neural Collapse”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 19550–19560.

[54] Ben Graham. *Sparse 3D convolutional neural networks*. 2015. URL: http://arxiv.org/pdf/1505.02890v2.

[55] Charles R. Qi et al. “Volumetric and Multi-view CNNs for Object Classification on 3D Data”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 5648–5656. ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.609.

[56] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. *OctNet: Learning Deep 3D Representations at High Resolutions*. 2016. URL: http://arxiv.org/pdf/1611.05009v4.

[57] Benjamin Graham and Laurens van der Maaten. “Submanifold Sparse Convolutional Networks”. In: *arXiv preprint arXiv:1706.01307* (2017).

[58] Jens Behley et al. “Towards 3D LiDAR-based semantic scene understanding of 3D point cloud sequences: The SemanticKITTI Dataset”. In: *The International Journal of Robotics Research* 40.8-9 (2021), pp. 959–967. ISSN: 0278-3649. DOI: 10.1177/02783649211006735.

[59] Maarten Bassier, Maarten Vergauwen, and Bjorn van Genechten. “Automated Semantic Labelling of 3D Vector Models for Scan-to-BIM”. In: *Annual International Conference on Architecture and Civil Engineering (ACE 2016)* (2016). DOI: 10.5176/2301-394X{\textunderscore}ACE16.83.

[60] H. Macher, T. Landes, and P. Grussenmeyer. “Point clouds segmentation as base for as-built BIM creation”. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* II-5/W3 (2015), pp. 191–197. DOI: 10.5194/isprsannals-II-5-W3-191-2015.

[61] Ioannis Anagnostopoulos et al. “Detection of Walls, Floors, and Ceilings in Point Cloud Data”. In: *Construction Research Congress 2016*. Ed. by José L. Perdomo-Rivera et al. Reston, VA: American Society of Civil Engineers, 2016, pp. 2302–2311. ISBN: 9780784479827. DOI: 10.1061/9780784479827.229.

[62] Georgios-Tsampikos Michailidis and Renato Pajarola. “Bayesian graph-cut optimization for wall surfaces reconstruction in indoor environments”. In: *The Visual Computer* 33.10 (2017), pp. 1347–1355. ISSN: 0178-2789. DOI: 10.1007/s00371-016-1230-3.

[63] Jean-Jacques Ponciano et al. “Object Semantic Segmentation in Point Clouds—Comparison of a Deep Learning and a Knowledge-Based Method”. In: *ISPRS International Journal of Geo-Information* 10.4 (2021), p. 256. DOI: 10.3390/ijgi10040256.

[64] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. “Efficient RANSAC for Point-Cloud Shape Detection”. In: *Computer Graphics Forum* 26.2 (2007), pp. 214–226.

[65] I. Armeni et al. “Joint 2D-3D-Semantic Data for Indoor Scene Understanding”. In: *ArXiv e-prints* (2017).

[66] Alexander Kirillov et al. *Segment Anything*. 2023. URL: http://arxiv.org/pdf/2304.02643v1.

[67] James MacQueen et al. “Some methods for classification and analysis of multivariate observations”. In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 1967, pp. 281–297.

[68] Martin Ester et al. “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD’96. AAAI Press, 1996, pp. 226–231.

[69] Christina Petschnigg et al. “From a Point Cloud to a Simulation Model-Bayesian Segmentation and Entropy Based Uncertainty Estimation for 3D Modelling”. In: *Entropy (Basel, Switzerland)* 23.3 (2021). DOI: 10.3390/e23030301.

[70] Mihael Ankerst et al. “OPTICS: Ordering points to identify the clustering structure”. In: *ACM Sigmod record* 28.2 (1999), pp. 49–60.

[71] Adrien Kaiser, Jose Alonso Ybanez Zepeda, and Tamy Boubekeur. “A Survey of Simple Geometric Primitives Detection Methods for Captured 3D Data”. In: *Computer Graphics Forum* 38.1 (2019), pp. 167–196. DOI: 10.1111/cgf.13451.

[72] Lingxiao Li et al. “Supervised Fitting of Geometric Primitives to 3D Point Clouds”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, pp. 2647–2655. ISBN: 978-1-7281-3293-8. DOI: 10.1109/CVPR.2019.00276.

[73] Tsahi Saporta and Andrei Sharf. “Unsupervised recursive deep fitting of 3D primitives to points”. In: *Computers & Graphics* 102 (2022), pp. 289–299. ISSN: 00978493. DOI: 10.1016/j.cag.2021.10.020.

[74] Viorica Pătrăucean et al. “State of research in automatic as-built modelling”. In: *Advanced Engineering Informatics* 29.2 (2015), pp. 162–171. ISSN: 1474-0346. DOI: 10.1016/j.aei.2015.01.001.

[75] Pingbo Tang et al. “Automatic reconstruction of as-built building information models from laser-scanned point clouds: A review of related techniques”. In: *Automation in Construction* 19.7 (2010), pp. 829–843. ISSN: 09265805. DOI: 10.1016/j.autcon.2010.06.007.

[76] Rebekka Volk, Julian Stengel, and Frank Schultmann. “Building Information Modeling (BIM) for existing buildings — Literature review and future needs”. In: *Automation in Construction* 38 (2014), pp. 109–127. ISSN: 09265805. DOI: 10.1016/j.autcon.2013.10.023.

[77] Yusheng Xu and Uwe Stilla. “Toward Building and Civil Infrastructure Reconstruction From Point Clouds: A Review on Data and Key Techniques”. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 14 (2021), pp. 2857–2885. ISSN: 1939-1404. DOI: 10.1109/JSTARS.2021.3060568.

[78] C. Gourguechon, H. Macher, and T. Landes. “AUTOMATION OF AS-BUILT BIM CREATION FROM POINT CLOUD: AN OVERVIEW OF RESEARCH WORKS FOCUSED ON INDOOR ENVIRONMENT”. In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLIII-B2-2022 (2022), pp. 193–200. DOI: 10.5194/isprs-archives-XLIII-B2-2022-193-2022.

[79] Maarten Bassier and Maarten Vergauwen. “Unsupervised reconstruction of Building Information Modeling wall objects from point cloud data”. In: *Automation in Construction* 120 (2020), p. 103338. ISSN: 09265805. DOI: 10.1016/j.autcon.2020.103338.

[80] Sebastian Ochmann et al. “Automatic reconstruction of parametric building models from indoor point clouds”. In: *Computers & Graphics* 54 (2016), pp. 94–103. ISSN: 00978493. DOI: 10.1016/j.cag.2015.07.008.

[81] Hélène Macher, Tania Landes, and Pierre Grussenmeyer. “From Point Clouds to Building Information Models: 3D Semi-Automatic Reconstruction of Indoors of Existing Buildings”. In: *Applied Sciences* 7.10 (2017), p. 1030. DOI: 10.3390/app7101030.

[82] Biao Xiong et al. “Knowledge-driven inference for automatic reconstruction of indoor detailed as-built BIMs from laser scanning data”. In: *Automation in Construction* 156 (2023), p. 105097. ISSN: 09265805. DOI: 10.1016/j.autcon.2023.105097.

[83] M. Bassier et al. “IFC WALL RECONSTRUCTION FROM UNSTRUCTURED POINT CLOUDS”. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* IV-2 (2018), pp. 33–39. DOI: 10.5194/isprs-annals-IV-2-33-2018.

[84] Sebastian Ochmann, Richard Vock, and Reinhard Klein. “Automatic reconstruction of fully volumetric 3D building models from oriented point clouds”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 151 (2019), pp. 251–262. ISSN: 09242716. DOI: 10.1016/j.isprsjprs.2019.03.017.

[85] Shengjun Tang et al. “BIM generation from 3D point clouds by combining 3D deep learning and improved morphological approach”. In: *Automation in Construction* 141 (2022), p. 104422. ISSN: 09265805. DOI: 10.1016/j.autcon.2022.104422.

[86] P.H.S. Torr and A. Zisserman. “MLESAC: A New Robust Estimator with Application to Estimating Image Geometry”. In: *Computer Vision and Image Understanding* 78.1 (2000), pp. 138–156. ISSN: 10773142. DOI: 10.1006/cviu.1999.0832.

[87] Maarten Bassier, Meisam Yousefzadeh, and Maarten Vergauwen. “Comparison of 2D and 3D wall reconstruction algorithms from point cloud data for as-built BIM”. In: *Journal of Information Technology in Construction* 25 (2020), pp. 173–192. DOI: 10.36680/j.itcon.2020.011.

[88] Uuganbayar Gankhuyag and Ji-Hyeong Han. “Automatic BIM Indoor Modelling from Unstructured Point Clouds Using a Convolutional Neural Network”. In: *Intelligent Automation & Soft Computing* 28.1 (2021), pp. 133–152. ISSN: 1079-8587. DOI: 10.32604/iasc.2021.015227.

[89] Ioannis Anagnostopoulos, Michael Belsky, and Ioannis Brilakis. “Object Boundaries and Room Detection in As-Is BIM Models from Point Cloud Data”. In: *Proceedings of the 16th International Conference on Computing in Civil and Building Engineering, Osaka, Japan*. 2016, pp. 6–8.

[90] Kaveh Mirzaei et al. “Automatic generation of structural geometric digital twins from point clouds”. In: *Scientific reports* 12.1 (2022), p. 22321. DOI: 10.1038/s41598-022-26307-7.

[91] Chao Wang, Yong K. Cho, and Changwan Kim. “Automatic BIM component extraction from point clouds of existing buildings for sustainability applications”. In: *Automation in Construction* 56 (2015), pp. 1–13. ISSN: 09265805. DOI: 10.1016/j.autcon.2015.04.001.

[92] Hyunsoo Kim and Changwan Kim. “3D as-built modeling from incomplete point clouds using connectivity relations”. In: *Automation in Construction* 130 (2021), p. 103855. ISSN: 09265805. DOI: 10.1016/j.autcon.2021.103855.

[93] Taehoon Kim et al. “PinSout”. In: *Proceedings of the 28th International Conference on Advances in Geographic Information Systems*. Ed. by Chang-Tien Lu et al. New York, NY, USA: ACM, 2020, pp. 211–214. ISBN: 9781450380195. DOI: 10.1145/3397536.3422343.

[94] Hyojoo Son and Changwan Kim. “Semantic as-built 3D modeling of structural elements of buildings based on local concavity and convexity”. In: *Advanced Engineering Informatics* 34.6 (2017), pp. 114–124. ISSN: 1474-0346. DOI: 10.1016/j.aei.2017.10.001.

[95] Charles Thomson and Jan Boehm. “Automatic Geometry Generation from Point Clouds for BIM”. In: *Remote Sensing* 7.9 (2015), pp. 11753–11775. DOI: 10.3390/rs70911753.

[96] R. Assi et al. “ENERGY FUNCTION ALGORITHM FOR DETECTION OF OPENINGS IN INDOOR POINT CLOUDS”. In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLII-2/W13 (2019), pp. 747–752. DOI: 10.5194/isprs-archives-XLII-2-W13-747-2019.

[97] Jingdao Chen and Yong K. Cho. “Exemplar-Based Building Element Retrieval from Point Clouds”. In: *International Conference on Smart Infrastructure and Construction 2019 (ICSIC)*. Ed. by M. J. DeJong, J. M. Schooling, and G. M.B. Viggiani. ICE Publishing, 2019, pp. 225–231. ISBN: 978-0-7277-6466-9. DOI: 10.1680/icsic.64669.225.

[98] Ulrich Krispel, Torsten Ullrich, and Martin Tamke. “Formalising Expert Knowledge for Building Information Models: Automated Identification of Electrical Wiring from 3D Scans”. In: *Keeping Up with Technologies to Create the Cognitive City*. Ed. by Eva Vani\Dukić Lazarević. Cambridge Scholars Publishing, 2019, pp. 318–328. ISBN: 978-1-5275-2048-6.

[99] Yongzhi Xu and Xuesong Shen. “Automatic As-Built BIM with 3D Object Detection by Learning Building Structure Knowledge”. In: *Construction Research Congress 2020*. Ed. by Pingbo Tang, David Grau, and Mounir El Asmar. Reston, VA: American Society of Civil Engineers, 2020, pp. 556–565. ISBN: 9780784482865. DOI: 10.1061/9780784482865.059.

[100] K. Babacan, L. Chen, and G. Sohn. “SEMANTIC SEGMENTATION OF INDOOR POINT CLOUDS USING CONVOLUTIONAL NEURAL NETWORK”. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* IV-4/W4 (2017), pp. 101–108. DOI: 10.5194/isprs-annals-IV-4-W4-101-2017.

[101] Inge Coudron et al. “Semantic Extraction of Permanent Structures for the Reconstruction of Building Interiors from Point Clouds”. In: *Sensors (Basel, Switzerland)* 20.23 (2020). DOI: 10.3390/s20236916.

[102] Maarten Bassier and Maarten Vergauwen. “Topology Reconstruction of BIM Wall Objects from Point Cloud Data”. In: *Remote Sensing* 12.11 (2020), p. 1800. DOI: 10.3390/rs12111800.

[103] Reza Maalek, Derek D. Lichti, and Janaka Y. Ruwanpura. “Automatic Recognition of Common Structural Elements from Point Clouds for Automated Progress Monitoring and Dimensional Quality Control in Reinforced Concrete Construction”. In: *Remote Sensing* 11.9 (2019), p. 1102. DOI: 10.3390/rs11091102.

[104] Wenzhong Shi et al. “Semantic Geometric Modelling of Unstructured Indoor Point Cloud”. In: *ISPRS International Journal of Geo-Information* 8.1 (2019), p. 9. DOI: 10.3390/ijgi8010009.

[105] Xuehan Xiong et al. “Automatic creation of semantically rich 3D building models from laser scanner data”. In: *Automation in Construction* 31 (2013), pp. 325–337. ISSN: 09265805. DOI: 10.1016/j.autcon.2012.10.006.

[106] Jingdao Chen, Zsolt Kira, and Yong K. Cho. “Deep Learning Approach to Point Cloud Scene Understanding for Automated Scan to 3D Reconstruction”. In: *Journal of Computing in Civil Engineering* 33.4 (2019). ISSN: 0887-3801. DOI: 10.1061/(ASCE)CP.1943-5487.0000842.

[107] Epic Games. *Unreal Engine 4*. 2023. URL: https://www.unrealengine.com.

[108] Charles R. Harris et al. “Array programming with NumPy”. In: *Nature* 585.7825 (2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2.

[109] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[110] OpenAI et al. *GPT-4 Technical Report*. 2023. URL: http://arxiv.org/pdf/2303.08774v6.

[111] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. “Open3D: A Modern Library for 3D Data Processing”. In: *arXiv:1801.09847* (2018).

[112] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. “The Quickhull Algorithm for Convex Hulls”. In: *ACM Trans. Math. Softw.* 22.4 (1996), pp. 469–483. ISSN: 0098-3500. DOI: 10.1145/235815.235821.

[113] European Committee for Standardization, ed. *EN 1992-1-1 Eurocode 2: Design of concrete structures - Part 1-1: General rules and rules for buildings*. Brussels: CEN, 2011.

[114] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[115] H. W. Kuhn. “The Hungarian method for the assignment problem”. In: *Naval Research Logistics Quarterly* 2.1-2 (1955), pp. 83–97. ISSN: 0028-1441. DOI: 10.1002/nav.3800020109.

[116] Nikhila Ravi et al. “Accelerating 3D Deep Learning with PyTorch3D”. In: *arXiv:2007.08501* (2020).

[117] Armen Avetisyan et al. *SceneScript: Reconstructing Scenes With An Autoregressive Structured Language Model*. 2024. URL: http://arxiv.org/pdf/2403.13064v1.

Annex

Annex 1: Declaration on AI tools

AI tools such as Large Language Models (LLMs) slowly becoming available to a wide range of applications, can be useful in scientific work and research. This section is to declare what tools have been used and what purpose these have been used for. This section only refers to tools for general applications. Specific methods that are part of this work are described in the respective topic sections.

The models GPT-3.5 and GPT-4 [110] from openAI have been used to support the following tasks:

- To rephrase sentences and paragraphs or to improve the style aiming for a more fluent and readable text. The drafted text has been presented to the model asking for rephrasing and improvement (prompt: 'please rephrase preserving LaTeXnotation: ... text to rephrase ...' or 'please improve style and grammar preserving LaTeXnotation: ... text to improve ...'). The model output has then been revised thoroughly and integrated into this thesis.

