DIGITAL ENGINEERING FAKULTÄT
Fachgebiet Computergrafische Systeme

Digital Twins for Indoor Built Environments

Dissertation
zur Erlangung des akademischen Grades
"doctor rerum naturalium"
(Dr. rer. nat.)
in der Wissenschaftsdisziplin IT-Systems Engineering

Hasso-Plattner-Institut // Digital Engineering Fakultät // Universität Potsdam

vorgelegt von
Vladeta Stojanovic

Aufgabenstellung und Anleitung:
Prof. Dr. Jürgen Döllner

Potsdam,
27. September 2020

Published online on the
Publication Server of the University of Potsdam:
https://doi.org/10.25932/publishup-50913
https://nbn-resolving.org/urn:nbn:de:kobv:517-opus4-509134

Contents

Acknowledgments

I would like to thank my supervisor Prof. Dr. Jürgen Döllner for his input and support during my PhD studies. His precise and clear vision, as well as his renowned expertise in all topics related to computer graphics, served both as a solid instruction and as well as a delightful inspiration. I would also like to thank my colleagues and senior researchers at our Computer Graphics Systems chair – Dr Matthias Trapp, Dr Rico Richter, Dr Benjamin Hagedorn, and Dr Jan Klimke for their support regarding the direction of my research, and for their contributions towards key research publications.

I would also like to thank all my other colleagues at our chair for their support, camaraderie and helpfulness during my time as a PhD student. I enjoyed all of our serious meetings as much as I enjoyed our not-so-serious discussions in the communal kitchen (for my future work I also plan to finally learn how to use the main coffee machine).

I would like to extend my deepest gratitude to the *HPI Research School for Service-Oriented Systems Engineering* for enabling me to work towards the completion of my PhD studies, as well as providing funding for my stipend and conference travels. I would like to specifically thank Prof. Dr. Andreas Polze, Prof. Dr. Robert Hirchsfeld and Frau Sabine Wagner for supporting my involvement with the HPI Research School. I would also like to thank my peers from the HPI Research School for the invaluable professional and social experience I managed to acquire during the four years of my studies. Being given the opportunity to travel half the world with most of you is something that I will never forget, and will always remember fondly.

Finally, and most importantly, I would like to thank my parents, my grandmother and Maša for their constant support and encouragement – no matter where life takes me.

For Lana.

"Focused, hard work is the real key to success. Keep your eyes on the goal, and just keep taking the next step towards completing it. If you aren't sure which way to do something, do it both ways and see which works better." – John Carmack.

Abstract

One of the key challenges in modern Facility Management (FM) is to digitally reflect the current state of the built environment, referred to *as-is* or *as-built* versus *as-designed* representation. While the use of Building Information Modeling (BIM) can address the issue of digital representation, the generation and maintenance of BIM data requires a considerable amount of manual work and domain expertise. Another key challenge is being able to monitor the current state of the built environment, which is used to provide feedback and enhance decision making. The need for an integrated solution for all data associated with the operational life cycle of a building is becoming more pronounced as practices from Industry 4.0 are currently being evaluated and adopted for FM use. This research presents an approach for digital representation of indoor environments in their current state within the life cycle of a given building. Such an approach requires the fusion of various sources of digital data. The key to solving such a complex issue of digital data integration, processing and representation is with the use of a Digital Twin (DT). A DT is a digital duplicate of the physical environment, states, and processes. A DT fuses *as-designed* and *as-built* digital representations of built environment with *as-is* data, typically in the form of floorplans, point clouds and BIMs, with additional information layers pertaining to the current and predicted states of an indoor environment or complete building (e.g., sensor data). The design, implementation and initial testing of prototypical DT software services for indoor environments is presented and described. These DT software services are implemented within a service-oriented paradigm, and their feasibility is presented through functioning and tested key software components within prototypical Service-Oriented System (SOS) implementations. The main outcome of this research shows that key data related to the built environment can be semantically enriched and combined to enable digital representations of indoor environments, based on the concept of a DT. Furthermore, the outcomes of this research show that digital data, related to FM and Architecture, Construction, Engineering, Owner and Occupant (AECOO) activity, can be combined, analyzed and visualized in real-time using a service-oriented approach. This has great potential to benefit decision making related to Operation and Maintenance (O&M) procedures within the scope of the post-construction life cycle stages of typical office buildings.

Zusammenfassung

Eine der wichtigsten Herausforderungen im modernen Facility Management (FM) besteht darin, den aktuellen Zustand der gebauten Umgebung digital wiederzugeben und die tatsächliche mit der geplanten Gebäudedarstellung zu vergleichen. Während die Verwendung von Building Information Modeling (BIM) das Problem der digitalen Darstellung lösen kann, erfordert die Generierung und Pflege von BIM-Daten einen erheblichen manuellen Aufwand und Fachkenntnisse. Eine weitere wichtige Herausforderung besteht darin, den aktuellen Zustand der gebauten Umgebung zu überwachen, um Feedback zu geben und die Entscheidungsfindung zu verbessern. Die Notwendigkeit einer integrierten Lösung für alle Daten im Zusammenhang mit dem Betriebslebenszyklus eines Gebäudes wird immer deutlicher, da derzeit Praktiken aus Industrie 4.0 evaluiert und für die FM-Nutzung übernommen werden. Diese Studie präsentiert einen Ansatz zur digitalen Darstellung von Innenräumen in ihrem aktuellen Zustand innerhalb des Lebenszyklus eines bestimmten Gebäudes. Ein solcher Ansatz erfordert die Fusion verschiedener Quellen digitaler Daten. Der Schlüssel zur Lösung eines solch komplexen Problems der Integration, Verarbeitung und Darstellung digitaler Daten liegt in der Verwendung eines Digital Twin (DT). Ein DT ist ein digitales Duplikat der physischen Umgebung, Zustände und Prozesse. Ein DT verschmilzt die entworfenen und gebauten digitalen Darstellungen der gebauten Umwelt mit aktuellen Repräsentationsdaten, typischerweise in Form von Grundrissen, Punktwolken und BIMs, mit zusätzlichen Informationsebenen, die sich auf die aktuellen und vorhergesagten Zustände einer Innenumgebung oder eines kompletten Gebäudes beziehen (z.B. Sensordaten). Das Design, die Implementierung und die ersten Tests prototypischen DT-Software-Dienstleistungen für Innenräume werden vorgestellt und beschrieben. Die DT-Software-Dienstleistungen werden innerhalb eines serviceorientierten Paradigmas implementiert, und ihre Machbarkeit wird durch funktionierende und getestete wichtige Softwarekomponenten in prototypischen SOS-Implementierungen dargestellt. Das Hauptergebnis dieser Forschung zeigt, dass Schlüsseldaten in Bezug auf die gebaute Umgebung semantisch angereichert und kombiniert werden können, um digitale Darstellungen von Innenumgebungen basierend auf dem Konzept eines DT zu ermöglichen. Darüber hinaus zeigen die Ergebnisse dieser Forschung, dass digitale Daten in Bezug auf FM und Architektur, Bauwesen, Ingenieurwesen, Eigentümer- und Insassenaktivitäten mithilfe eines serviceorientierten Ansatzes in Echtzeit kombiniert, analysiert und visualisiert werden können. Dies hat ein großes Potenzial für die Entscheidungsfindung in Bezug auf Betriebs- und Wartungsverfahren im Rahmen der Lebenszyklusphasen typischer Bürogebäude nach dem Bau.

List of Relevant Acronyms

AABB	Axis-Aligned Bounding Box
AEC	Architecture, Engineering and Construction
AECOO	Architecture, Engineering, Construction, Owner and Occupant
AI	Artificial Intelligence
API	Application Programming Interface
AR	Augmented Reality
BAS	Building Automation System
BEM	Building Energy Modeling
BIM	Building Information Modeling
BMS	Building Management System
BPA	Ball-Pivoting Algorithm
B-Rep	Boundary Representation
BSP	Binary Space Partitioning
CAD	Computer-Aided Design
CAFM	Computer-Aided Facility Management
CCD	Charged-Coupled Device
CDE	Common Data Environment
CMMS	Computerized Maintenance Management System
CNN	Convolutional Neural Network
COBie	Construction Operations Building Information Exchange
CPU	Central Processing Unit
CQA	Construction Quality Assessment

CSG Constructive Solid Geometry

CSV Comma Separated Value

DBMS Database Management System

DBSCAN Density-based Spatial Clustering of Applications with Noise

DIM Dense Image Matching

DT Digital Twin

EMS Environmental Management Systems

ERP Enterprise Resource Planning

FEA Finite Element Analysis

FDM Finite Difference Mesh

FM Facility Management

FMIS Facility Management Information System

GIS Geographic Information System

GPS Global Positioning System

GPU Graphics Processing Unit

GUI Graphical User Interface

HTML Hyper Text Markup Language

HTTP Hypertext Transfer Protocol

ICP Iterative Closest Point

ICT Information and Communications Technology

IFC Industry Foundation Classes

IoT Internet-of-Things

IR Infrared

IT Information Technology

IWMS Integrated Workplace Management System

JPEG Joint Photographic Experts Group

JSON JavaScript Object Notation

KNN k-Nearest Neighbors

LiDAR Light Detection and Ranging

LOD Level-of-Detail

MEP Mechanical, Electrical and Plumbing

ML Machine Learning

NFC Near-field communication

NZEB Net Zero Energy Building

OGC Open Geospatial Consortium

O&M Operations and Maintenance

OOBB Object-Oriented Bounding Box

OpenGL Open Graphics Library

PCA Principal Component Analysis

PCCT Point Cloud Capture Technology

PCL Point Cloud Library

RAM Random Access Memory

RANSAC Random Sample Consensus

REST Representational State Transfer

RFID Radio-frequency identification

RGB Red-Green-Blue

RGBA Red-Green-Blue-Alpha

RGB-D Red-Green-Blue and Depth

ROI Return on Investment

ROR Radius Outlier Removal

RPS Random Point Selection

PSR Poisson Surface Reconstruction

SCADA Supervisory Control and Data Acquisition

SfM Structure from Motion

SL	Structured Light
SLS	Structured-Light Stereo
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SOR	Statistical Outlier Removal
SOS	Service-Oriented System
SQL	Structured Query Language
SVG	Scalable Vector Graphics
TCP	Transmission Control Protocol
TLS	Terrestrial Laser Scanner
ToF	Time-of-Flight
UAV	Unmanned Aerial Vehicle
URI	Uniform Resource Identifier
WebGL	Web Graphics Library
WFS	Web Feature Service
WSN	Wireless Sensor Network
XML	Extensible Markup Language

List of Figures

List of Tables

Chapter 1

Introduction

The modern world is becoming digitized and connected. From smart phones and tablets for communication on a daily basis, to self-driving autonomous vehicles, to smart health devices, there is currently a beginning of a convergence between the realms of the real and digital world. The built environment is no exception to this phenomena [6], and in the last two decades there have been great leaps in terms of digitizing its representation. In turn, this has proved to be beneficial by enabling stakeholders to inspect, assess and forecast the current and future states of the built environment, specifically related to its sustainability, and humans comfort, safety, productivity and well-being within it.

In recent years, the term Internet-of-Things (IoT) has been used to describe the concept of multiple connected devices all used to provide a link to real-time and historic data for specific applications [101, 282]. With advancements in computational resources, novel systems can combine digital representations of cities, specifically buildings where we live and work in, with IoT data. The concept of a *Smart City* [20], takes form by fusing various digital data sources, obtained from multiple connected digital devices (e.g., sensor data measuring natural and man-made phenomena), historical documentation and digitized representations [139].

When such data sources are combined, useful insights and results generated from visualization and analysis methods can contribute to enhanced decision making, and help create safer, more comfortable and more energy efficient working and living environments [35]. This also contributes greatly to the sustainability of the environment, human comfort and health, safety as well as providing greater return of investment for stakeholders. For example, city planners can assess various sustainability factors for a specific building or factory before it is given building permission – by analysing its relation to the surrounding environment within a digital representation system [28].

Such digital representation and analysis systems have historically been tied to Geographic Information System (GIS) software for large scale environment assessment and planning [63]. However, for finer analysis of individual buildings, greater detail and insight is required. In the last two decades Building Information Modeling (BIM) has been standardized and used to represent digital versions of buildings for complete life cycle analysis and management [67]. This includes the digital representation of buildings, including all related documentation pertaining to the design, planning, construction, operation and demolition phases of a building.

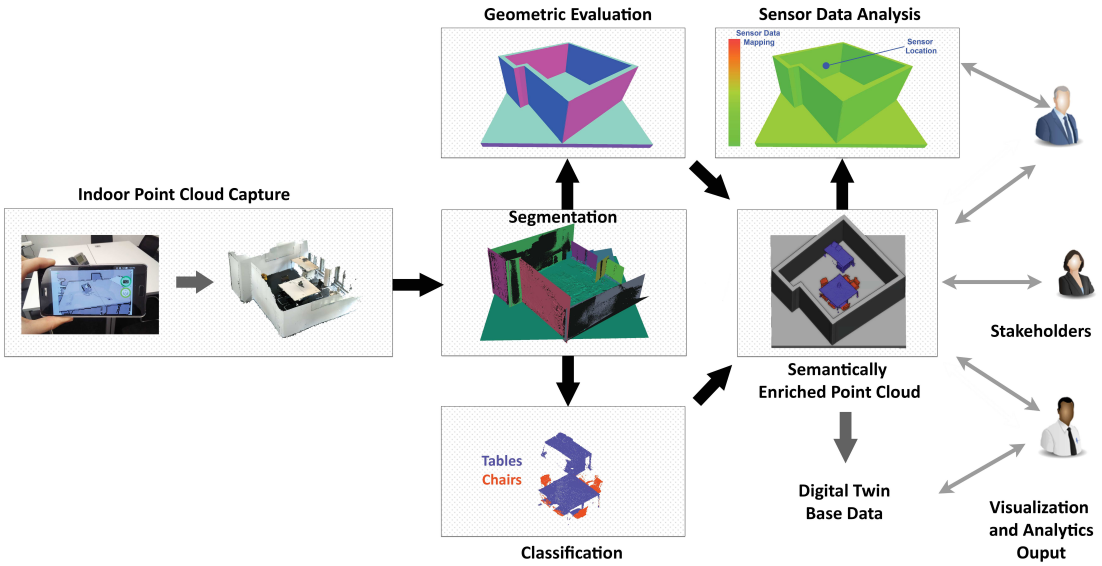


Figure 1.1: *High-level overview of semantic enrichment and use of indoor point clouds as base data for DT representations within the context of FM and Real Estate 4.0.*

A Digital Twin (DT) is a digital duplicate of the physical environment, states, and processes [97], and can be thought of an "umbrella term" for the concept of fusing related historic and current data sources together in an unified model. Thus a DT representation fuses *as-designed* and *as-is* physical representations, with additional information layers pertaining to the current and predicted states of an indoor environment or complete building. The use of a DT and its associated paradigm therefore plays a critical role in enabling the implementation of systems for smart buildings and built infrastructure [158].

Data sources used by a DT for analysis, visualization, prediction and enhancing decision making include digital data related to a given building or infrastructure object (e.g., bridges, roads, power plants, etc.). This type of data can be defined as either *historic* or *current* data. Historic data may include 2D and 3D Computer-Aided Design (CAD) drawings and models, as-designed or as-built BIM data, floorplans, historic sensor data and digitized documents [252]. Current data may include as-is BIM and other digital representations, and real-time sensor data. The fusion and analysis of these data sources by a DT software system enables the generation of useful analytics for enhanced decision making, particularly concerning everyday Operations and Maintenance (O&M) tasks within the domain of Facility Management (FM) [242]. Thus, it can be argued that the DT and is associated paradigm is one of the main driving forces (along with BIM and IoT), of the fourth technological revolution within the domain of property development, infrastructure and management i.e., *Real Estate 4.0* [274].

Since a DT relies on both historic and current data sources, the use of the later is more beneficial for up-to-date reflection, analysis and resulting decision making. One of the key current data sources are *point clouds*. Point clouds allow for capturing of the current state of the built environment, at varying scales and densities. This may include point clouds of entire cities generated from Light Detection and Ranging (LiDAR)

data [204], down to point cloud representations of small office spaces [236]. A DT can also be expended to different scales, and may include a *Geospatial* DT [60], which enables for the combination of data sources not just related to a single building or an object, but an entire geographical area e.g., a city [140].