- To support the implementation of methods, algorithms and this pipeline as described in the thesis using the python programming language. The LLMs have been prompted to provide small code snippets, functions or implementations of common algorithms. All outputs have been checked thoroughly, integrated into functions and/or classes and tested to ensure the expected behaviour.

- Provide LaTeX code snippets, e.g. to arrange images or figures, create and edit tables, or convert text into LaTeX code.

Annex 2: Student theses relevant to this work

Name	Title
Peter Erhard	Umsetzung der integrierten Planungsmethode BIM in Entwurfsplanung, Konstruktion, Kalkulation, Fertigteilproduktion und Bauausführung am Beispiel eines Bauunternehmens
Marie Honorine Mukiza Kirezi	Entwicklung von methodischen Ansätzen für die Materialklassifikation in Laserscans von Gebäuden
Margarita Schick	Erzeugung von synthetischen Datasets für die semantische Segmentierung von Punktwolken
Serdy Metangmo Nguekeu	Ansätze zur teilautomatisierten Nachmodellierung von Punktwolken im open source CAD-System FreeCAD
Lasitha Hofmann-Ganegoda	Vergleich marktverfügbarer Softwarelösungen zur Überführung von Punktwolken in BIM-Modelle (Scan to BIM)
Dimitri Heil	Vergleich von Technologielösungen zur digitalen Erfassung und automatisierten BIM-Modellierung von Bauwerken
Yuyiu Alfred Au	Vorschläge für die Erweiterung des IfC bridge standards für bestehende Bauwerke unter Berücksichtigung vorhandener Regelwerke für das Monitoring
Christian Flieger	Geometrieextraktion aus Punktwolken mittels visuell programmierbarer, parametrischer Modellierungswerkzeuge
Serdy Metangmo Nguekeu	BIM im Ingenieurbau: Parametrische Modellierung zur Erzeugung informierter, digitaler Bauwerksmodelle für statische Berechnungen mit Hilfe visueller Programmierschnittstellen
Jan Niklas Rohde	Modellierungsrichtlinie für BIM-Bestandsbauteile unter Berücksichtigung der Bemessungsvorgaben zum Bauen im Bestand und Standards für die Datenübergabe
Mohammad Alrefaei	Methodische Ansätze für die Materialklassifikation in Laserscans von Gebäuden – Weiterentwicklung und praktische Erprobung
Eric Höh	Der Blick ins Bauteil - Möglichkeiten zur dreidimensionalen Erfassung des Bauteilinneren - Stand der Technik und Potenziale für die digitale Bestandserfassung
Hanna Kirsch	Erzeugung von synthetischen Trainingsdaten aus BIM-Modellen für die semantische Segmentierung von Punktwolken im Rahmen von Scan to BIM workflows
Cheng Lu	Bildbasierte semantische Erkennung und Segmentierung von Betonschädigungen als Beitrag zur Bewertung von bestehenden Betontragwerken
Nadja Danter	Entwicklung und Validierung von Verfahren zur geometrischen Rekonstruktion von Bauteilen im Rahmen von Scan to BIM
Eric Höh	Rebar-to-BIM: Validierung der Genauigkeit und Zuverlässigkeit eines Verfahrens zur dreidimensionalen Rekonstruktion von Bewehrung in Betonbauteilen
Dongling Liu	Softwarewerkzeuge und Workflows zur Annotation von Punktwolken
Cheng Lu	Geometrische Klassifikation von segmentierten Punktwolken mittels Machine Learning

Annex 3: List of repositories and resources

This annex provides a list of repositories and online sources that supplement this work. Unless otherwise specified, all resources are openly accessible and free to use.

HumanTech data annotation

This repository contains the annotation guidelines that were used to annotate the data used in this work. It does not only specify labels in the building category, but also in the interior and construction class.

`https://gitlab.rhrk.uni-kl.de/kaufmann/humantech-data-annotation`

The complete annotation guidelines in the building category are provided in Annex 4.

ScaleBIM

This repository contains the entire source code that implements the end-to-end pipeline as specified in chapter 4 including test data and scripts and instructions. Currently, it is not openly accessible, access can be granted on demand.

`https://gitlab.rhrk.uni-kl.de/scan-to-bim/scaleBIM`

Pystruct3D

This repository includes methods for creating, fitting, and manipulating bounding boxes, along with annotation tools, metrics calculations, and various utility functions. Currently under development, it is not yet publicly accessible but will be made available once it reaches a sufficient level of maturity for an initial release.

`https://github.com/humantecheu/pystruct3d`

OpenBIMxD

This repository contains a python module that can be used to create BIM models from the previously reconstructed geometry. Beyond the IFC objects creation, it contains methods to create semantic labels in a point cloud from BIM geometry and other methods for IFC object filtering, updating and parsing.

`https://github.com/humantecheu/openbimxd`

Annex 4: Data annotation guidelines

In the following section, the complete data annotation guidelines of the building category will be presented. The other two categories interior and construction are excluded as they are not relevant to this work. Note, that the annotation guidelines are subject to continuous changes and improvements. The most recent version can be found online: `https://gitlab.rhrk.uni-kl.de/kaufmann/humantech-data-annotation`

Building

Slab

HT ID	Name	IFC class	R	G	B	Hex color
1	slab	IfcSlab	170	0	0	AA0000

Description

IFC entity definition

A slab is a component of the construction that normally encloses a space vertically. The slab may provide the lower support (floor) or upper construction (roof slab) in any space in a building. Only the core or constructional part of this construction is considered to be a slab. The upper finish (flooring, roofing) and the lower finish (ceiling, suspended ceiling) are considered to be coverings. A special type of slab is the landing, described as a floor section to which one or more stair flights or ramp flights connect. Landing is annotated as part of the Stair instance.

Additional description

A slab is just the structural element, for example, a structural concrete monolithic slab. Typically, the slab is only visible in buildings under construction, with a few exceptions where the surfaces of slabs are left exposed intentionally for aesthetic or functional reasons, such as exposed concrete. Any layers above, such as screed or flooring, or layers below, such as drywall ceilings are part of the respective classes. (Floor or Ceiling)

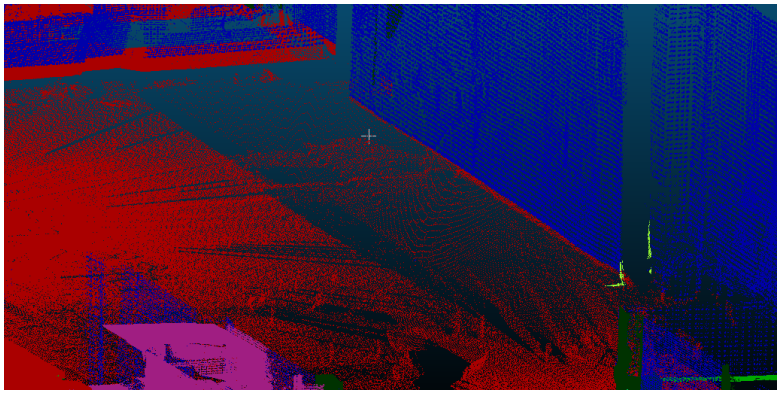
Annotation rules

In some cases it might be hard to distinguish the slab from a floor or a ceiling. Depending on the state of a building under construction the slab might be still visible or already covered with layers associated to floor and ceiling. This needs to be decided according to the texture of the surfaces if possible. If accessible, 'as-built' documentation should be referred to for additional clarity on the instance's identity. Precision is key, particularly in relation to measurements and details on slab edges. Slabs are generally horizontal. Inclined elements enclosing spaces or a building are ramps or roofs. Slabs are structural.

Slab may only be annotated when the structural slab (concrete, timber framing, ...) is visible. In other cases floor or ceiling labels should be used. Balconies should be annotated as slab. There is no dedicated balcony class. This applies for balconies that are part of the monolithic slab of the inside of the building. Other types of balcony constructions independent from the rest of the building need to be treated separately.

Image annotation

Point cloud annotation



Floor

HT ID	Name	IFC class	R	G	B	Hex color
2	floor	IfcCovering	70	0	20	460014

Description

IFC entity definition

Note: The class 'IfcCovering' is a general classification for coverings, and isn't exclusive to floor coverings. A covering is defined as an element which covers some part of another element and is fully dependent on this parent element. The occurrence of a covering type is defined by 'IfcCovering' and,

when available, the exact type is specified by 'IfcCoveringType'. Coverings are elements with relationships to the parent element and the space on the other side. They may contain openings, which are assigned by IfcRelVoidsElement, and material information,which is assigned by IfcRelAssociatesMaterial, among others. For example, coverings can include wall claddings, floorings, suspended ceilings, moldings, and skirting boards.

Additional description

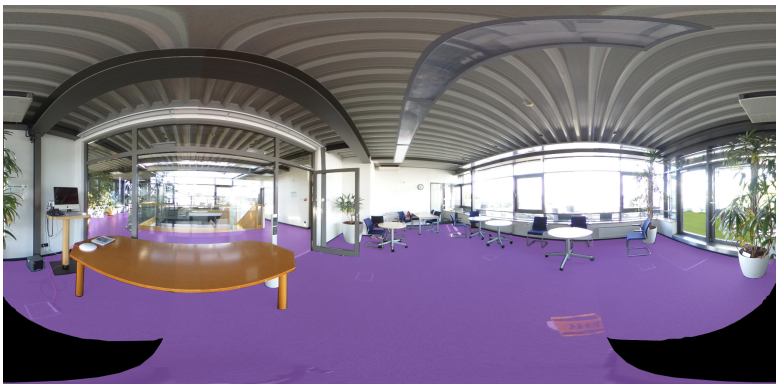
Floors are the bottom horizontal enclosing element of spaces.

Annotation rules

Precision is key, particularly in relation to measurements and details on floor edges. Floors are generally horizontal, inclined surfaces should be annotated as ramps or roofs. If the structural part of the floor is visible, it should be annotated as slab. Any layer above the slab is considered as part of the floor even if it is unfinished.

Image annotation

The Floor should be a one piece instance in RGB-D Images. There are two coned shaped missing data parts to the bottom right and left of each indoor image, these black parts are not to be annotated as part of the floor.



Point cloud annotation

Ceiling

HT ID	Name	IFC class	R	G	B	Hex color
3	ceiling	entity	70	0	120	460078

Description

IfC entity definition

Note: The class 'IfcCovering' is a general classification for coverings, and isn't exclusive to ceiling coverings. A covering is defined as an element which covers some part of another element and is fully dependent on this parent element. The occurrence of a covering type is defined by 'IfcCovering' and, when available, the exact type is specified by 'IfcCoveringType'. Coverings are elements with relationships to the parent element and the space on the other side. They may contain openings, which are assigned by IfcRelVoidsElement, and material information,which is assigned by IfcRelAssociatesMaterial, among others. For example, coverings can include wall claddings, floorings, suspended ceilings, moldings, and skirting boards.

Additional description

Ceilings are the top horizontal enclosing element of spaces. Ceilings include suspended ceilings and coverings of structural elements.

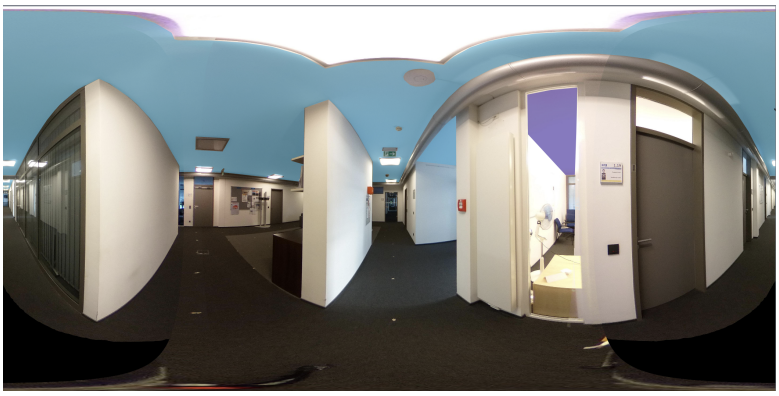
Annotation rules

Precision is key, particularly in relation to measurements and details on ceiling edges. Ceilings are generally horizontal, inclined surfaces should be annotated as ramps or roofs. If the structural part of the ceiling is visible, it should be annotated as slab. Any layer above the slab is considered as part of the floor even if it is unfinished.
markdownRendererInterblockSeparator

Image annotation

Ceiling should be annotated as a one piece instance not including lights, CCTV cameras, exit signs, pipes or anything else covering the visible flat surface of the ceiling.

In some Images the lights are stretched, these stretched lights are not part of the ceiling.



Point cloud annotation

Wall

HT ID	Name	IFC class	R	G	B	Hex color
4	wall	IfcWall	0	0	170	0000AA

Description

IfC entity definition

A wall represents a vertical construction that bounds or subdivides spaces. Walls are usually vertical, or nearly vertical, planar elements, often designed to bear structural loads. A wall is however not required to be load bearing.

Additional description

Annotation rules

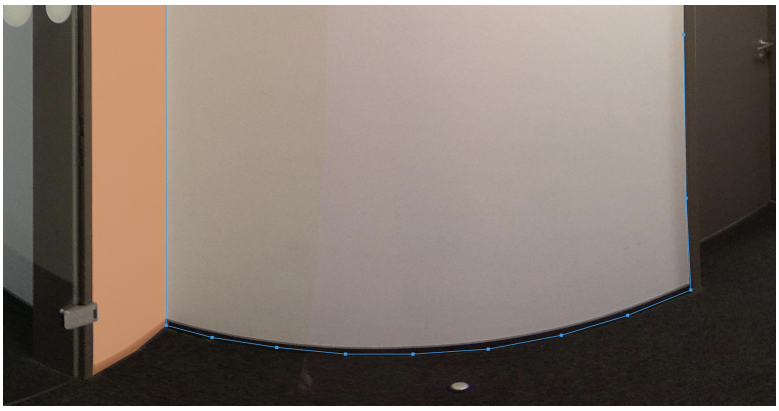
Precision is key, particularly in relation to measurements and details on wall edges. Walls may be from glass, bigger glass wall areas should be considered walls (see annotation rules for doors and

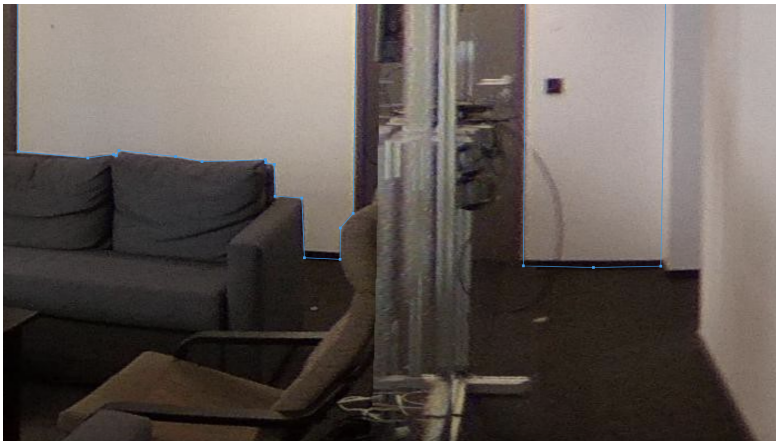
windows). A wall is generally limited vertically by a floor or slab underneath it and (possibly) a ceiling or slab surface above it, exceptions include short walls, walls under windows, and openings. If a vertical member is connected to the floor but not with a horizontal top surface below the ceiling, it is annotated as wall, i. e. walls do not have to reach up to the ceiling.

Image annotation

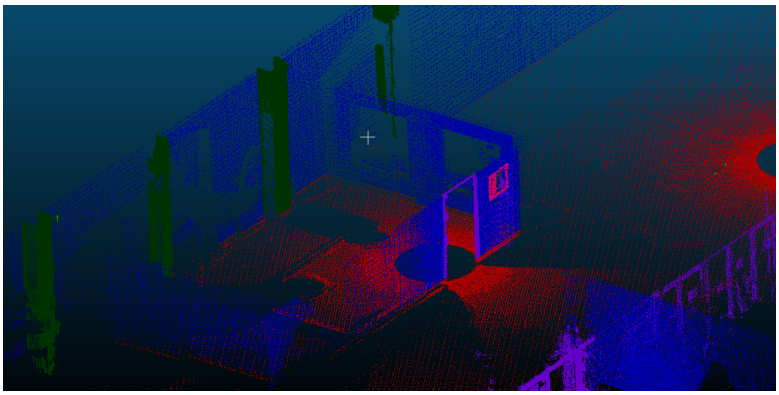
Light switches and maps (with negligible thickness) that stick to the wall, are part of the wall instance. In flat walls: If the normal to the surface is changed, another instance of a wall should be introduced. In curved Walls: The curved part should be annotated separately from connecting walls.

Note that the wall's baseboard (the darker bottom part of the wall) should be annotated as part of the wall.





Point cloud annotation



Pipe segment horizontal

HT ID	Name	IFC class	R	G	B	Hex color
5	pipe segment horizontal	IfcPipeSegment	0	100	200	0064C8

IfC entity definition

A pipe segment is used to typically join two sections of a piping network.

Additional description

Only horizontal pipe segments will be annotated under this label.

Annotation rules

The separation line with the pipe fitting should be as precise as possible and perpendicular to the pipe segment's center axis.
Attention: Pipe and pipe fitting are two separete entities.

Image annotation

Point cloud annotation

Pipe fitting

HT ID	Name	IFC class	R	G	B	Hex color
6	pipe fitting	IfcPipeFitting	0	150	150	009696

IfC entity definition

A pipe fitting is a junction or transition in a piping flow distribution system used to connect pipe segments, resulting in changes in flow characteristics of the fluid such as direction or flow rate. Pipe fittings include elbows, junctions, manifolds, and plumbing boxes.

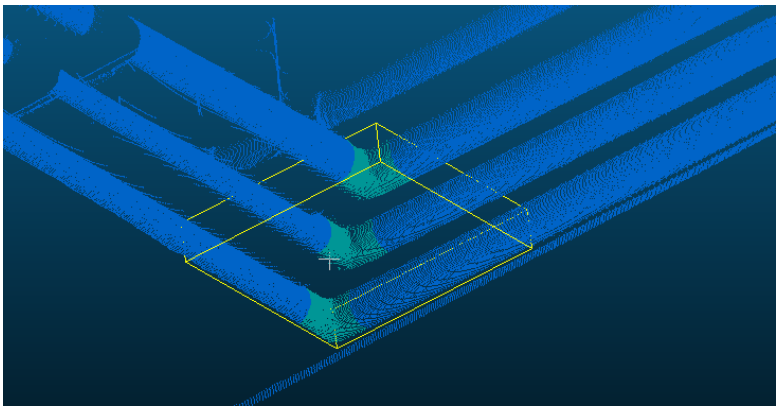
Additional description

Annotation rules

If the separation line cannot be visually identified, it should be annotated as perpendicular to the pipe segment that is connected to the pipe fitting. An educated guess about the measurements can be made based on other images in the dataset. This condition applies in both directions.

Image annotation

Point cloud annotation



Pipe segment vertical

HT ID	Name	IFC class	R	G	B	Hex color
7	pipe segment vertical	IfcPipeSegment	0	200	100	00C864

IfC entity definition

A pipe segment is used to typically join two sections of a piping network.

Additional description

Only vertical pipe segments are annotated under this label.

Annotation rules

Image annotation

Point cloud annotation

Door

HT ID	Name	IFC class	R	G	B	Hex color
8	door	IfcDoor	100	0	200	6400C8

IfC entity definition

A door is a building element that is predominately used to provide controlled access for people and goods. It includes constructions with hinged, pivoted, sliding, and additionally revolving and folding operations. A door consists of a lining and one or several panels.

NOTE Definition according to ISO 6707-1: construction for closing an opening, intended primarily for access with hinged, pivoted or sliding operation.

Additional description

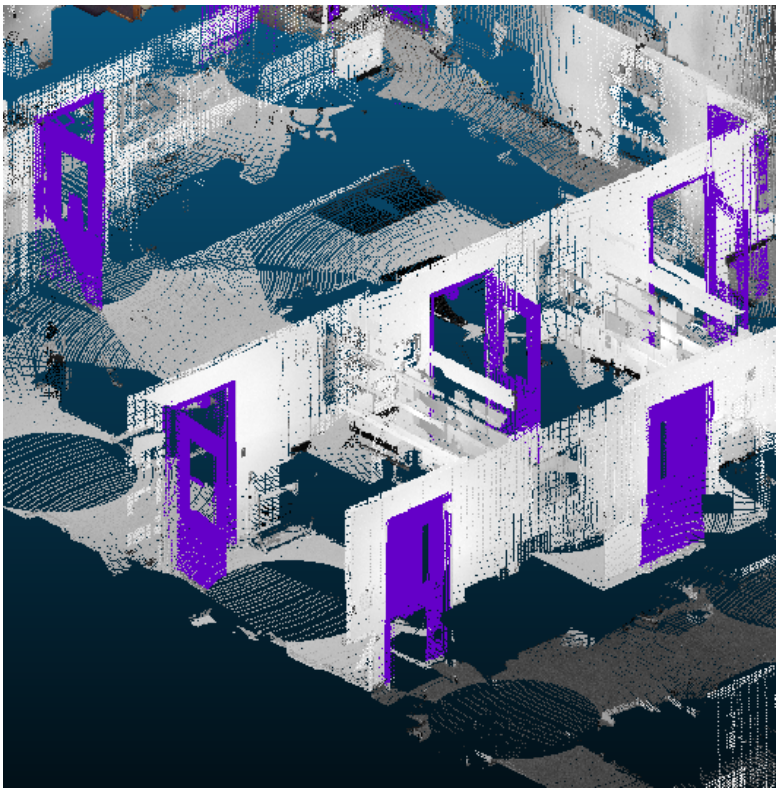
Annotation rules

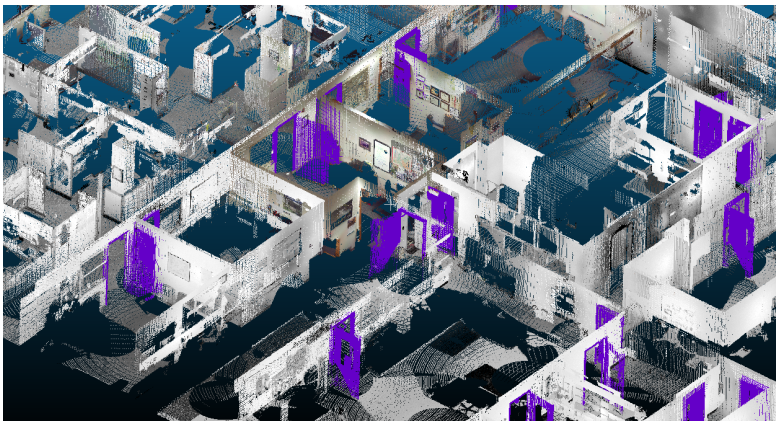
Annotation includes only door frame, leaf and linings. A glass door leaf or glass panes inside a door leaf will be annotated as door since they represent design choice not functionality. Any objects in the opening area of the door, but not door frame, leaf or lining must be excluded and annotated under the respective class! Objects visible through a glass pane or glass door leaf are annotated with their respective label and with an occlusion attribute. Fixed glass area will be labeled as door if it is 1.5 * door area. In other cases it should be annotated as wall. There is no generic object such as an opening. If an opening in a wall is surrounded by a door frame, but no leaf is present, it should be annotated as door. A plain opening will not receive a label, because it is not represented by data and thus cannot be annotated. Instead, the object that contains this opening should be annotated according to its respective category. Elevator doors are also annotated as doors.

Image annotation

Since most of the doors in the Dataset are closed, the default property of the door is closed; if it is open, just the frame and lining of the door (if leaf is not visible) should be annotated with the 'open' boolean property set to TRUE.

Point cloud annotation





Window

HT ID	Name	IFC class	R	G	B	Hex color
11	window	IfcWindow	200	0	100	C80064

IfC entity definition

The window is a building element that is predominately used to provide natural light and fresh air. It includes vertical opening but also horizontal opening such as skylights or light domes. It includes constructions with swinging, pivoting, sliding, or revolving panels and fixed panels. A window consists of a lining and one or several panels. NOTE Definition according to ISO 6707-1: Construction for closing a vertical or near vertical opening in a wall or pitched roof that will admit light and may admit fresh air.

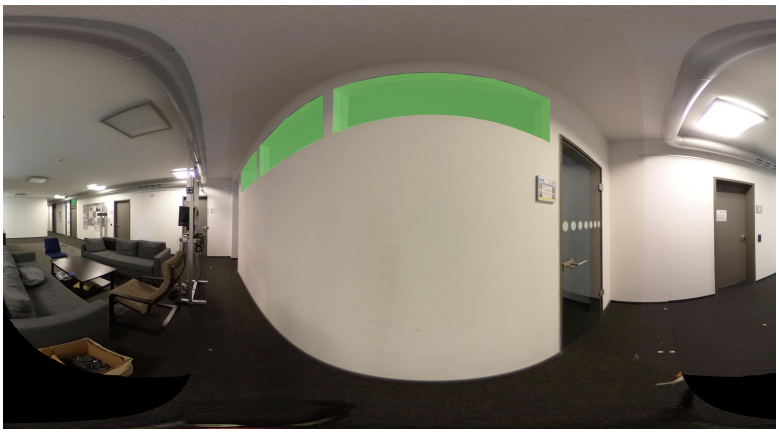
Additional description

Annotation rules

Windows shall be annotated including window frame and lining. Glass panes should have the attribute glassLayers set to TRUE. Objects visible through glass panes should be assigned their respective label but with an occlusion attribute. Glass panes including their frame (i. e. fixed window inside a wall) should be annotated as a window. Windows may be of any size and dimension, but note that larger ones may actually be (external) curtain walls. Windows may reach from the floor to the ceiling, especially windows with fixed panes. However, when full-height windows function as access points to specific areas, rooms, or spaces, they should be annotated as doors instead. Skylights should also be annotated as window, see Ifc entity definition.

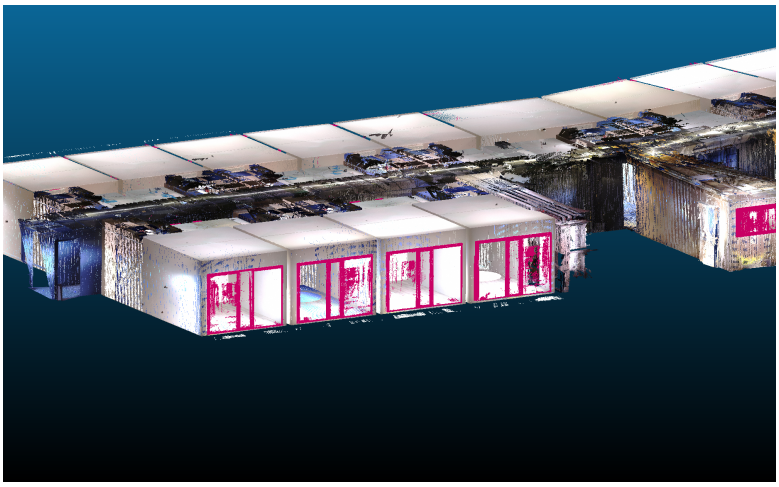
Image annotation

Since most of the windows in the Dataset are closed, the default property of the window is closed; if its open, just the frame and lining of the door should be annotated with a boolean property Open set to TRUE.



Point cloud annotation

In point clouds reflections of the glass pane will be annotated as window, see example. Artefacts and noise outside the window must be annotated as invalid.



Stair

HT ID	Name	IFC class	R	G	B	Hex color
12	stair	IfcStair	160	30	130	A01E82

IfC entity definition

A stair is a vertical passageway allowing occupants to walk (step) from one floor level to another floor level at a different elevation. It may include a landing as an intermediate floor slab. NOTE Definition according to ISO 6707-1: Construction comprising a succession of horizontal stages (steps or landings) that make it possible to pass on foot to other levels.

Additional description

Annotation rules

Stair category includes landings and flights, excluding walls and railings which have their own respective categories.

Image annotation

Point cloud annotation

Roof

HT ID	Name	IFC class	R	G	B	Hex color
13	roof	IfcRoof	60	80	100	3C5064

IfC entity definition

A roof is the covering of the top part of a building, it protects the building against the effects of wheather.
NOTE Definition according to ISO 6707-1: construction enclosing the building from above.

Additional description

Roofs may be inclined or horizontal. In case a roof is horizontal the top surface should be annotated as roof, the bottom surface (from inside) as ceiling. The class slab should only be used for a visible (concrete) structural slab.

Annotation rules

Image annotation

Point cloud annotation

Column

HT ID	Name	IFC class	R	G	B	Hex color
14	colum	IfcColumn	0	50	0	003200

IfC entity definition

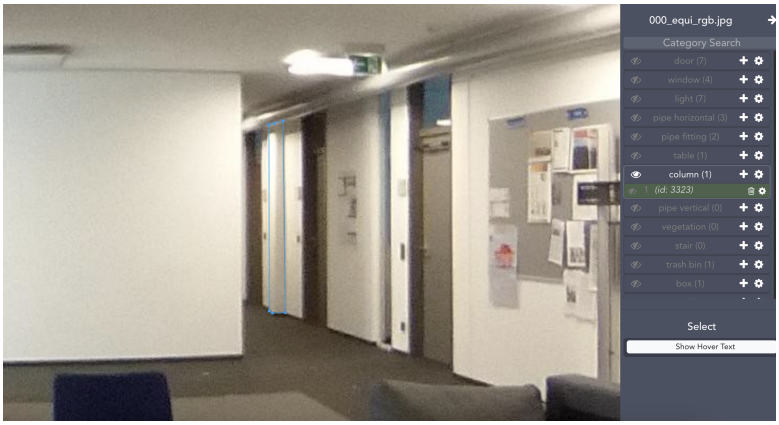
IfcColumn is a vertical structural member which often is aligned with a structural grid intersection. It represents a vertical, or nearly vertical, structural member that transmits, through compression, the weight of the structure above to other structural elements below. It represents such a member from an architectural point of view. It is not required to be load bearing. NOTE Definition according to ISO 6707-1: structural member of slender form, usually vertical, that transmits to its base the forces, primarily in compression, that are applied to it.

Additional description

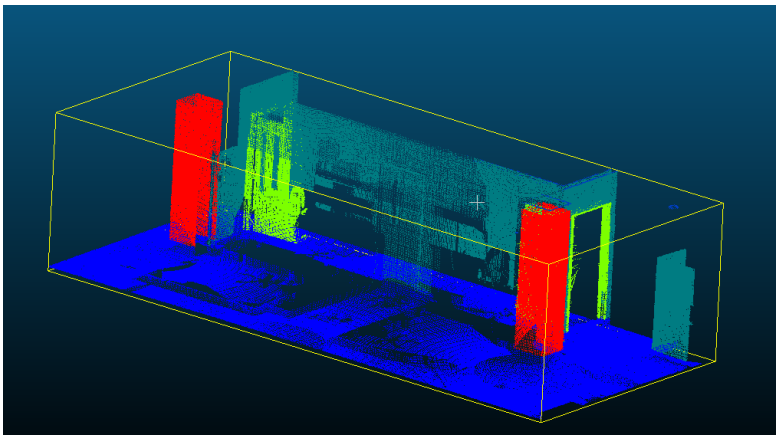
Annotation rules

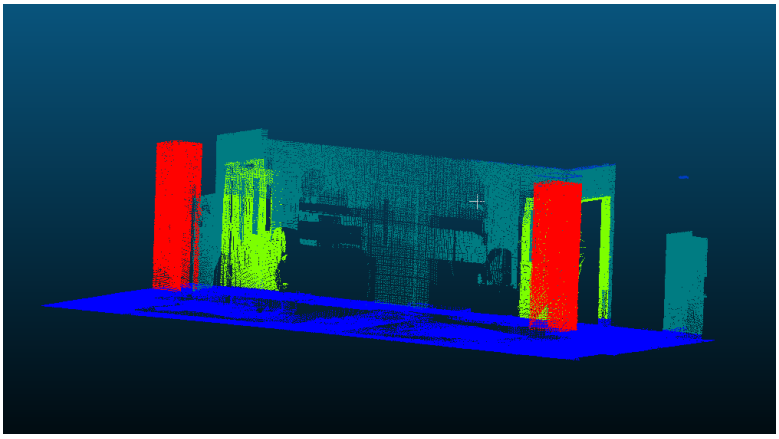
Precision is key, particularly in relation to measurements and details on floor edges.