1.1 Motivation

General Motivation The main motivation behind this research is to investigate, implement and evaluate key software components, based on the paradigm of a DT, for indoor environments and to demonstrate their feasibility for further use in the realm of Real Estate 4.0 (Fig. 1.1). For FM applications, especially O&M tasks, stakeholders need access to the current representation of the built environment, which in turn requires routine generation of an up-to-date BIM. Unfortunately, the generation of up-to-date digital representation of buildings, specifically using BIM, is an expensive, time consuming and cumbersome process [265].

The generation and maintenance of BIM data is also largely tied to monolithic software systems. The use of such software systems requires domain expertise, and they cannot easily be integrated into existing software platforms used for assessing the current and predicted operational state of a building. This includes the combination of additional data sources i.e., IoT data and existing digital documentation, and creates new problems concerning the analysis of such data in a way that is meaningful and insightful to FM stakeholders for decision making. The key to solving such a complex issue of digital data integration, processing and representation is with the use of a DT.

Indoor Point Cloud Capture and Processing Point clouds can be used as base-data for a DT, but for any meaningful representations and automated assessment point clouds need to be processed further in order to generate useful semantics. The use of point clouds poses two particular challenges: (1) Their capture, and (2) their processing required for generation of useful semantics. While point clouds can provide a *snapshot* of the physical environment, point cloud data itself does not feature any semantics by default. With recent hardware advances in consumer mobile devices, it is now possible to capture point clouds in an affordable and flexible manner [127], using phones and tablets with integrated LiDAR and *depth perception* hardware as part of on-board camera systems. With this paradigm shift, such hardware can be used by FM operators to capture point clouds of specific indoor areas without the need of using expensive LiDAR equipment.

Segmentation of captured point clouds is used to mark similar features of point clusters, which allows for quicker identification of particular objects (e.g., structural features, furniture in an office, cars on the street, etc.). Evaluations of shape contours derived from point clouds can further be used to generate 2D and 3D boundary approximation representations (e.g., computed geometric envelopes or floorplans). Additionally, data clustering methods can also be used to generate segmented 3D point clusters, and partitioning algorithms can be used to accelerate their spatial evaluation.

Point Cloud Classification Segmented point clusters need to be classified to generate understandable and usable semantics for further evaluation. Classification of point clouds typically uses *deep learning* to recognize 2D or 3D spatial and visual features [113]. Training data used to train a deep learning model (e.g., a Convolutional Neural Network (CNN)) can either be segmented 3D point cloud clusters, geometric approximations of point regions (e.g., voxels), or 2D images of a specific point cloud or 3D objects (or real-life photographs of their counterparts).

A flexible and practical method for classification of point clouds is generating multiple images of point clusters and classifying them using a 2D CNN. This is known as *multiview classification*. Generated multiview images can be classified in a service-oriented manner, and the classification results can be streamed back and associated with each partitioned cluster of the 3D point cloud. This allows for automated semantic enrichment of point clusters, and these generated semantics can be further used for reconstruction purposes, or to provide useful information to stakeholders.

Sensor Data Processing and Visualization Another important component of a DT representation is the use of sensor data. Sensors measuring building-related natural and man-made phenomena (e.g., temperature, humidity, electricity usage, carbon emissions, etc.), can provide information and insights about the current operational status of a building or office space within it. Visualization of such sensor data can in turn provide useful insights into the current operational status of a building. Additionally, processed sensor data can be visualized alongside a 3D representation of the building e.g., those captured by semantically enriched point clouds.

Interactive Visualization The complete DT representation can be visualized using interactive 2D and 3D computer graphics. The use of interactive visualization can benefit stakeholder engagement, information sharing and collaboration by allowing real-time display and analysis of 2D and 3D visual outputs generated from the captured data. Furthermore, the use of interactive visualization software libraries and frameworks based on Web3D standards encourages their implementation within a service-oriented paradigm.

Service-Oriented Architectures and Systems A Service-Oriented Architecture (SOA) is able to provide scalable integration of processing components, interfaces and data sources for varying application requirements, thus making it suitable for implementing a DT representation system. Additionally, an Service-Oriented System (SOS) implementation typically allows for decoupling of hardware requirements from client devices used by stakeholders.

1.2 Problem Statement

The original research presented in this thesis is used to form the answer to the fundamental research question: *"How can digital data representing the built environment be semantically enriched and combined in order to enable digital representations of such environments within a service-oriented paradigm, and based on the concept of a Digital Twin (DT)?"*

This original research presents methods and approaches for a DT-based representation and analysis of indoor environments, with a focus on FM-related applications. Specific challenges and problems addressed by this research include:

1. *How can indoor point clouds be processed and semantically enriched in order to make them useful as base-data for further FM-related analysis and applications?*
2. *How can as-is point clouds be compared against as-built and as-designed BIM geometry in order to highlight any spatial deviations?*
3. *How can indoor sensor data be processed and visualized alongside an indoor point cloud?*
4. *How can data related to the the digital representation of indoor environments be accessed, processed, analyzed and visualized within a service-oriented paradigm?*

1.3 Research Contributions

The outcome of this research demonstrates the feasibility of key prototypical software components and services implementations for digitized FM within a SOS paradigm, and in turn proves the feasibility of a future DT platform for FM applications within the realm of Real Estate 4.0. This has the potential to aid in important decision making related to O&M procedures within the scope of the post-construction life cycle stages of typical office buildings. Additionally, the following research contributions are presented in this thesis:

1. A conceptual SOA for a FM-oriented DT platform, which was used as a reference for implementing and testing key prototypical SOS implementations and software components.
2. Semantic enrichment of indoor point clouds using deep-learning.
3. Generation and spatial analysis of corresponding geometry obtained from point cloud data.
4. Processing and visualization of sensor data for indoor point clouds.
5. Processing and visualization of indoor point cloud data.

Specifically, the following software components and systems have been designed, implemented and tested as part of the original research contributions for this thesis:

- A service oriented point cloud processing pipeline (Chpt. 3).
- A software component used for generating 2D and 3D floorplan approximations from 3D point clouds (Chpt. 5).
- A deviation analysis software component capable of highlighting spatial differences between an *as-is* point cloud and corresponding *as-designed* or *as-built* BIM geometry (Chpt. 6).
- A classification software component capable of classifying 3D point clusters using both multiview and object-based classification (Chpt. 7).
- A sensor data processing software component used for parsing spatio-temporal data, alongside a point cloud, for combined visual analytics (Chpt. 8).

Key Publications The key research contributions described in this thesis have previously been published in the following 11 peer-reviewed journal and conference papers:

1. **Stojanovic, V.**, Richter, R., Döllner, J., and Trapp, M. (2018). "Comparative visualization of BIM geometry and corresponding point clouds". In: *International Journal of Sustainable Development and Planning*, 13, 1, pp. 12-23.
2. **Stojanovic, V.**, Trapp, M., Richter, R., and Döllner, J. (2018). "A service-oriented approach for classifying 3D points clouds by example of office furniture classification". In: *Proceedings of the 23$_{rd}$ International ACM Conference on 3D Web Technology (Web3D '18)*, p. 9.
3. **Stojanovic, V.**, Trapp, M., Richter, R., Hagedorn, B., and Döllner, J. (2018). "Towards the Generation of Digital Twins for Facility Management Based on 3D Point Clouds". In: *Gorse, C and Neilson, C J (Eds.), Proceedings 34$_{th}$ Annual ARCOM Conference*. Association of Researchers in Construction Management, pp.270–279.
4. **Stojanovic, V.**, Trapp, M., Richter, R., and Döllner, J. (2019). "Generation of Approximate 2D and 3D Floor Plans from 3D Point Clouds". In: *Proceedings of the 14$_{th}$ International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 1: GRAPP*, pp. 177-184.
5. **Stojanovic, V.**, Trapp, M., Richter, R., and Döllner, J. (2019). "Classification of Indoor Point Clouds Using Multiviews". In: *Web3D '19: The 24th International Conference on 3D Web Technology (Web3D '19)*, p.9.

6. **Stojanovic, V.**, Trapp, M., Richter, R., Hagedorn, B., and Döllner, J. (2019). "Semantic Enrichment of Indoor Point Clouds: An Overview of Progress towards Digital Twinning". In: *Architecture in the Age of the 4_{th} Industrial Revolution - Proceedings of the 37_{th} eCAADe and 23rd SIGraDi Conference - Volume 2.* pp. 809-818.