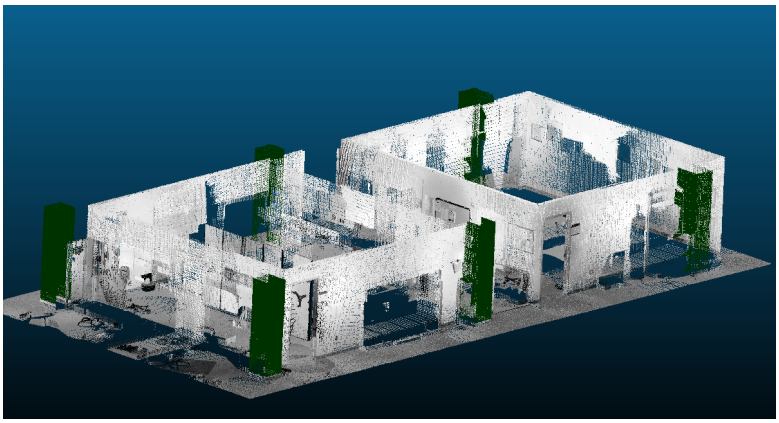
Image annotation



Point cloud annotation







Beam

HT ID	Name	IFC class	R	G	B	Hex color
15	beam	IfcBeam	0	170	0	00AA00

IfC entity definition

An IfcBeam is a horizontal, or nearly horizontal, structural member that is capable of withstanding load primarily by resisting bending. It represents such a member from an architectural point of view. It is not required to be load bearing. NOTE Definition according to ISO 6707-1: structural member for carrying load(s) between or beyond points of support, usually narrow in relation to its length and horizontal or nearly so.

Additional description

A steel girder will also be annotated as beam. Structural members consisting of multiple members (trusses, framework) will be annotated as truss.

Annotation rules

Beams may be connected with slabs or ceilings. Any vertical elements that do not extend between the floor and ceiling, but are connected to the ceiling, are annotated as beams.

Image annotation

Point cloud annotation

Truss

HT ID	Name	IFC class	R	G	B	Hex color
16	truss	IfcMember	150	250	50	96FA32

IfC entity definition

An IfcMember is a structural member designed to carry loads between or beyond points of support. It is not required to be load bearing. The orientation of the member (being horizontal, vertical or sloped) is not relevant to its definition (in contrary to IfcBeam and IfcColumn). An IfcMember represents a linear structural element from an architectural or structural modeling point of view and shall be used if it cannot be expressed more specifically as either an IfcBeam or an IfcColumn.

Additional description

IfcMember is a generic class and not specific to a truss, see description. A truss is a structural element, consisting of multiple members, which work together as one structural member. A truss may be a framework of steel or timber members.

Annotation rules

All appearances of truss will be annotated, also timber frameworks, etc.

Image annotation

Point cloud annotation

Chimney

HT ID	Name	IFC class	R	G	B	Hex color
17	chimney	IfcChimney	200	100	0	C86400

IfC entity definition

Chimneys are typically vertical, or as near as vertical, parts of the construction of a building and part of the building fabric. Often constructed by pre-cast or insitu concrete, today seldom by bricks. NOTE Definition according to ISO 6707-1: construction containing one or more flues. Flue: Duct designed to convey the products of combustion to the open air. Chimney stack: Part of the chimney that projects above a roof.

Additional description

Annotation rules

Image annotation

Point cloud annotation

Railing

HT ID	Name	IFC class	R	G	B	Hex color
18	railing	IfcRailing	140	40	30	8C281E

IfC entity definition

The railing is a frame assembly adjacent to human circulation spaces and at some space boundaries where it is used in lieu of walls or to compliment walls. Designed to aid humans, either as an optional physical support, or to prevent injury by falling.

Additional description

Annotation rules

Railings may consist of multiple members such as posts, handrails, glass or metal panes, etc. A parapet wall shall be annotated as wall.

Image annotation

Railings along stairs are annotated separately.

Point cloud annotation

Ramp

HT ID	Name	IFC class	R	G	B	Hex color
19	ramp	IfcRamp	250	150	150	FA9696

IfC entity definition

A ramp is a vertical passageway which provides a human circulation link between one floor level and another floor level at a different elevation. It may include a landing as an intermediate floor slab. A ramp normally does not include steps. NOTE Definition according to ISO 6707-1: Inclined way or floor joining two surfaces at different levels.

Additional description

Annotation rules

Image annotation

Point cloud annotation

Elevator

HT ID	Name	IFC class	R	G	B	Hex color
20	elevator	IfcTransportElement	200	150	250	C896FA

IfC entity definition

A transport element is a generalization of all transport related objects that move people, animals or goods within a building or building complex. The IfcTransportElement defines the occurrence of a trans-

port element, that (if given), is expressed by the IfcTransportElementType. EXAMPLE Transportation elements include elevator (lift), escalator, moving walkway, etc.

Additional description

An elevator is a technical installation that helps to transport persons and goods within a building vertically. An elevator consists of a cage, doors, guiderails, ropes, drive motor, brakes and electrical control components.

Annotation rules

All objects being part of the elevator are annotated as such. The structural walls of the elevator shaft, that are joint with the entire structure shall be annotated as wall. Glass coverings around an elevator are considered part of the elevator. Doors of elevators shall be annotated as elevator.

Image annotation

Point cloud annotation

Pavement

HT ID	Name	IFC class	R	G	B	Hex color
21	pavement	IfcPavement	50	200	250	32C8FA

IfC entity definition

see IFC 4x3 release candidate documentation[1] Type of built element in a road or other paved area to provide an even surface sustaining loads from vehicles or pedestrians, usually comprising several courses. NOTE Definition from ISO 6707-1: road, runway, or similar construction above the subgrade.

[1] <https://standards.buildingsmart.org/IFC/DEV/IFC4_3/RC2/HTML/schema/ifcsharedinfrastructureelements/lexical/ifcpavement.htm>

Additional description

Annotation rules

Image annotation

Point cloud annotation

Gravel pad

HT ID	Name	IFC class	R	G	B	Hex color
22	gravel pad	IfcGeographicElement	100	100	50	646432

IfC entity definition

An IfcGeographicElement is a generalization of all elements within a geographical landscape. It includes occurrences of typical geographical elements, often referred to as features, such as trees or terrain. Common type information behind several occurrences of IfcGeographicElement is provided by the IfcGeographicElementType.

Additional description

A gravel pad is an area on the outside of buildings that is covered with gravel. Usually gravel pads serve as an optical feature in landscaping along with vegetation and pavement areas.

Annotation rules

Curbstones at the perimeter of the gravel pad shall be annotated as gravel pad.

Image annotation

Point cloud annotation

—

Vegetation

HT ID	Name	IFC class	R	G	B	Hex color
23	vegetation	IfcGeographicElement	20	120	40	147828

IfC entity definition

An IfcGeographicElement is a generalization of all elements within a geographical landscape. It includes occurrences of typical geographical elements, often referred to as features, such as trees or terrain. Common type information behind several occurrences of IfcGeographicElement is provided by the IfcGeographicElementType.

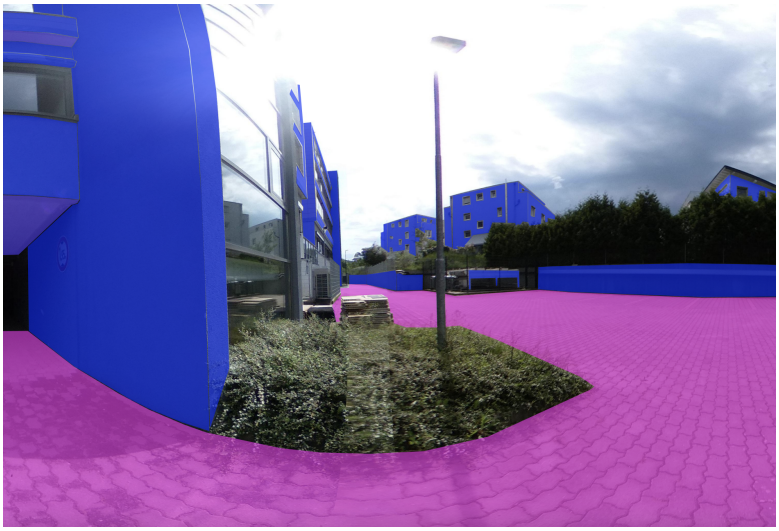
Additional description

All forms of vegetation such as trees, bushes, lawn, flower beds, ... including indoor plants shall be annotated as vegetation.

Annotation rules

Image annotation

Vegetation is not annotated in this example:



Point cloud annotation

—

Terrain

HT ID	Name	IFC class	R	G	B	Hex color
24	Terrain	IfcGeographicElement				

IfC entity definition

Tba

Space heater

HT ID	Name	IFC class	R	G	B
25	space heater	IfcSpaceHeater	230	240	10

IfC entity definition

Space heaters utilize a combination of radiation and/or natural convection using a heating source such as electricity, steam or hot water to heat a limited space or area. Examples of space heaters include radiators, convectors, baseboard and finned-tube heaters.

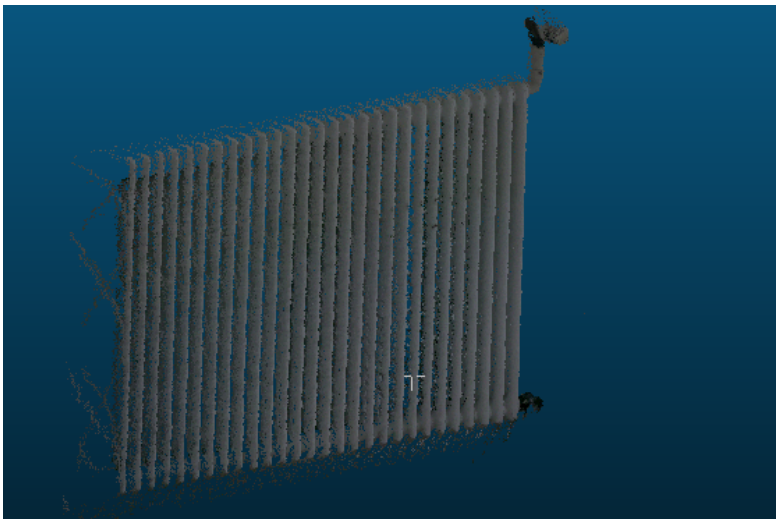
Additional description

Annotation rules

Space heaters will be annotated including controls, breathers and other fittings.

Image annotation

Point cloud annotation



Light Fixture

HT ID	Name	IFC class	R	G	B	Hex color
26	Light Fixture	IfcLightFixture	90	130	180	5a82b4

IfC entity definition

see IFC 4x3 release candidate documentation[2] A light fixture is a container that is designed for the purpose of housing one or more lamps and optionally devices that control, restrict or vary their emission.

Additional description

This label contains all light emitting objects fixed to the building (firmly mounted ceiling lights, ...).

Annotation rules

Movable light emitting objects (lamps) should be annotated as lamp

Image annotation

[2] <https://standards.buildingsmart.org/IFC/DEV/IFC4_3/RC2/HTML/link/ifclightfixture.htm>

Point cloud annotation

Curtain wall

HT ID	Name	IFC class	R	G	B	Hex color
27	curtain wall	IfcCurtainWall	225	0	250	E100FA

IfC entity definition

A curtain wall is an exterior wall of a building which is an assembly of components, hung from the edge of the floor/roof structure rather than bearing on a floor. Curtain wall is represented as a building element assembly and implemented as a subtype of IfcBuildingElement that uses an IfcRelAggregates relationship. NOTE Definition according to ISO 6707-1: non load bearing wall positioned on the outside of a building and enclosing it.

Additional description

Annotation rules

Image annotation

Point cloud annotation

Fence

HT ID	Name	IFC class	R	G	B	Hex color
28	fence	IfcBuildingElementProxy	250	180	110	FAB46E

IfC entity definition

The IfcBuildingElementProxy is a proxy definition that provides the same functionality as subtypes of IfcBuildingElement, but without having a predefined meaning of the special type of building element, it represents.

Proxies can also be used as spatial place holders or provisions, that are later replaced by special types of elements.

One use of the proxy object is a provision for voids, i.e. where a particular volume of space is requested by an engineering function that might later be accepted or rejected. If accepted it is transformed into a void within a building element, like a wall opening, or a slab opening. The provision for voids is exchanged as an IfcBuildingElementProxy with the PredefinedType = ProvisionForVoid. Such proxy shall have a swept solid geometry, where the profile of the swept solid lies on/near the surface of the referred building element and the extrusion depths is equal to or bigger then (in case of round or otherwise irregular element shape) the thickness of the building element. The appropriate property set should be attached.

In addition to the provision for voids, the building element proxy can also represent a provision for space, often the necessary space allocation for mechanical equipment that will be determined in a later design phase. The provision for space is exchanged as an IfcBuildingElementProxy with the PredefinedType = ProvisionForSpace.

Other usages of IfcBuildingElementProxy include:

The IfcBuildingElementProxy can be used to exchange special types of building elements for which the current specification does not yet provide a semantic definition. The IfcBuildingElementProxy can also be used to represent building elements for which the participating applications can not provide a semantic definition.

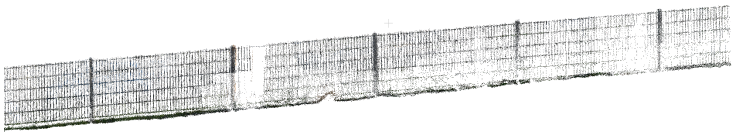
Additional description

A fence is a permanent installation to enclose a property. It may contain of posts and fence elements and from various materials.

Annotation rules

Image annotation

Point cloud annotation



Annex 5: CV4AEC dataset

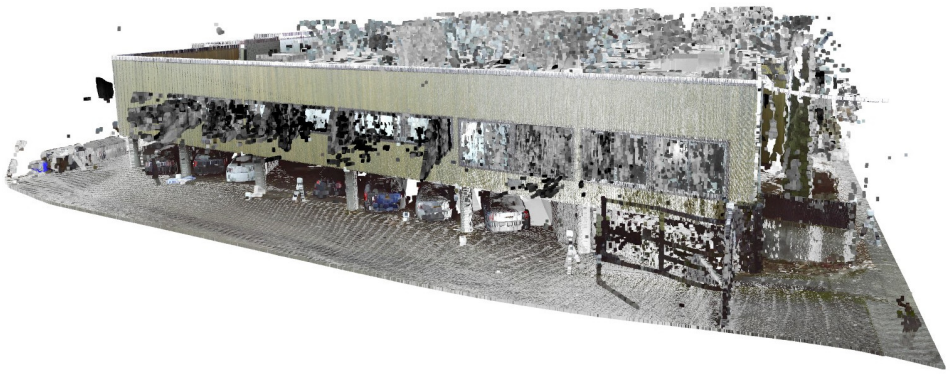


Figure A5.1: CV4AEC 01_OficeLab_01 scene.

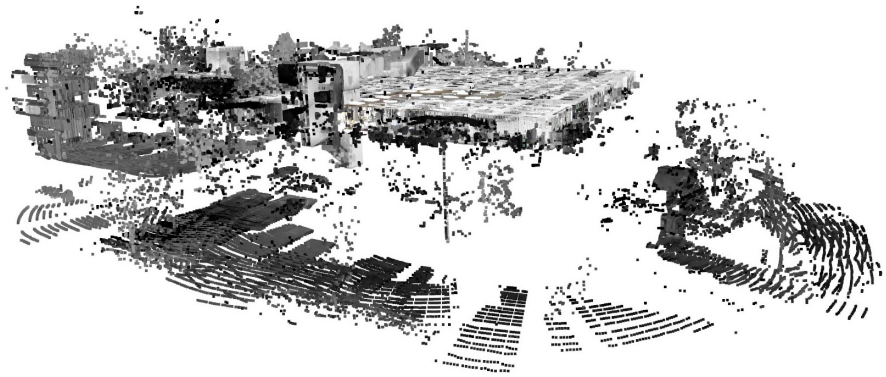


Figure A5.2: CV4AEC 05_MedOffice_01_F2 scene.

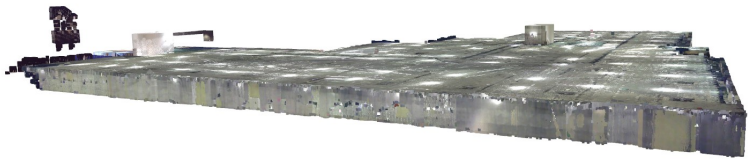


Figure A5.3: CV4AEC 06_MedOffice_02_B1 scene.

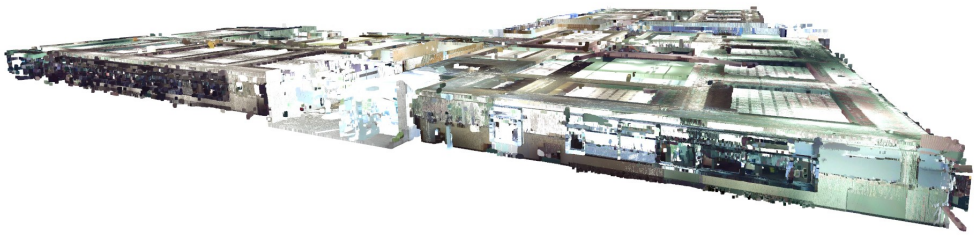


Figure A5.4: CV4AEC 06_MedOffice_02_F1 scene.

Figure A5.5: CV4AEC 06_MedOffice_02_F2 scene.

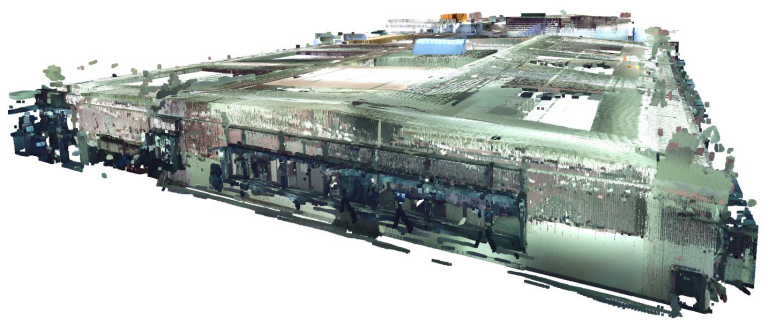


Figure A5.6: CV4AEC 06_MedOffice_02_F3 scene.

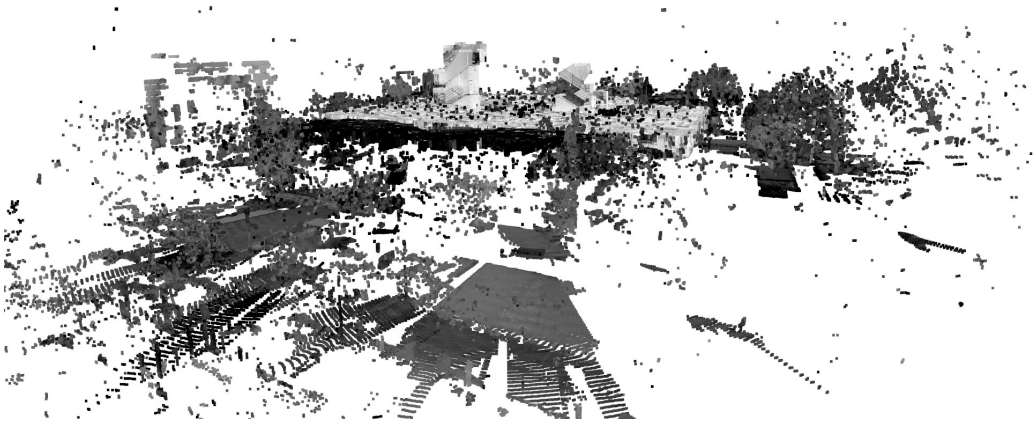


Figure A5.7: CV4AEC 07_MedOffice_03_F3 scene.

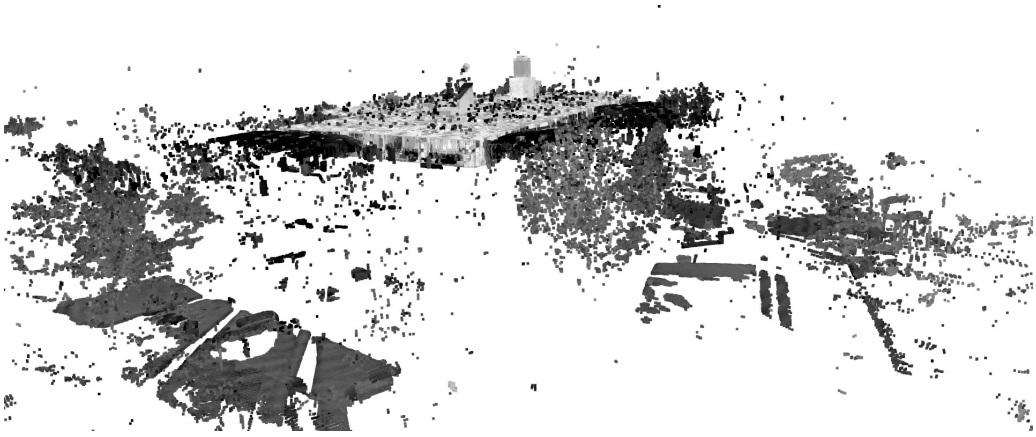


Figure A5.8: CV4AEC 07_MedOffice_03_F4 scene.

Figure A5.9: CV4AEC 07_MedOffice_03_F5 scene.

Figure A5.10: CV4AEC 08_ShortOffice_01_F1 scene.

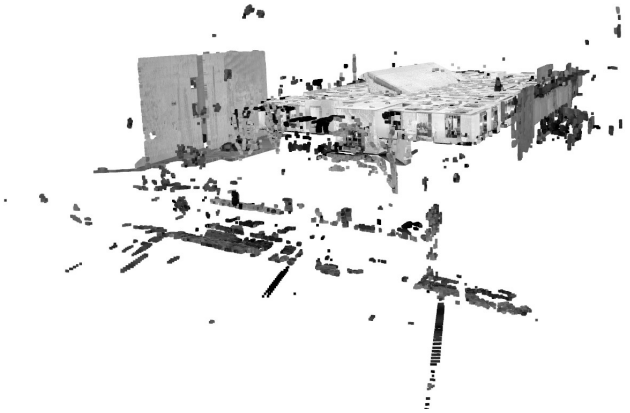


Figure A5.11: CV4AEC 08_ShortOffice_01_F2 scene.

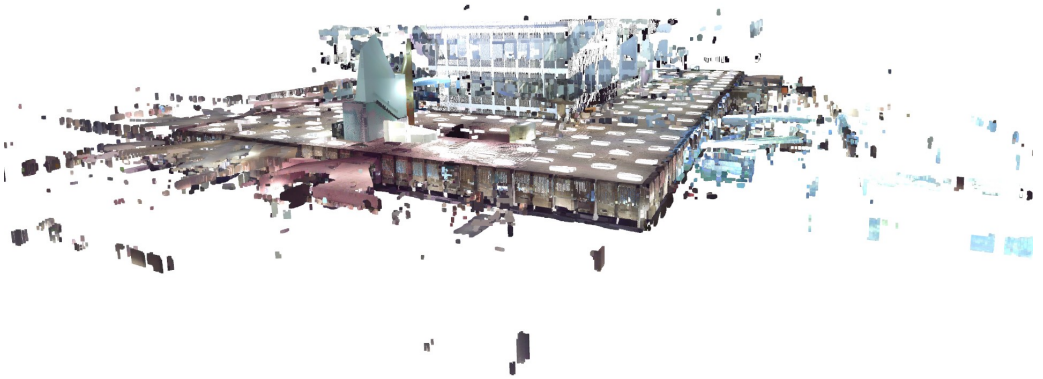


Figure A5.12: CV4AEC 11_MedOffice_05_F2 scene.

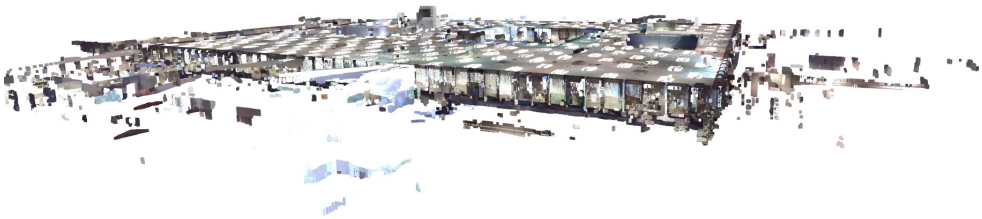


Figure A5.13: CV4AEC 11_MedOffice_05_F4 scene.

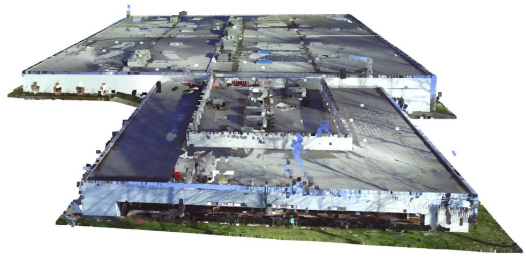


Figure A5.14: CV4AEC 16_Facility_01_F1 scene.

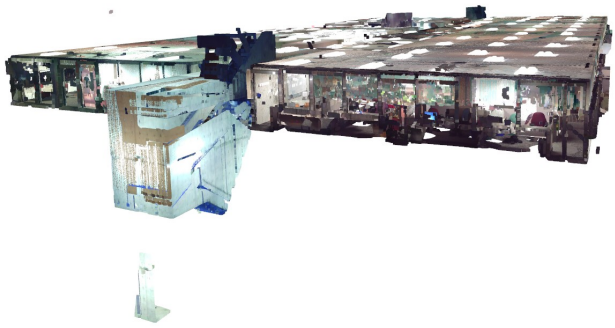


Figure A5.15: CV4AEC 19_MedOffice_07_F4 scene.

Figure A5.16: CV4AEC 22_Annex_01_F1 scene.

Figure A5.17: CV4AEC 25_Parking_01_F1 scene.

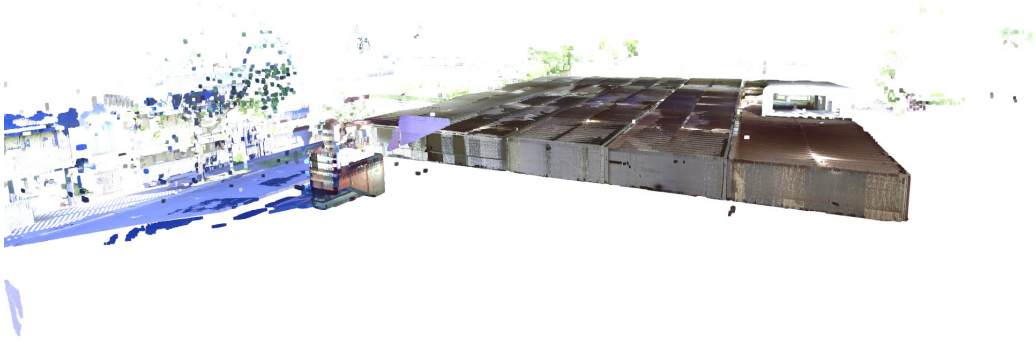


Figure A5.18: CV4AEC 25_Parking_01_F2 scene.

Figure A5.19: CV4AEC 32_ShortOffice_05_F1 scene.

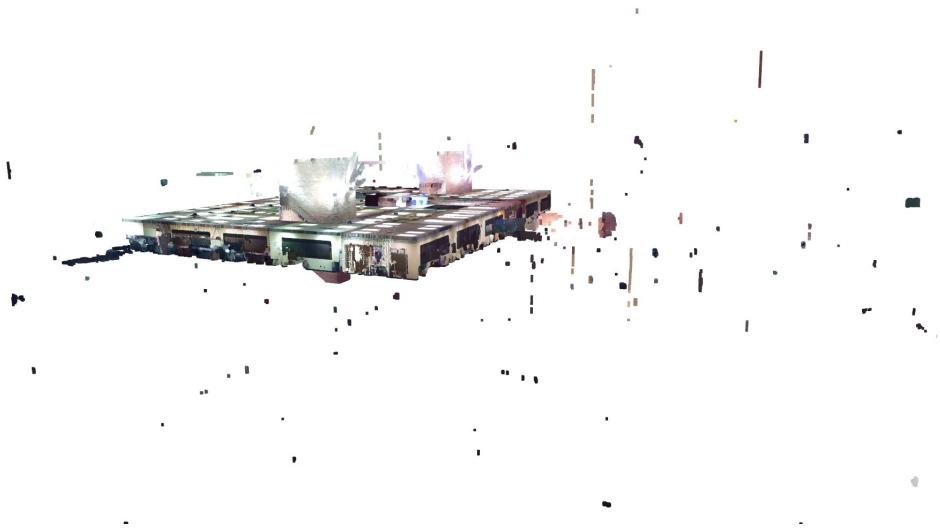


Figure A5.20: CV4AEC 32_ShortOffice_05_F2 scene.

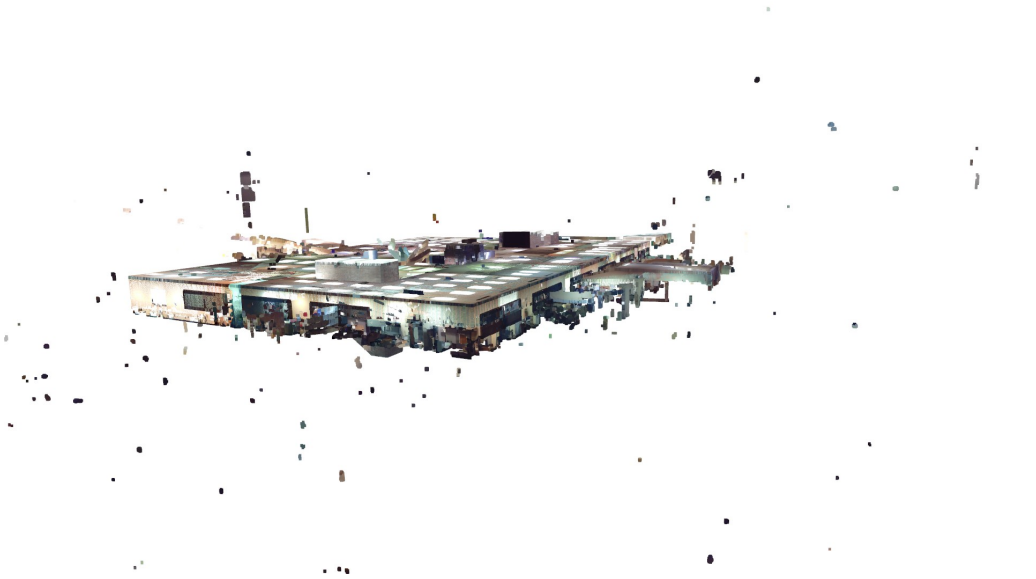


Figure A5.21: CV4AEC 32_ShortOffice_05_F3 scene.

Figure A5.22: CV4AEC 33_SmallBuilding_03_F1 scene.

Figure A5.23: CV4AEC 34_Parking_04_F1 scene.