7. **Stojanovic, V.**, Trapp, M., Richter, R., and Döllner, J. (2019). "Service-Oriented Semantic Enrichment of Indoor Point Clouds using Multiview-Based Classification". In: *Graphical Models.* Elsevier.

8. **Stojanovic, V.**, Trapp, M., Hagedorn, B., Klimke, J., Richter, R., and Döllner, J. (2019). "Sensor Data Visualization for Indoor Point Clouds". In: *Advances in Cartography and GIScience of the ICA*, 2.

9. **Stojanovic, V.**, Trapp, M., Richter, R., and Döllner, J. (2019). "A Service-oriented Indoor Point Cloud Processing Pipeline". In: *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 42, pp.339-346.

10. Isailović, D., **Stojanovic, V.**, Trapp, M., Richter, R., Hajdin, R., and Döllner, J. (2020). "Bridge Damage: Detection, IFC-Based Semantic Enrichment and Visualization". In: *Automation in Construction.* Elsevier.

11. **Stojanovic, V.**, Trapp, M., Richter, R., and Döllner, J. (2020). "Comparison of Deep-Learning Classification Approaches for Indoor Point Clouds". In: *Publikationen der DGPF, Band 29, 2020.* pp.437-447.

Additionally, one other published paper describes a conceptual approach to the formulation of the answer to one of the potential future research questions(Chpt. 10.3 - future work question two): **Stojanovic, V.**, Hagedorn, B., Trapp, M., and Döllner, J. (2020). "Ontology-Driven Analytics for Indoor Point Clouds". In: *RE: Anthropocene, Design in the Age of Humans - Proceedings of the 25_{th} CAADRIA Conference - Volume 2.* pp.537-546.

1.4 Structure

The remaining nine chapters of this thesis are structured as follows:

Chapter 2, *Foundations and Related Work* Introduction to key ideas and concepts that influenced the formation of the research problem as well as an overview of all related work.

Chapter 3, *Service-Oriented Architecture and Systems* Presentation and discussion of a conceptual reference SOA used for the prototypical implementation of key software components, along with system designs for each of the corresponding prototypical SOS implementations.

Chapter 4, *Capture and Processing Indoor Point Clouds using Commodity Mobile Devices* Description of key approaches and methods used for capturing, processing and visualizing indoor point clouds – with a specific focus on indoor point clouds captured using commodity mobile devices.

Chapter 5, *Point Cloud Reconstruction* Overview and description of methods used for geometric reconstruction of point clouds as well as approximate floorplan generation.

Chapter 6, *Spatial Deviation Analysis* Description of the spatial deviation analysis approach used to highlight spatial differences between point clouds and corresponding BIM geometry.

Chapter 7, *Semantic Enrichment of Indoor Point Clouds* Methods and approaches for classification and semantic enrichment of indoor point clouds, with a specific focus on deep learning-based approaches.

Chapter 8, *Processing and Visualization of Indoor Sensor Data* Methods and approaches for processing and visualization of indoor sensor data, with a focus on combined visualization using point clouds.

Chapter 9, *Discussion* Presentation of the answer to the key research question, and the answers to the related research questions. A general discussion of approaches, case studies and outcomes is also presented.

Chapter 10, *Conclusions and Outlook* Concluding discussion of the presented original research based on the initial research question and motivation, summary of key findings and outline of future research topics.

Chapter 2

Foundations and Related Work

This section provides a detailed overview of core foundations and related work used to address the stated research problem (Chpt. 1.2). The foundations behind the approaches and methods presented in this thesis are strongly related to software engineering, particularity computer graphics systems, service-oriented computing and Artificial Intelligence (AI) (including the subset fields of machine and deep learning). The areas of application are related to Architecture, Engineering, Construction, Owner and Occupant (AECOO) and FM domains.

2.1 Overview of Digital Representations of Built Environments

The digital representation of the built environment was historically restricted to 2D CAD drawings and static rendered images of 3D models, or "walk-through" animated videos. Along with various advancements in real-time 3D rendering at the beginning of the 21st century, it became possible to visualize complete 3D representations of buildings using various interactive 3D software and game engines [225]. Such software allowed stakeholders (e.g., architects, city planners, draftsman, etc.), to inspect various regions of the 3D model representation of a building, and to ultimately aid in decision making concerning the design, construction, and intended use of a building [107].

However, static 3D models and 2D CAD drawings can only show the building at a particular stage during its life cycle, and cannot be used to detect any spatial or temporal changes that are crucial for examining various important factors e.g., construction and maintenance times, building status and operational costs. This sort of multi-dimensional analysis requires the use of a more generalized model that can encompass the digital representation of the building at various stages during its life cycle, and at a varying Level-of-Detail (LOD), while factoring in additional variables e.g., time and money [230]. The concept of a BIM was created to address this issue.

According to Eastman *et al.* [67] BIM encompasses all of the key representational, analytical and documentation properties in order to enable information sharing between stakeholders involved in the life cycle of a building. The sharing of information and multidimensional analysis enables stakeholders to check for specific *clashes* between different building elements within a Common Data Environment (CDE) (e.g., structural components, electrical fittings, etc.) [117]. The use of BIM for engaging stakeholders who may not have domain-specific expertise in Architecture, Engineering and Construction

(AEC) also provides a distinct advantage, as simplified models shown with specific attributes related to decision making, can be viewed and assessed without needing to use complex CAD-related visual metaphors [276].

BIM is therefore supposed provide a greater Return on Investment (ROI) for its practitioners, and ultimately replace non-digital documentation and provide centralized data sharing between involved stakeholders. While the adaptation of BIM has also become mandated in various countries (e.g., in the UK, the use of BIM is mandatory for public procurement) [100, 65], the adaptation of BIM in various AECOO sectors poses particular challenges, which have been summarized by Volk *et al.* [265]:

- Increasing costs, effort and time for generation of usable and semantically-rich data for generation of up-to-date BIMs.
- Lack of flexible and accurate automated methods for updating of existing BIM documentation, often requiring domain expertise for manual generation of models.
- Paucity of approaches and methods for processing and analyzing digitized documentation, entities and relations within the BIM model.

Thus, while the use of BIM, in theory, can provide a solution for completely centralized and digitized building documentation, in reality its adaptation and use are still problematic. This is especially the case within the realms of FM and Real Estate 4.0, where up-to-date representations are crucial for daily assessment and decision making activities.

2.2 Facility Management and Real Estate 4.0

Facility managers are becoming more aware of the need to update and reference building features with existing or newly generated BIM datasets in order to provide an updated building O&M infrastructure. According to Kensek [134], one key challenge in modern FM is to digitally reflect the current state of the built environment. Another key challenge is to enable users to monitor and forecast the current state of the built environment, especially if digital documentation is integrated with real-time or historic data that is used to provide feedback and support decision making [52]. These challenges create a demand for automated generation, and updating of digital representations of indoor areas - using methods that suit the modern IoT environment. There is currently a disparity in FM between owners and tenants, specifically as to which groups have what specific access to O&M data (Fig. 2.1).

Current documentation practices for O&M procedures within the FM realm include using redundant, often outdated, information that is usually available in paper format only [252]. A centralized and digitized system that can be used to replace any paper-based documentation, while giving specific access to legible stakeholders, can greatly increase the operational efficiency of modern buildings. This need for integrated solutions is also becoming more pronounced as practices from Industry 4.0 are currently being evaluated