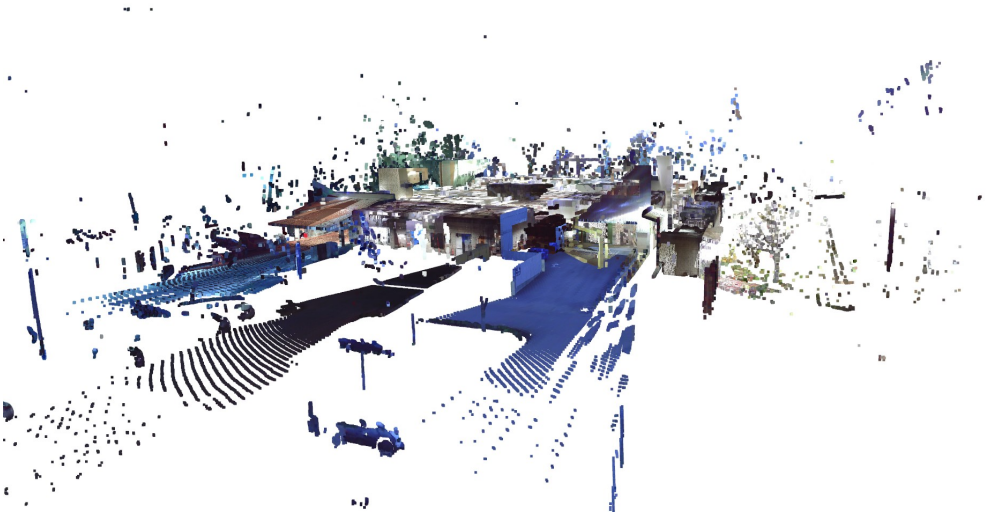


Figure A5.24: CV4AEC 35_Lab_02_F1 scene.

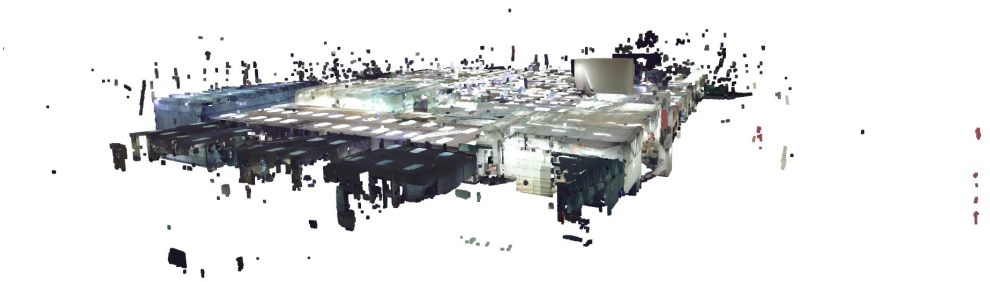
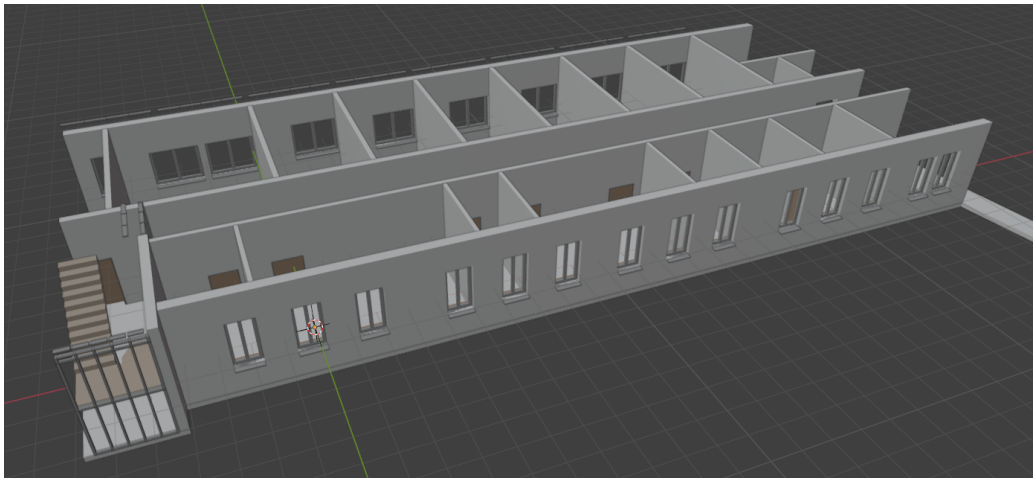


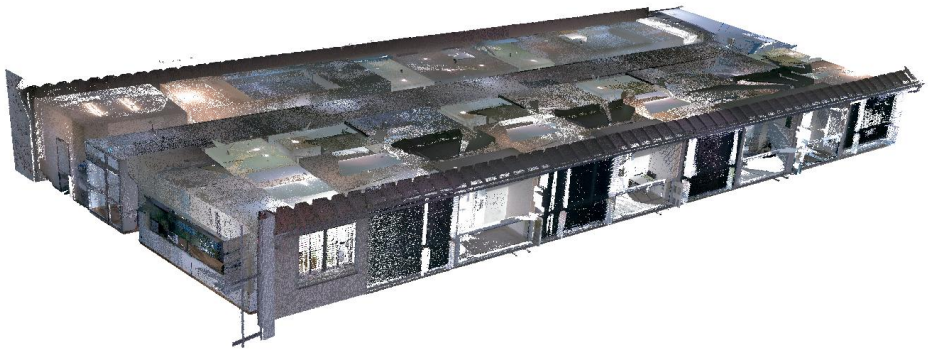
Figure A5.25: CV4AEC 35_Lab_02_F2 scene.

Annex 6: 14 NH dataset

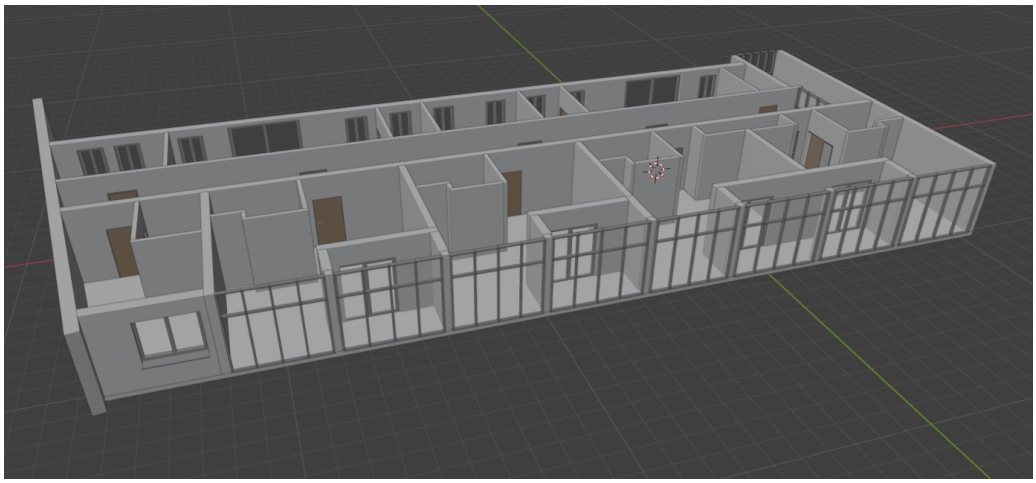
14NH A Building first floor: point cloud



14NH A Building first floor: manually created reference BIM model

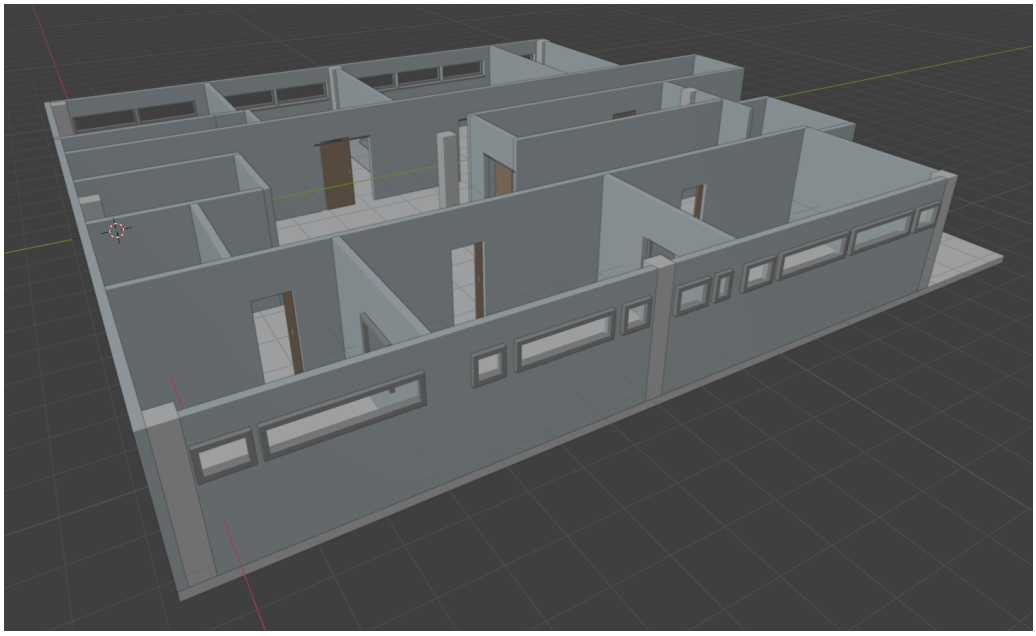


14NH A Building second floor: point cloud

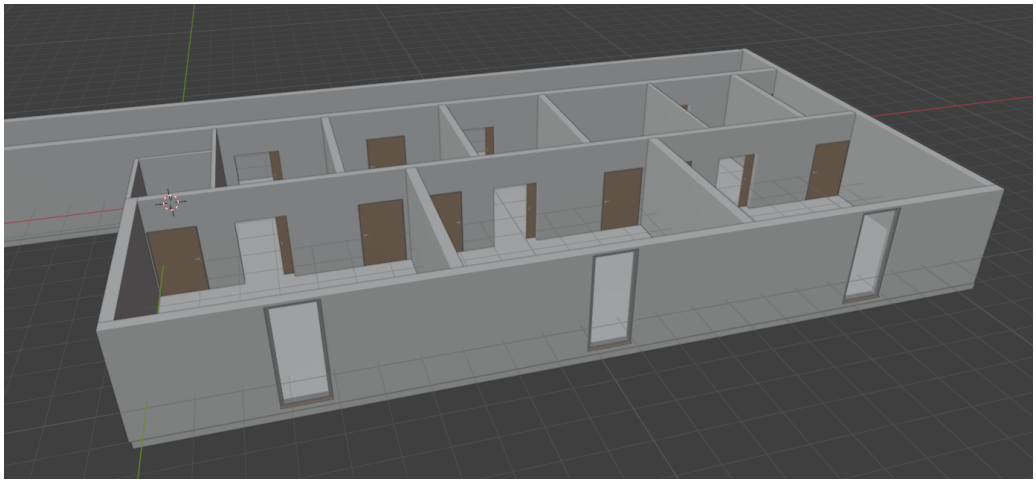


14NH A Building second floor: manually created reference BIM model

14NH B ICU: point cloud



14NH B ICU: manually created reference BIM model



14NH C surgery: manually created reference BIM model

Annex 7: Pipeline configuration

Parameter	Type	Explanation
valid_ids	List[int]	List of label IDs and objects to be processed. [4, 8, 14] refers to processing walls, doors and columns
eval_mode	String	Controls the mode of evaluation. Options are: *ifc_gt* using a ground truth IFC file provided, *cv4aec_gt* using the ground truth provided in the CV4AEC benchmark dataset in JSON format, *cv4aec_test* no evaluation, provide result in CV4AEC JSON format.
baseline	Int	Controls whether the *ScaleBIM* procedure is used or the baseline pipeline. Options are: 1 for baseline, 0 for *ScaleBIM*
debug	Int	Controls the debug mode, essentially the amount of visual and terminal output. Options are: 0 for headless mode, no visualization only terminal output. 1 Visualization of key steps e.g. all wall bounding boxes reconstructed. 2 Maximum visualization, even of intermediate steps.
density_ filtering	Int	Apply voxel density filtering to the input point cloud.
df_voxel_size	Float	Voxel size for density filtering. A voxel size too small can cause exhaustive memory consumption.
df_min_points	Int	Minimum number of floats in a voxel for density filtering.
translate_ min_z	Int	Translate all objects to minimum Z, i.e. to floor level elevation. Set 1 for translation, 0 for no translation.
translate_ max_z	Int	Translate all objects to maximum Z, i.e. to ceiling level elevation. Set 1 for translation, 0 for no translation.

Table A6.1: Pipeline configuration. Parameters for data preprocessing and general parameters for all reconstruction processes. The parameter name is written as in the configuration file.

Parameter	Type	Explanation
outside_ walls	Int	Apply outside walls reconstruction algorithm. SSet to 1 to enable the algorithm, or 0 to disable it.
wall_voxel_size	Float	Voxel size for wall points downsampling.
wall_normal_ threshold	Float	Threshold for axis sorting. The normals are normalized, so the value should be chosen between 0.0 and 1.0. With any value above 0.5, normals of the perpendicular direction will be sorted as well. About 0.02 seems to work fine for most data.
wall_dbscan_ eps	Float	ϵ, i.e. search radius for DBSCAN clustering. Must be set in conjunction with the minimum points parameter as both interact.
wall_dbscan_ min_points	Int	Minimum number of points in a cluster of a given radius ϵ.
wall_plane_ fitting	String	Algorithm for wall plane fitting. Either ”hysac” or ”ransac”.
hysac_min_ inlier_ratio	Float	Ratio of inliers over the input points for a plane to be accepted.
hysac_search_ space	Float	Perpendicular search offset of the histogram peak in both directions. The cleaner the data the lower the search space can be set.
hysac_ threshold	Float	Applied in the initial plane fitting using the seed points only. Threshold for initial plane to be accepted.
hysac_max_ error	Float	Maximum distance of all plane inlier points. If this is exceeded, the plane will not be accepted.
hysac_peak_ distance	Float	Relative vertical distance of peaks in the histogram for seed finding, i.e. a peak's height needs to be $(1 + 0.1) * neighbourbinheight$ to be accepted.
group_plane_ threshold	Float	Up to this limit, two planes will be grouped into one wall instance. Typically 0.4 to 0.6 depending on the type and age of the building.
one_plane_ handler_est_ thickness	Float	Constant wall thickness for wall instances reconstructed based on one plane.
wall_merge_ close	Int	Toggles whether the wall merging and closing algorithm should be applied. Set to 1 for closing and merging, and 0 for not topology refinement.
merge_walls_ threshold	Float	Maximum distance of wall endpoints to be merged. Should not exceed the expected minimum width of hallways and passages, typically 1 m.
wall_remove_ cubical	Int	Toggles if over segmented walls including cubicals should be removed. Set to 1 to enable the algorithm, or 0 to disable it.
wall_min_ volume	Float	Minimum volume of bounding boxes to be accepted as walls.
find_col_ suspects	Int	Find bounding boxes with a small volume and cuboid shape, which 'look like columns'. Set to 1 to enable the algorithm, or 0 to disable it.

Table A6.2: Pipeline configuration. Parameters for wall reconstruction. The parameter name is written as in the configuration file. Some parameters relate to distances or measurements in an Euclidean space, their unit is [m].

Parameter	Type	Explanation
hist_doors	Int	Apply inverse door reconstruction algorithm. Set to 1 to enable the algorithm, or 0 to disable it.
door_search_offset	Float	Wall bounding box will be extended by this offset to search for door points.
door_dbscan_eps	Float	ϵ, i.e. search radius for DBSCAN clustering. Must be set in conjunction with the minimum points parameter as both interact.
door_dbscan_min_points	Float	Minimum number of points in a cluster of a given radius ϵ.
close_doors	Int	Apply the closing doors algorithm. Set to 1 to enable the algorithm, or 0 to disable it.
close_door_voxel_size	Float	Voxel size for voxel downsampling of closed doors.
close_door_min_voxel_points	Int	Needs to be w. r. t. the previously set voxel size and the density of the data.
door_min_width	Float	Minimum opening width of bounding box. If set, all doors with smaller width will be extended to the minimum width.
door_split_width	Float	Door bounding boxes above this limit will be split into two.
door_fixed_width	Float	Transform all door bounding boxes to a fixed width. Set to -1 to not apply the procedure.

Table A6.3: Pipeline configuration. Parameters for door reconstruction. The parameter name is written as in the configuration file. Some parameters relate to distances or measurements in an Euclidean space, their unit is [m].

Parameter	Type	Explanation
col_voxel_size	Float	Voxel size for voxel downsampling.
col_sor_nb_neighbors	Int	Number of neighbours for statistical outlier removal.
col_sor_std_ratio	Float	Standard deviation for statistical outlier removal.
col_dbscan_eps	Float	ε, i.e. search radius for DBSCAN clustering. Must be set in conjunction with the minimum points parameter as both interact.
col_dbscan_min_points	Int	Minimum number of points in a cluster of a given radius ε.
col_max_volume	Float	Any column reconstructed with a volume above will be discarded.
col_transform_max	Int	Transform columns to fixed width and depth. Set to 1 to enable the procedure, or 0 to disable it.
col_max_width_depth	Float	Maximum width and depth the column will be clipped to if enabled (see above parameter).