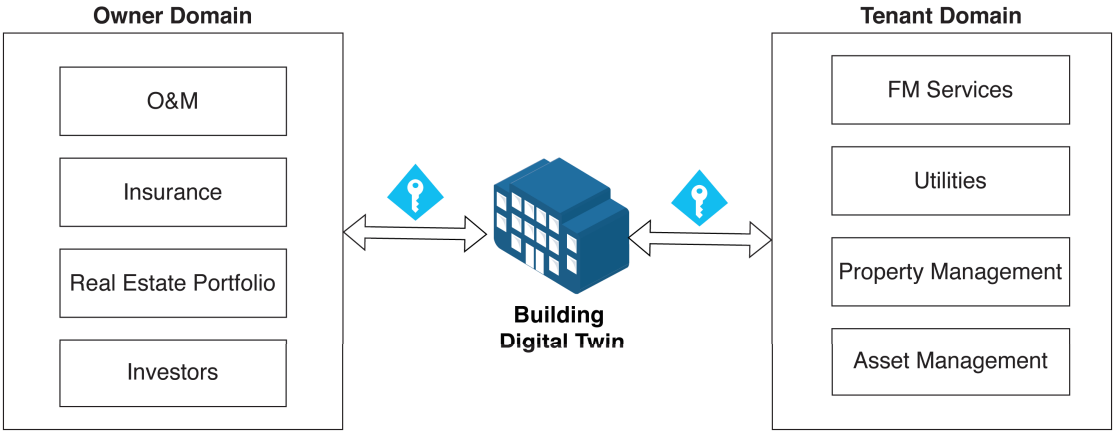


Figure 2.1: *The current landscape of building management is usually divided between owner and tenant/occupant access to processes and decision making. A key concept of using Industry 4.0 practices within FM is creating a new paradigm, called "Real Estate 4.0", which in turn enables the unification of owner and tenant stakeholder decision making using a centralized digital representation of a building in the form of a DT.*

and adopted for FM use [253, 146]. This in turn leads to the term of "Real Estate 4.0" or "Real Estate Management 4.0" when describing the use of such practices within in post-construction context of a given building [180]. There are a number of different acronyms used in FM for referring to integrated building management systems and software. Certain systems are best suited for complex operational environments e.g., warehouses and factories, and others are more suited for office buildings. While there is some overlap between these acronyms, it is important to define each one for clarity [123]:

- *Building Management System (BMS) or Building Automation System (BAS)*: Integrated systems used for monitoring various Mechanical, Electrical and Plumbing (MEP) equipment and resources. They are also used to control safety-critical systems e.g., fire, security, and occupancy comfort systems – using automated features often tied to enterprise software solutions.

- *Computerized Maintenance Management System (CMMS)*: Software systems used for management of maintenance specific tasks, often aimed at technical maintenance personnel (e.g., engineers), and include in-depth representations of various MEP installations (e.g., air-conditioning and ventilation system diagrams).

- *Computer-Aided Facility Management (CAFM)*: Similar to a CMMS, but includes a broader range of monitoring and management components e.g., machinery, room usage, stock control and health and safety.

- *Integrated Workplace Management System (IWMS) or Facility Management Information System (FMIS)*: Also similar to CMMS and CAFM, but focused more on real-estate management, an particular subsets of O&M e.g., space management.

- *Environmental Management Systems (EMS)*: Systems use for monitoring environmental features related to sustainability e.g., carbon footprint calculations, water and electricity usage and alternative energy sources management (e.g., solar panels).

According to Roper and Payant [206] the use of building automation, combined with computer controlled and digitally represented state of a building, can be accomplished with the use of an IWMS. These IWMSs must be able to communicate and provide an informative analytical output of the state of a given FM operation to stakeholders, provided adequate Information Technology (IT) infrastructure is present within the organization using them. It is therefore crucial that these systems have access to current and historical operational data of a given building. The use of digitized representations of the built environment, along with various IoT and digital documentation integration, has lead to the development of prototypical software solutions aimed at providing useful insights into the the operational status of buildings to FM stakeholders. An early example of such an approach has been described by Khan and Hornbæk [137]. Other recent examples include:

- The use of modern games engines to integrate various data sources and present them interactively to stakeholders via a BMS-oriented user interface [136].
- A web-based, BAS software prototype for assessing energy usage within a school building, making use of interactive 3D visualization of a BIM model and 2D data analysis of related energy usage readings [148].
- An integrated BMS, encompassing a sensor data framework, and making use of BIM models for representation and focusing on occupancy comfort analysis [184].
- A conceptual software system for *Visual Management* of construction progress monitoring, integrating various digital data sources and streaming this data to a database for further use by an Enterprise Resource Planning (ERP) system [254].
- A cloud-based BIM platform for building services management, making use of detailed indoor BIM models with associated identification numbers for all items and areas, alongside location-specific sensor data visualization [49].

While the cited prototypical and experimental software solutions can be considered feasible, most of them rely on the use of existing BIM data as the primary data source. There is currently a paucity for software solutions for integrated FM and Real Estate 4.0 that are not dependant on BIMs for up-to-date building representations.

2.3 Visualization for Facility Management

FM stakeholders require support for updating and referencing available building features with respect to existing or newly generated BIMs. Taking into account the whole life cycle, the operational cost of a building is generally far higher than the construction cost (five to

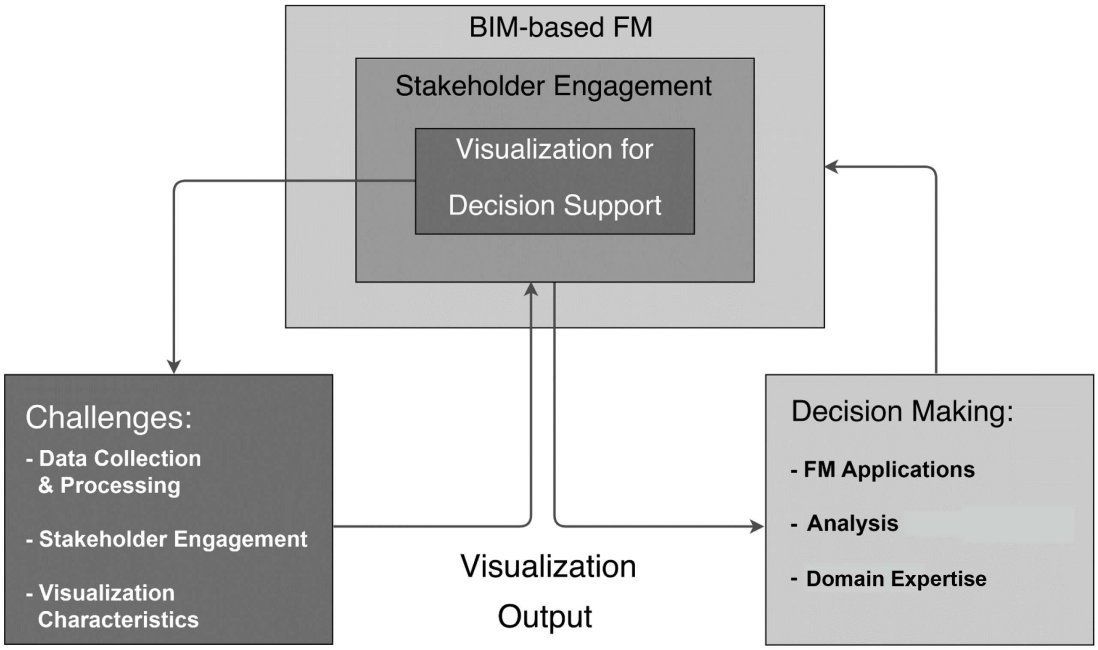


Figure 2.2: *Illustration showing the scope of the visualization problem within FM applications.*

seven times) and, therefore, the use of intuitive, transparent, and informative stakeholder engagement systems based on BIM provides benefits to operations, optimizations, and cost reduction [149]. The use of visualization and analysis of relative data is therefore crucial for transparent and collaborative decision making amongst involved stakeholders. For example, to visualize any spatial differences between as-designed/as-built versus as-is data (Chpt. 6), it is important to understand where this visualization fits into the FM decision-making scope.

Additionally, the use of interactive visualization allows FM stakeholders to gain a better understanding of the built environment and make decisions more quickly. This also enables the important practice of data sharing, and enables FM stakeholders from all other related FM sub-domains to have access to essential building operation information [134]. Therefore, each level of decision making for stakeholder engagement (including visualization) can be thought of as a layer within a decision making system associated with the FM stage of the building life cycle process (Fig. 2.2).

2.4 Digital Twin Representation of Buildings

In terms of built environment representations, a DT representation fuses *as-designed*, *as-built* and *as-is* physical representations, with additional information layers pertaining to the current and predicted states of their corresponding physical entity [188]. Therefore, a DT can be thought of as a system with constantly updated inputs and processing

parameters, which can adapt to changes and provide outputs in terms of analytics. Such analytics can be used for assessment and forecasting of the state of a given process or the current physical representation of a built environment element (e.g., building, office, machinery, etc.).