Table A6.4: Pipeline configuration. Parameters for column reconstruction. The parameter name is written as in the configuration file. Some parameters relate to distances or measurements in an Euclidean space, their unit is [m].

Annex 8: 14 NH results

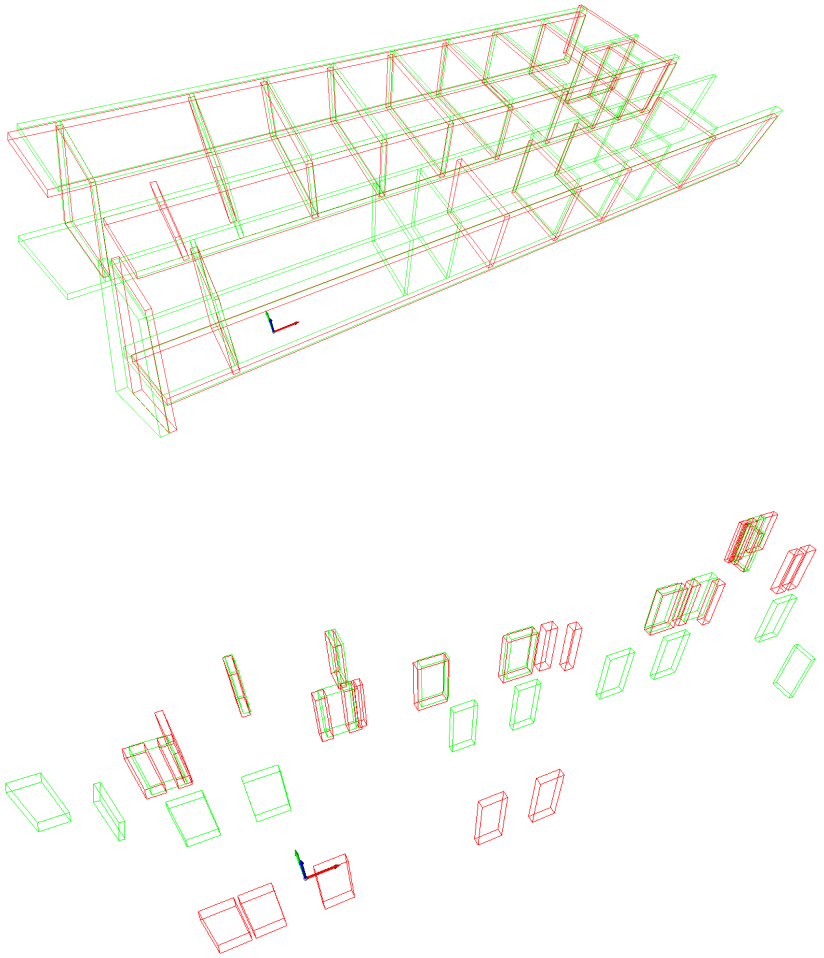


Figure A7.1: 14 NH A building first floor, from top to bottom: wall and door bounding boxes. No columns in the scene. Reconstructed bounding boxes red, ground truth green.

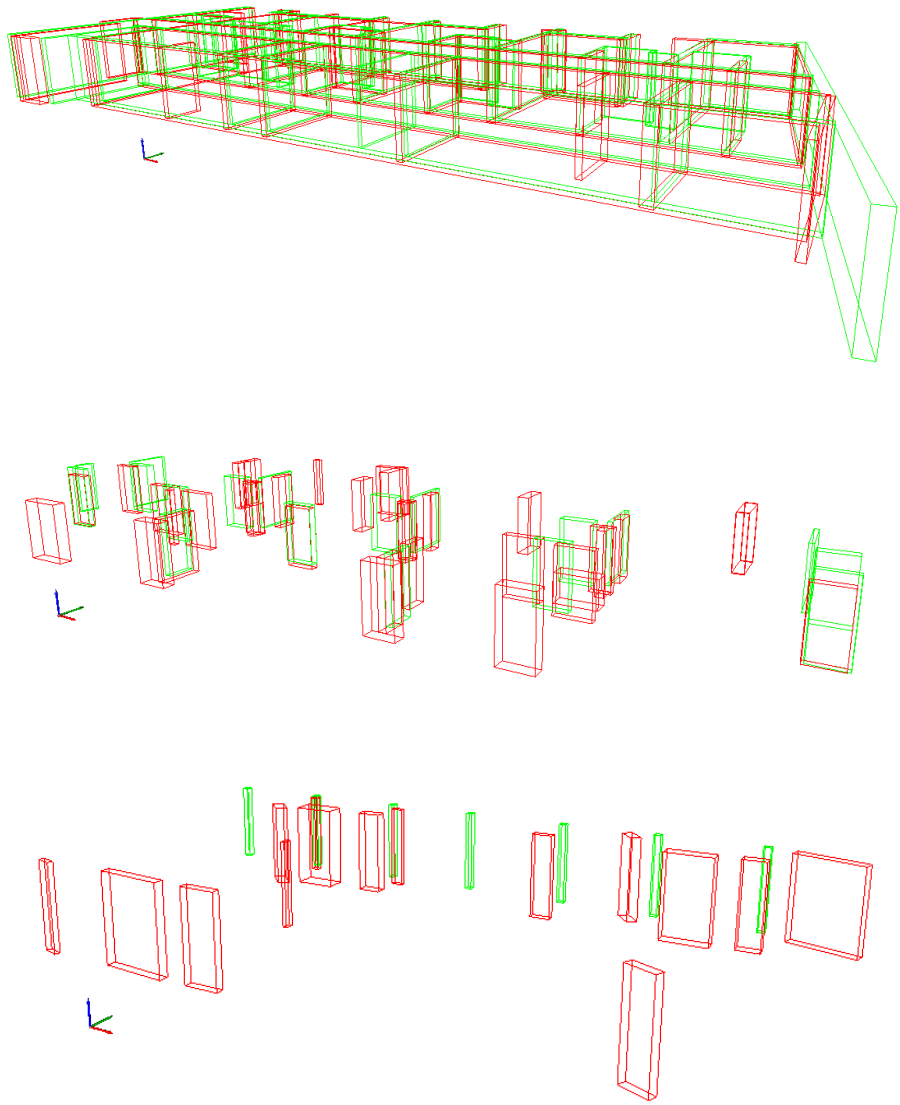


Figure A7.2: 14 NH A building second floor, from top to bottom: wall, door and column bounding boxes. Reconstructed bounding boxes red, ground truth green.

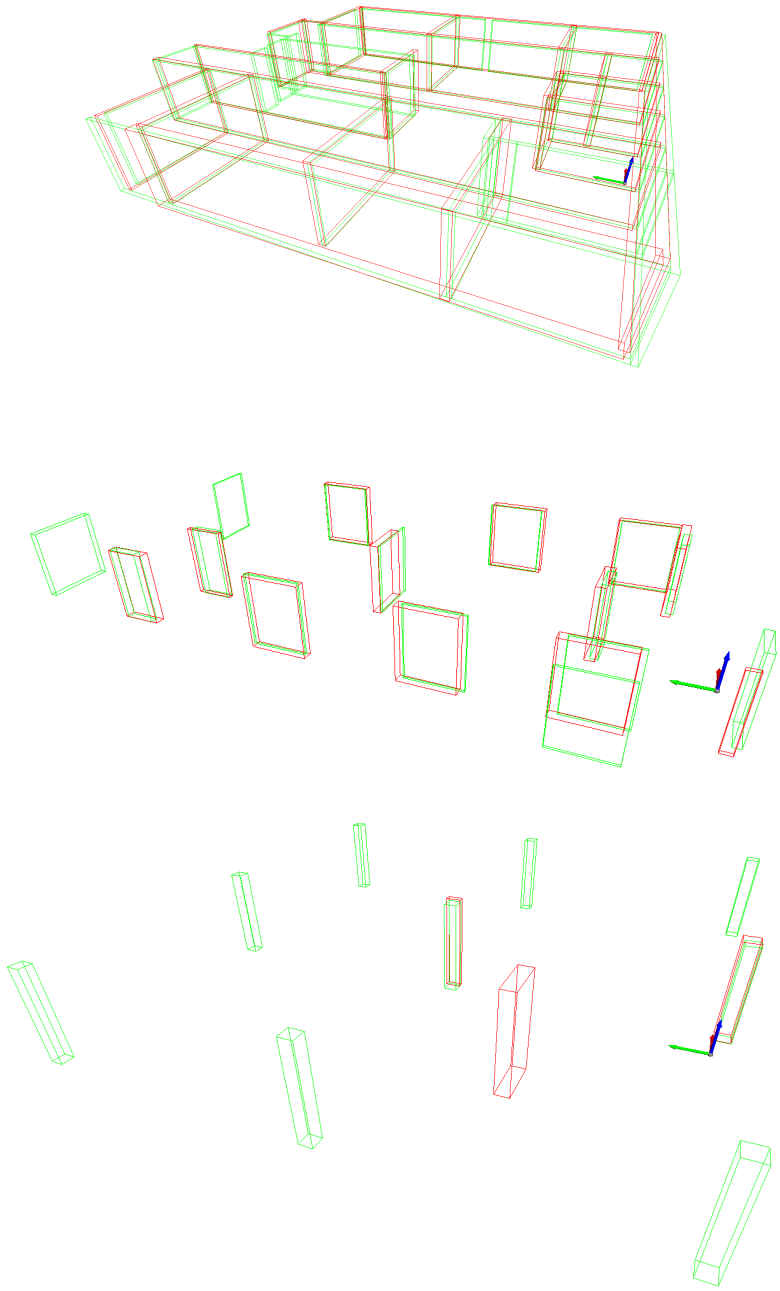


Figure A7.3: 14 NH B ICU, from top to bottom: wall, door and column bounding boxes. Reconstructed bounding boxes red, ground truth green.

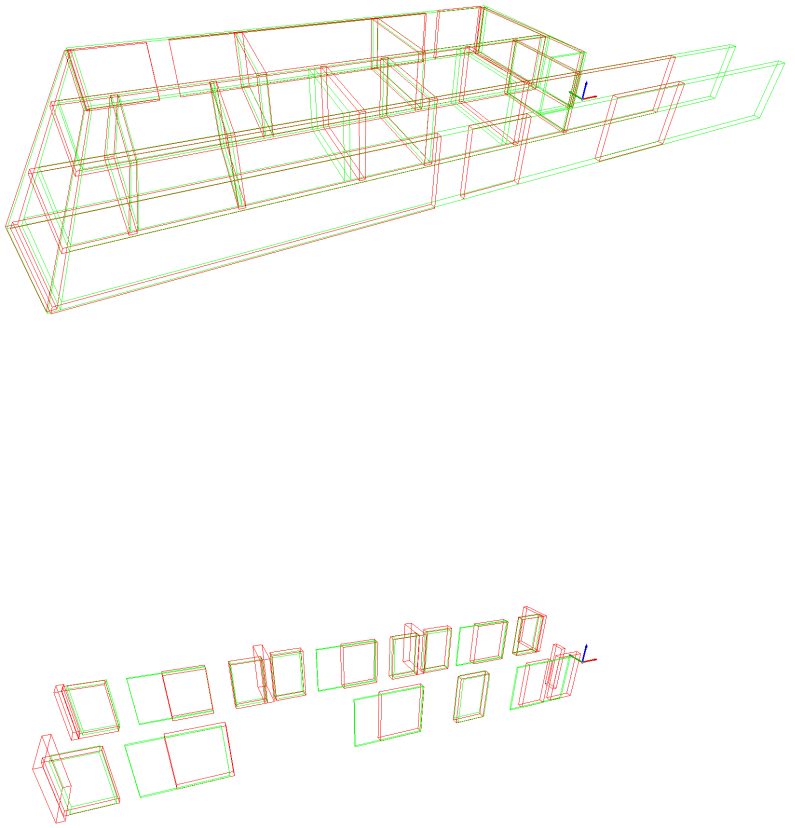


Figure A7.4: 14 NH C surgery, from top to bottom: wall and door bounding boxes. No columns in the scene. Reconstructed bounding boxes red, ground truth green.

Annex 9: CV4AEC results

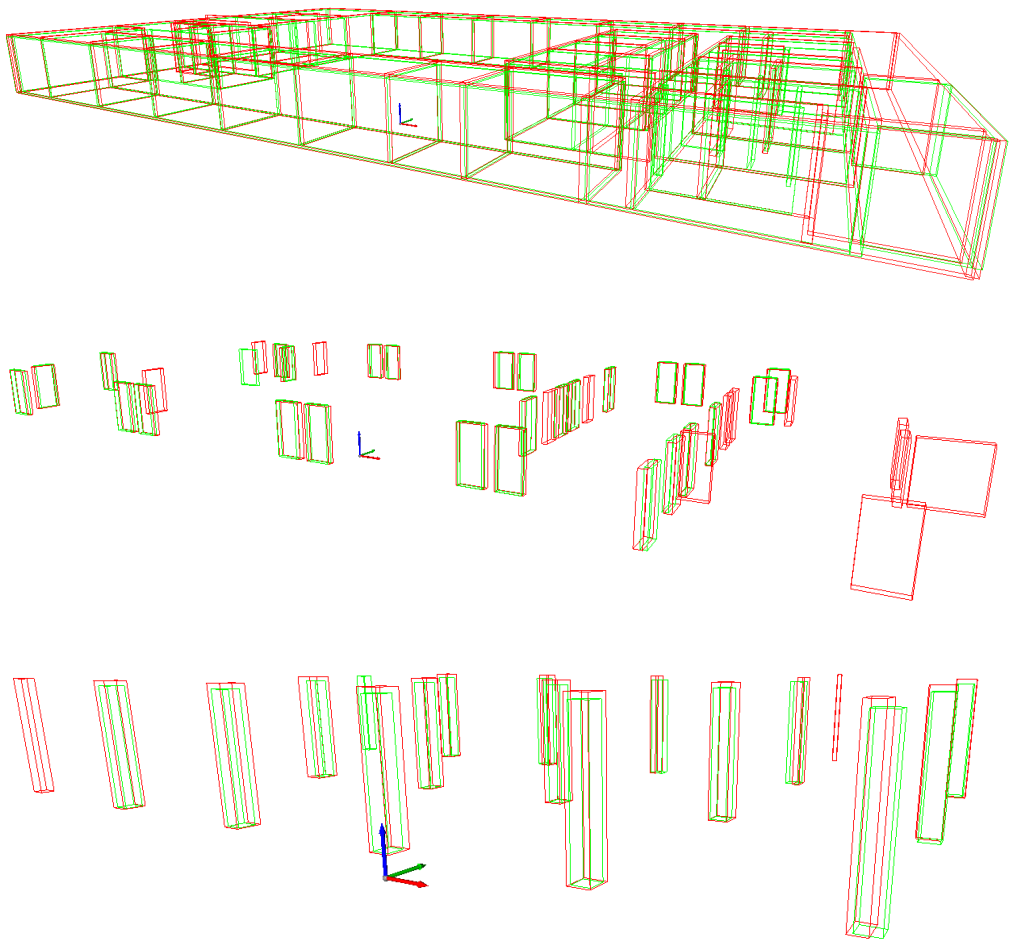


Figure A8.1: 08_ShortOffice_01_F1, from top to bottom: wall, door and column bounding boxes. Reconstructed bounding boxes red, ground truth green.