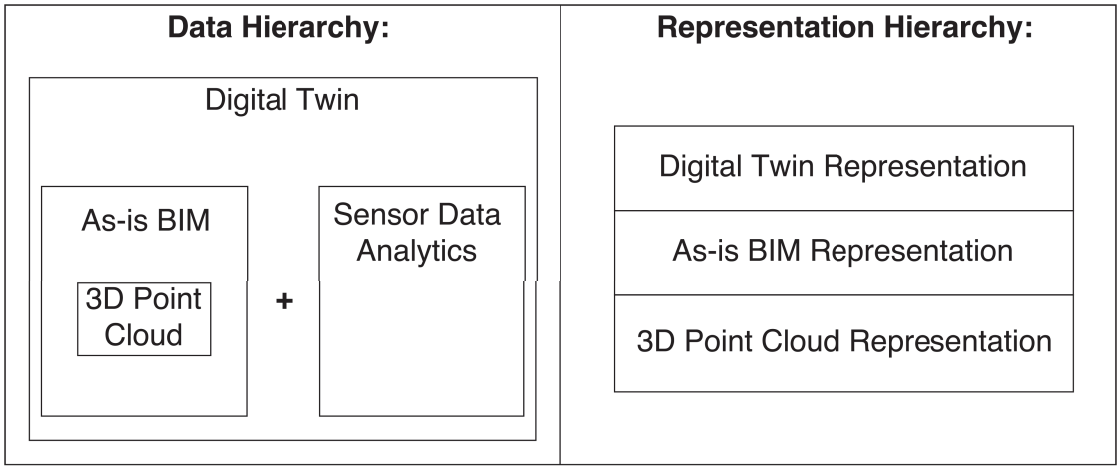


Figure 2.3: *A comparative diagram illustrating the hierarchies between data and representations associated with an existing building. The Data Hierarchy (left), illustrates the concept of a DT representation that fuses current and historic data sources in order to generate an up-to-date representation of a building. This includes the physical representation of a building, real-time and historic data (e.g., sensor data), and any other associated digital documentation. The Representation Hierarchy (right) illustrates the DT as a highest level representation when compared to as-is BIM and semantically-enriched point clouds. This level in the representation hierarchy is based on the ability of the DT to make use of data analytics in addition to as-is representations, and to potentially forecast states based on current analysis of outputs concerning various processes associated with the O&M stages of a buildings management.*

The use of BIM-related data, along with real-time or historic sensor data as well as digital documentation (e.g., floorplans, contracts or manuals, etc.), can be combined, analyzed and provide useful analytic insight into the building that can be addressed by FM personnel [242]. Although there is some overlap in terms of applications between DT and BIM representations for visualization, analytics and decision making, for this research DTs are treated as a higher-level representation that includes both *as-designed*, *as-built* and *as-is* BIM data and any other associated semantics. A DT representation fuses these representations with additional information layers pertaining to the current state of the built environment (Fig. 2.3).

DT systems greatly benefit from AI concepts and methods, particularly with the use of Machine Learning (ML) for training certain components of a system to predict outputs based in given inputs (e.g., historic spatio-temporal data found in recorded sensor data readings) [60]. There have been previously published case-studies where prototypical DT platform implementations and framework components have been evaluated:

- Qiuchen Lu *et al.* [197] describe the development and testing of a prototypical DT of a university campus. They propose an integrated system architecture, making

use of *as-is* BIM models as the main representational data source, along with other data sources (e.g., real-time sensor data, IoT devices, asset management data, tags and other digital documentation).

- Kaewunruen and Xu [126] describe the development of a DT for a metropolitan railway station, using a multi-dimensional (6D) BIM, implemented with Autodesk Revit. The railway DT is able to model simulations using time and cost schedule variables as well as carbon emissions and renovation plans.
- Kaewunruen *et al.* [125] discuss the use of a DT for evaluating low-carbon potential for existing buildings, using a BIM model along with a hierarchical flowchart used for the selection of Net Zero Energy Building (NZEB) refurbishment and integration specifications.
- Jain *et al.* [119] describe a prototypical approach for incorporating DTs of buildings into existing Supervisory Control and Data Acquisition (SCADA) systems, with a focus on evaluating energy usage and performance.
- Jian *et al.* [121] present a theoretical framework for integration of various data layers pertaining to DT representations of buildings. They stress that the use of a DT to highlight dynamic processes and simulation results, via a centralized and IoT-enabled platform, provides great potential for enhancing building intelligence and sustainability.
- Baeder *et al.* [14] present and discuss a DT platform for simulation of city infrastructure and population dynamics. The platform integrates open-source and commercial data, with the current focus of simulation being on traffic analysis.
- Borth *et al.* [32] evaluate DT integration into system of systems architectures, with a focus on smart grid and smart building domains. The authors advocate a modular approach for DT integration into such systems.
- Martínez *et al.* [162] describe the use of simulation-based DTs in order to forecast future states of plant machinery for industrial manufacturing operations. They use a 3D plant model along with related data in order to simulate processes alongside actual ones, with the aim of acquiring data for analysis and further decision making.
- Lu and Brilakis [158] describe a prototypical DT of a reinforced concrete bridge, based on semantically enriched point clouds, and corresponding reconstructed geometry, which can be used for as-is BIM representations in further condition assessment and forecasting tasks.

It can be observed from the cited literature that there is a paucity for non-BIM based representations and analysis, especially for frequently updated digital models of buildings. Furthermore, there is a discrepancy between implementation architectures as most solutions rely on using monolithic and commercial software applications, while others advocate the use of system architectures tied to common IoT implementation specifications.

The implementation of a DT *platform* requires all of the data access to be centralized, so that relevant stakeholders can have access to specific outputs needed for their sector of operations (e.g., room management, energy management, occupancy comfort, health and safety, etc.). The platform must generally satisfy the following requirements, inspired by the formulation of typical software requirements as described by Wiegers and Beatty [273]:

- Be non-monolithic and scalable for different Information and Communications Technology (ICT) infrastructures.
- Provide centralized access to stakeholders.
- Have ability to process different data types as inputs.
- Have ability to analyse the different data types.
- Generate both visual and numeric outputs.
- Must be reliable, robust, secure, maintainable and portable.

These requirements are important as currently the majority of approaches for 3D visualization in FM rely on desktop-based applications, which do not necessarily offer flexibility in terms of hardware and software portability, maintenance and usage [103]. In order to meet these requirements for a DT platform implementation, a suitable software system architecture must be selected. A prime system architecture candidate for this is a SOA.

2.5 Service-Oriented Architectures and Systems

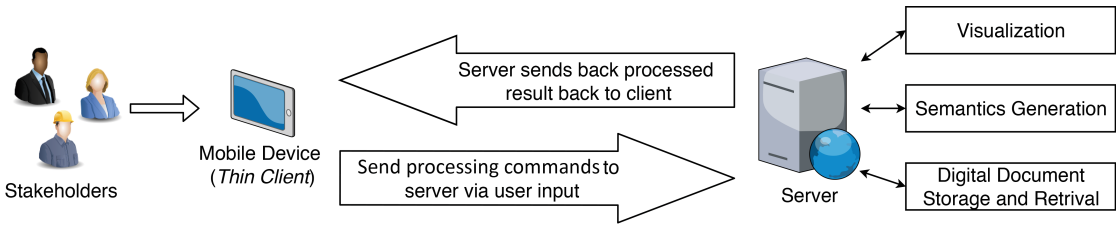


Figure 2.4: *A high-level example of a service-oriented approach for FM stakeholder engagement.*

The fundamental concept behind SOAs, and their SOS implementations, is the abstraction between the processing components (known as *service-components*), typically implemented as back-end software running on dedicated servers, and the front-end client interface used for requesting and receiving the computed results (Fig. 2.4). An important feature of SOSs is that they can decouple hardware requirements, needed for computing complex visualizations and analytics, from client devices that may not have certain hardware capabilities [108].

For example, service-based visualisation can enable streaming of complex visualisation results to clients running various hardware configurations e.g., smartphones and tablets. Some mobile devices may be older generation and may not have the ability to process visualisation data in real-time using their native hardware (known as *thin clients*). Thus, data can be pre-processed and streamed from a server in real-time to a *thin client* using an SOS implementation [62]. Apart from visualization, this approach can also be used for complex computation such as ML-based classification of certain data (e.g., point clouds [241]). Other examples of SOAs and SOSs used for built environment visualization, BIM and DT representations include:

- Franz *et al.* [83] describe the development and use of a SOS platform for collaborative reconstruction of indoor point cloud scenes, captured using commodity mobile 3D scanning devices, and with the stakeholder focus on crime scene investigation.
- Qiao *et al.* [196] provide an overview of the state-of-the-art for Augmented Reality (AR), via web-based applications and service-oriented architectures. They describe how the use of lightweight web-based software components can benefit the application of AR for practical visualization scenarios.
- Research by Zhao *et al.* [287] describes the use of a SOA for updating BIM and energy models during the design phase of a building construction project.
- Scully *et al.* [220] describes how web-based 3D visualization can be used for tracking of related 3D assets through a service-oriented and BIM-enabled online repository.
- Lee *et al.* [150] provide details about a prototypical web-based portal for collaborative BIM-based FM decision making, using an interactive 3D visualization approach, and implemented using an SOS.
- Richter and Döllner [205] describe the design, implementation and testing of an SOS for visualization of massive point cloud datasets, using Graphics Processing Unit (GPU)-based rendering methods.