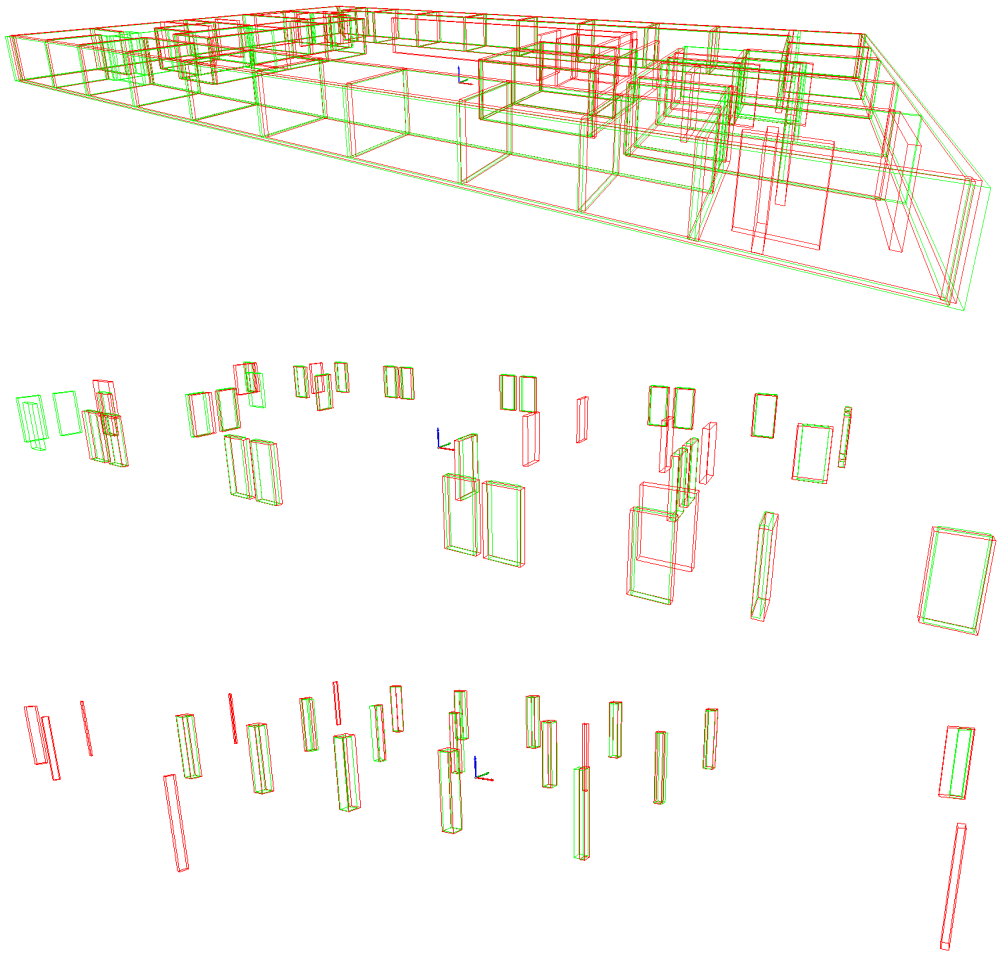


Figure A8.2: 08_ShortOffice_01_F2, from top to bottom: wall, door and column bounding boxes. Reconstructed bounding boxes red, ground truth green.

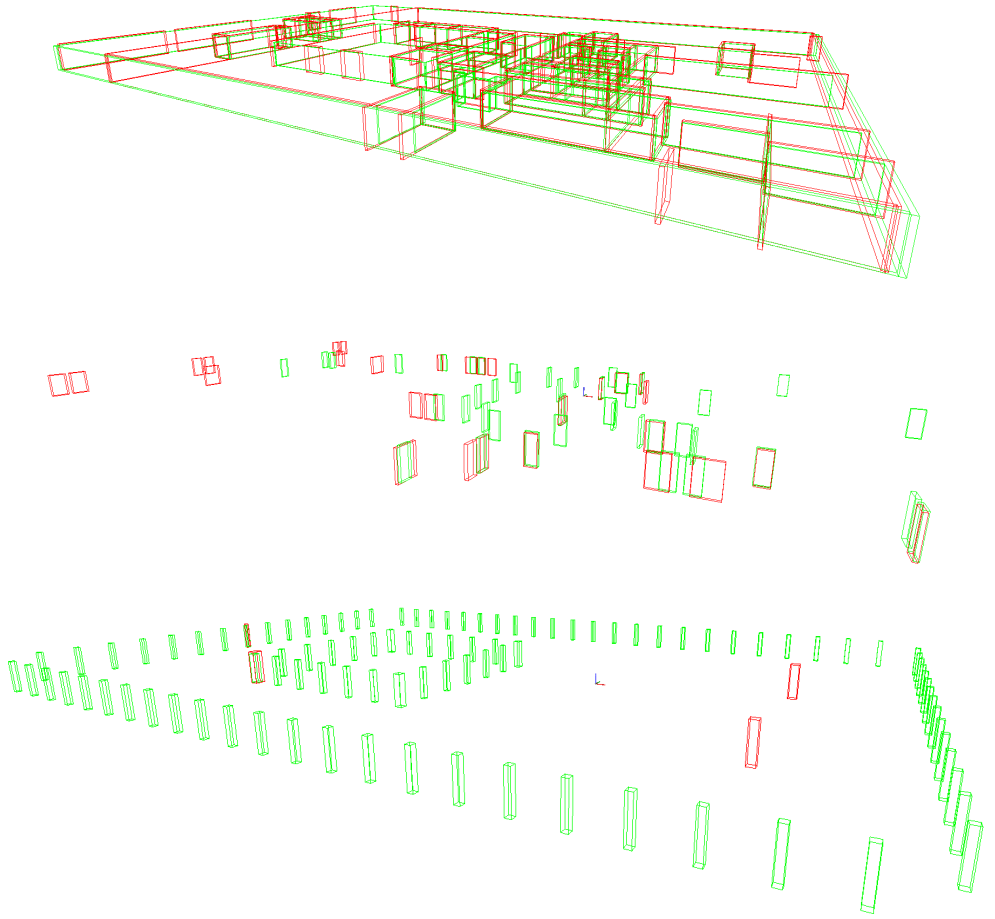


Figure A8.3: 11_MedOffice_05_F2, from top to bottom: wall, door and column bounding boxes. Reconstructed bounding boxes red, ground truth green.

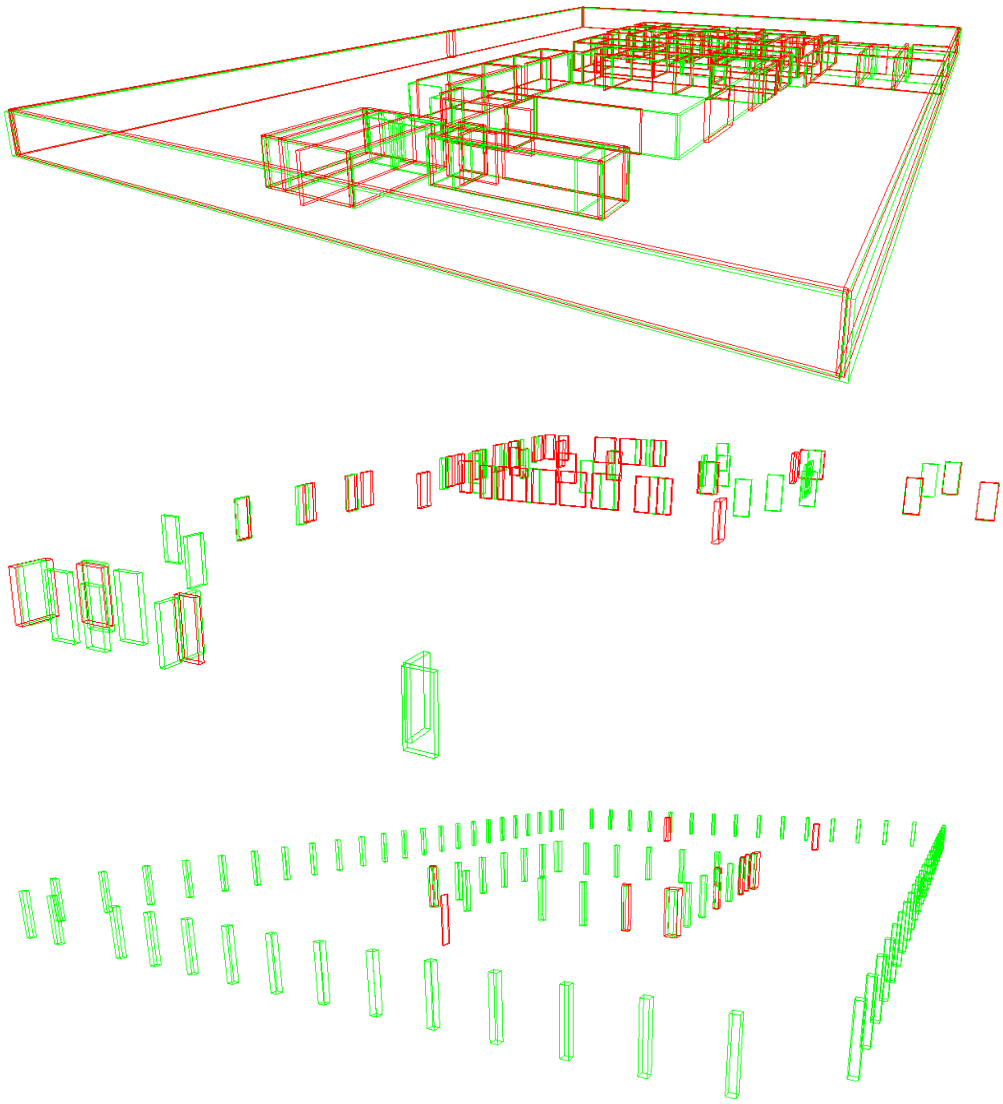


Figure A8.4: 11_MedOffice_05_F4, from top to bottom: wall, door and column bounding boxes. Reconstructed bounding boxes red, ground truth green.

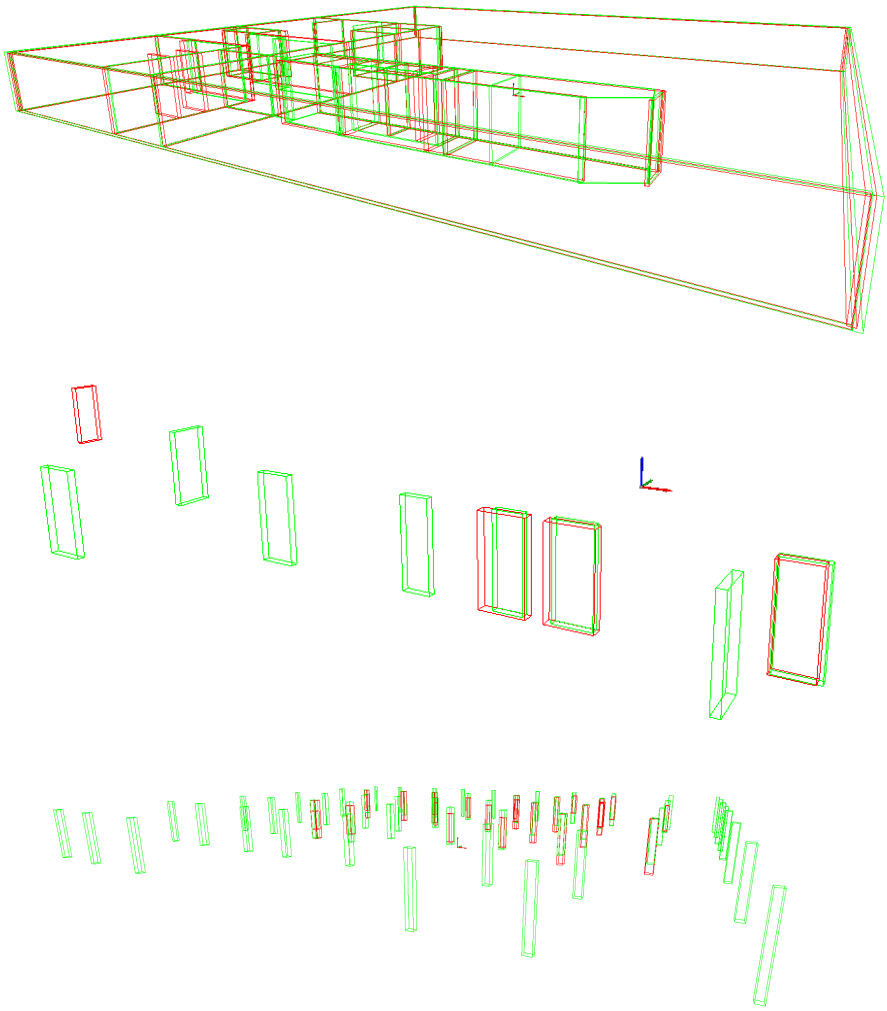


Figure A8.5: 25_Parking_01_F1, from top to bottom: wall, door and column bounding boxes. Reconstructed bounding boxes red, ground truth green.

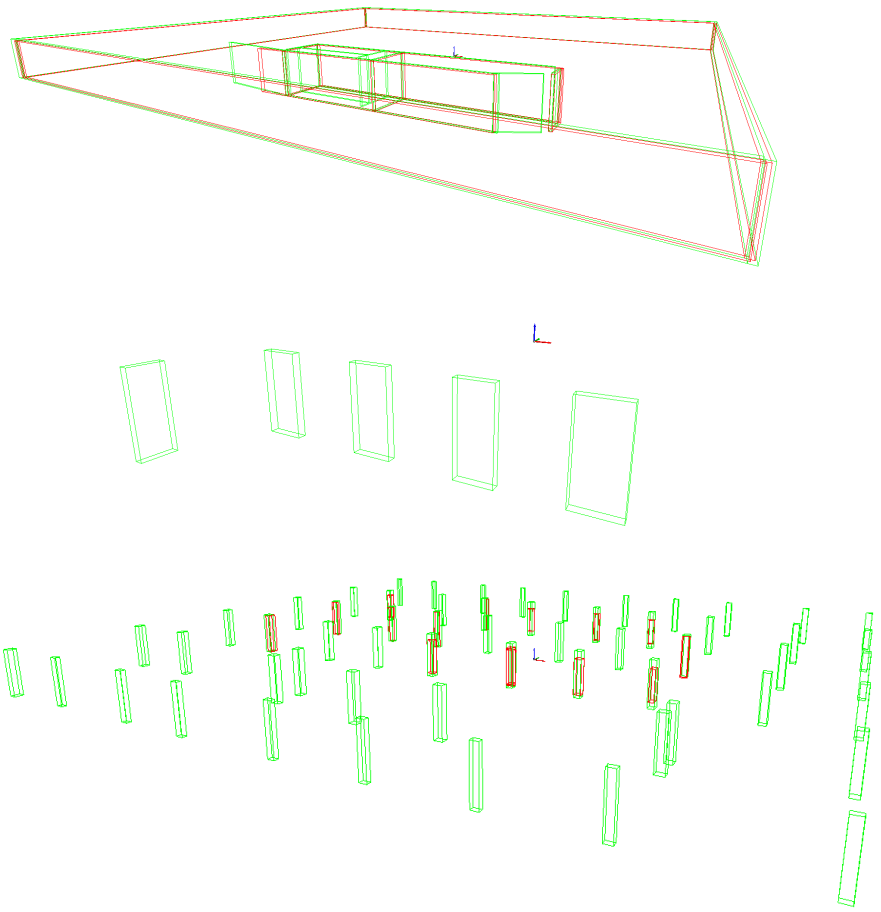


Figure A8.6: 25_Parking_01_F2, from top to bottom: wall, door and column bounding boxes. Reconstructed bounding boxes red, ground truth green.

Lebenslauf

Persönliche Daten

Name	Fabian Kaufmann
Staatsangehörigkeit	Deutsch

Berufserfahrung

2018 - heute	Wissenschaftlicher Mitarbeiter im Fachgebiet Massivbau und Baukonstruktion an der Rheinland-pfälzischen Technischen Universität Kaiserslautern-Landau (vormals Technische Universität Kaiserslautern)
2017 - 2018	Mitarbeiter im Bildungszentrum Holzbau, Biberach
2012 - 2013	Zimmerer bei Holzbau Kleinert, Wolfschlugen

Ausbildung

2016 - 2018	Projektmanagement(Bau), Master of Engineering, Hochschule Biberach
2013 - 2016	Projektmanagement/Bauingenieurwesen, Bachelor of Engineering, Hochschule Biberach
2009 - 2011	Ausbildung zum Zimmerer, Zimmerei Wager, Esslingen am Neckar und Holzbau Kleinert, Wolfschlugen
2009	Allgemeine Hochschulreife, Georgii-Gymnasium, Esslingen am Neckar