The case studies in the cited literature demonstrate that the use of an SOA is a suitable software system design candidate for implementing DT platforms – by enabling the integration of various data sources, processing them and delivering the analysis results to stakeholders [253]. Furthermore, the advantages of using an SOS for visualizing BIM models, IoT-enabled smart home visualization, and point cloud for robotics applications have been described by Zhang *et al.* [284], Pouke *et al.* [189] and Toris *et al.* [256].

Due to the routine and frequent nature of FM operations, a centralized online system with on-demand processing capabilities and streaming to thick, medium or thin clients provides an advantage over using monolithic enterprise software on workstations [57]. This enables routinely captured data (e.g., point clouds) to be processed and visualized, and allows for analytic insight into various spatio-temporal data (e.g., sensor data readings related to occupant comfort [252]).

2.6 Point Cloud Representations of Indoor Environments

Figure 2.5: *Typical representations of indoor point clouds include Red-Green-Blue (RGB) or intensity-based point clouds, which may include additional attributes e.g., pre-computed normal vectors.*

Point clouds can be defined as a collection of points in space, with each having the mandatory attribute of a 3D position (usually in Cartesian coordinates), and optionally either an RGB color value (optionally with an additional opacity value), or an intensity-based color value (usually mapped as a single value to a grayscale gradient) as well as an optional normal vector that is computed post-capture (Chpt. 4.3.4). As such, once captured (e.g., by means of recorded signals from a scanning device), they can represent a real-world physical object or space (usually in its static state) (Fig. 2.5). Most notably, point clouds can provide 3D and textured up-to-date physical representations of the built environment up to high resolutions (millions of points per square meter), and featuring intricate details [58]. According to Richter [204] point clouds can *"represent almost any type of physical object, site, landscape, geographic region, or infrastructure...at all scales and with any precision"*.

However, point clouds consist of what can be defined as *non-interpreted data* – data that is open to visual interpretation but does not have any semantics associated with it. While point clouds can be used by themselves to represent the current state of the physical environment for practical needs (e.g., visual observation and assessment of space usage in a room), for more complex representations and assessment, the point cloud needs to be processed further in order generate useful semantics for it. The final output of a semantically enriched point cloud can then provide great benefit to AECOO practitioners and stakeholders e.g., FM operators who want a deeper, more detailed and intelligent insight into the current state of the built environment [198], or for use as e.g., base-data for reconstruction of indoor 3D floorplans in emergency and disaster management and planning [174]. Detailed descriptions of key approaches and methods for indoor point cloud capture, processing, analysis and visualization are further expanded in Chpt. 4 and 5.

Capture LiDAR systems have been used in the previous few decades for remote sensing tasks, and have since been adopted for use in AEC sectors [44]. However, the use of LiDAR devices was up until recently prohibitively expensive for regular use, and requires domain expertise to setup and to operate [183]. The use Terrestrial Laser Scanner (TLS) systems for LiDAR-based scanning enable the capture of highly dense point cloud representations (commonly in ten's to hundred's of Gigabytes in size), with an accuracy between one to ten millimeters for BIM applications, depending on the TLS method used (e.g., Triangular, Phase Difference or Time-of-Flight (ToF) [157]). These representations are usually encoded as intensity-based color values (e.g., grayscale), but may be combined with depth-sensing and Charged-Coupled Device (CCD) technologies (i.e., digital cameras), in order to fuse digital photography RGB and depth images with the point cloud – thus adding additional features to each point [18]. With recent technological advances in commodity mobile devices, some integrated LiDAR solutions are currently also available at affordable costs [221].

A second alternative method for capturing point clouds is with the use of *photogrammetry* techniques. Photogrammetry is based on the concept of matching sequentially recorded images in order to create a 3D reconstruction of the object or area in focus, using methods such as Structure from Motion (SfM) and Dense Image Matching (DIM) [175, 202, 94]. Photogrammetry can be more practical for routine capture of point clouds, but can still require significant processing time depending on the image-matching algorithm used as well as longer processing time for the generated dense point clouds. Additionally, close-range photogrammetry requires knowledge of photography methods and domain expertise when capturing objects e.g., bridges [122]. However, the generation of suitable point clouds for as-is BIM from photogrammetry is possibly a cheaper, quicker and practical alternative than using LiDAR-based systems such as TLS [40].

The third alternative method is the use commodity mobile devices, which can combine LiDAR and photogrammetry-based methods for capture of point clouds. The use of commodity mobile devices with depth perception capabilities (including recent commodity mobile devices with integrated LiDAR sensors), for capture of indoor point clouds, is advocated as the optimal method for capture of indoor spaces for FM applications (Chpt. 4.2). Commodity mobile devices that feature such capabilities can be termed as being Point Cloud Capture Technology (PCCT)-enabled. There are particular advantages and disadvantages for using PCCT-enabled mobile devices for point cloud capture, specifically concerning the built environment:

- Senthilvel *et al.* [222] provide a comparison between two earlier prototype devices based on the Google Tango and ZED camera specifications. They note that the devices can acquire a point cloud using the depth-estimation provided by a ToF and stereo-vision capture and reconstruction methods. The authors also noted the low cost, eas of use, high portability and adequate resolution of captured point clouds, making the use of such devices suitable for practical indoor capture.

- Virtanen *et al.* [263] describe a case study where they used a Matterport mobile scanning device to acquire an indoor representation of an abandoned psychiatric hospital for heritage-oriented stakeholder engagement through interactive 3D visualization. The authors noted that the Matterport mobile scanning system was best suited for capturing details of areas of smaller sizes, as for larger areas (e.g., long corridors or halls), the captured detail diminished. This point cloud was compared with a point cloud of the same area acquired using a TLS, using deviation analysis methods in order to verify its accuracy and suitability for heritage visualization applications.
- Boboc *et al.* [30] evaluated scanning results of an outdoor statue for heritage visualization, captured with a Google Tango-based mobile device. The authors evaluated the captured point cloud against a photogrammetry-based point cloud for surface deviations, and found that the point cloud captured with the Google Tango device was of suitable quality for heritage visualization applications (e.g., important surface details could be preserved in the reconstructed mesh representation). The authors also noted that the point cloud captured by the Google Tango devices was far smaller in terms of points, while being able to preserve all of the important heritage details of the statue.
- Kalyan *et al.* [127] present a comprehensive evaluation of the quality of as-built model capture for Construction Quality Assessment (CQA), using an early Google Tango prototype. The authors noted that the captured point clouds were not of high enough quality to be useful for CQA applications, but that they are more suitable for applications where the accuracy threshold error is tolerated above the limits of CQA.

Registration Registration of point clouds is the process of transforming and aligning a point set with another similar point set, 3D geometry or a 2D floorplan. While in most cases point alignment can be performed manually by the user, an automated approach can also be used if dealing with more complex point sets. Such automated approaches make use of spatial transformation in order to transform a point set to match either the complete set or control points of corresponding geometry, usually using iterative transformation algorithms [22]. Chen *et al.* [45] present a plane/line comparison approach for registering complex real-world point clouds.

An iterative point matching approach is also useful for aligning a given point cloud to a floorplan section, in order to add location-based semantics [269]. Such an alignment approach can also be extended to make use of latitude and longitude Global Positioning System (GPS) coordinates of the points if available (known as *geotags*), in order to align a point cloud with a map, site layout or floorplan [128].

Sub-sampling and Noise Removal Captured point clouds usually need to be sub-sampled, and have any visual noise and clutter removed using manual or automated methods [55]. In certain cases, the captured point cloud may need to be sub-sampled if it is too dense (thus takes too much memory or time to load and process).

Point cloud sub-sampling methods sample the given distribution of points using a selected point sampling scheme, where a certain number or percentage of points is removed. Such methods attempt to preserve the overall visual fidelity of the mesh [168, 277]. Point cloud noise that may be introduced due to surface roughness, partially captured data or overlapping point clusters, can be removed using manual or automated noise removal methods. For automated methods, the use of Principal Component Analysis (PCA)-based methods can be used to detect outlier points [215, 171].

Normal Vectors Calculation The normal vectors for each of the points can be computed post-capture, and used as an additional feature. The use of vector normals is important for tasks such as geometric surface reconstruction and segmentation. The computation of point normals can be accomplished by analyzing the local neighborhood of a point [167], where the normal vector is oriented according to the represented neighbouring points. The neighborhood of a point can be computed using the covariance matrix of the k-Nearest Neighbors (KNN), and corresponding eigenvectors and eigenvalues [109].

Segmentation The segmentation of a point cloud can be described as dividing and marking similar regions of point clusters, which allows for quicker identification of physical features [173]. Segmentation can either be performed manually, or using an automated approach. Most segmentation methods are suited for detection of clusters conforming to the spatial distribution resembling standard geometric objects e.g., cylinders, spheres, cubes, and planar surfaces. Using geometric primitive-based approximation methods such as RANSAC, which try to fit these points to a given geometric surface, objects e.g., walls, floors and ceilings can be detected and segmented [218]. Additional methods such as *region growing* and *semantic segmentation* are discussed in Chpt. 4.3.5 and Chpt. 7.6.

Visualization Point clouds can be visualized in 3D by projecting each point as a vertex primitive in a 3D Certesian space, along with its assigned material properties (this usually includes the color and the *shape* of the point based on an assigned texture image). The use of the modern GPU-based programmable graphics pipeline enables real-time rendering of larger point cloud data (with millions of points), along with stylized rendering for further enhancement of visualization results (Fig. 2.6). The use of *shaders* (special programs that run on the GPU) can additionally be used to implement specific rendering styles and effects on given geometry (e.g., point clouds, BIM data, CAD models, etc.) [5].

With the standardization of the Web Graphics Library (WebGL) Application Programming Interface (API) for most modern web browsers, it is possible to visualize in real-time 3D various models and generated outputs on client devices, ranging from desktop personal computers to mobile phones and tablets. However, one limit of most web-based 3D visualization frameworks and APIs is the lack of support for out-of-core rendering of massive amounts of point-cloud data. This requires resorting to streaming and optimization methods based on scene-partitioning, using data structures e.g., octrees or k-d trees [219, 205].

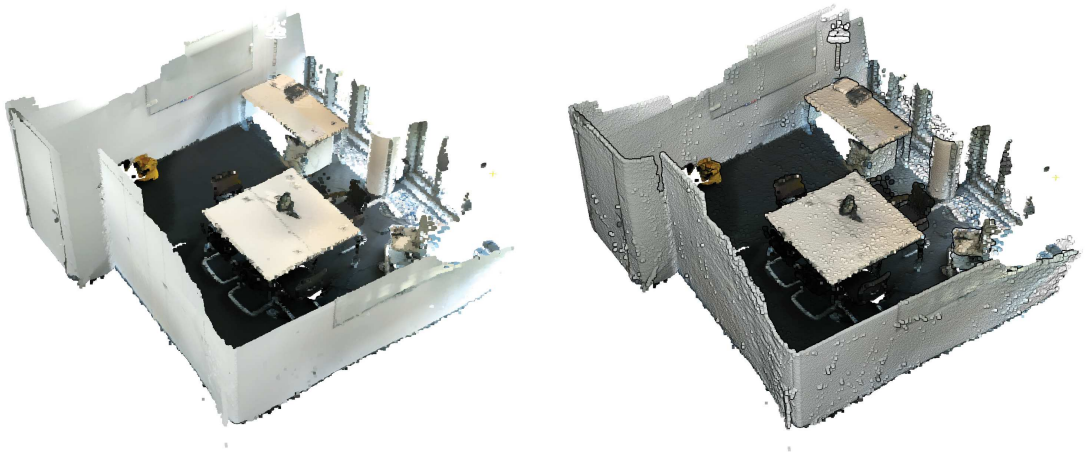
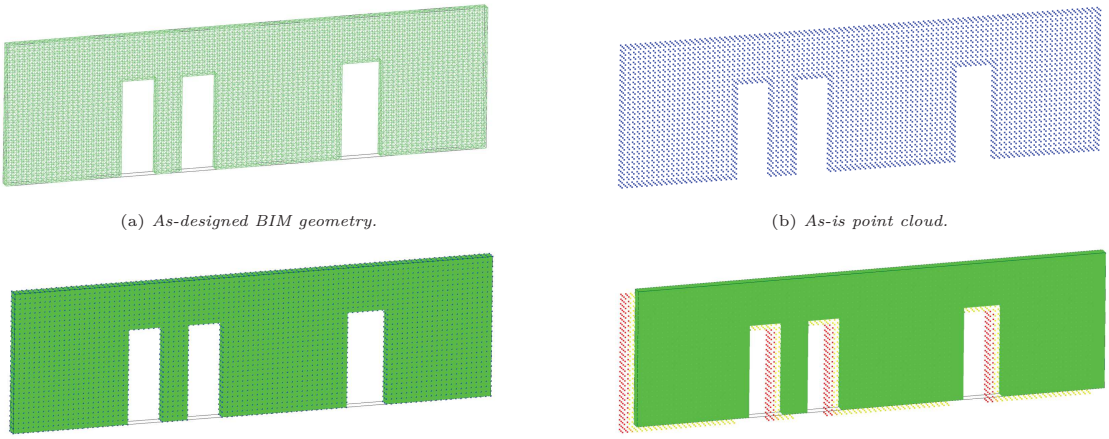


Figure 2.6: *Example of edge detection stylized rendering for an indoor point cloud (right), compared to the default point shading method using the color value of the points only (left).*

2.7 Spatial Deviation Analysis



(a) *As-designed BIM geometry.*

(b) *As-is point cloud.*

(c) *Aligned as-is point cloud with as-designed BIM geometry.*

(d) *Misaligned as-is point cloud and as-designed BIM geometry, with spatially deviating points highlighted in red.*

Figure 2.7: *Example of spatial deviation analysis between as-designed BIM geometry and as-is point cloud when aligned and misaligned.*

The problem of comparing spatial differences between aligned geometric representations of *as-is* or *as-built* versus *as-designed* geometry representations is a common occurrence, especially when comparing point cloud and BIM geometry representations (Fig. 2.7). This may include comparing the inclusion or exclusion of certain building elements when compared to the original *as-designed* built environment representation, their spatial alignment corresponding to existing documentation (e.g., floorplans), and changes of core structural elements (e.g., wall constructions or demolitions, or addition of new elements

such a rough openings for doors and windows frames), which deviate between *as-designed* and *as-is* representations [234].

Previous literature discusses and attempts to find the solution to the problem of spatial deviation analysis:

- Anil *et al.* [8] focused on using an existing commercial software tool to import corresponding *as-is* BIM and point cloud representations, in order to perform comparative deviation analysis – using a computationally expensive method based on minimum Euclidean distance comparison.

- Bosché and Guenet [34] presented a set of methods for evaluating surface regularity, based primarily on matching points from a point cloud to an *as-built* BIM component representation. The points are matched to the corresponding BIM geometry using either surface proximity (measuring the orthogonal distance of a point to the BIM geometry surface), or using surface normal similarity (comparing the pre-computed normal vectors of the each point to the normal vectors of the BIM geometry surface).

- Research by Turkan *et al.* [260] looks at using spatial deviation analysis methods for tracking of construction progress on building sites, specifically looking at detecting primary and secondary components from point clouds. They present three techniques for deviation analysis, first two of which are based on earlier research by Bosché and Haas (2008) – based on aligning the *as-built* BIM geometry within the corresponding the point cloud, where the range between the *as-is* and *as-designed* point/triangle intersection is measured using ray tracing, while their third method uses the bounding volumes of negative areas of each BIM geometry component to detect a number of points contained inside in post alignment [33].

- Bassier *et al.* [19] evaluated the use of a Finite Element Analysis (FEA) method for structural analysis of heritage timber roof structures. They compared two aligned models and checked for deviations in the post-registration phase - one was a simpler wireframe model and the other was a complex discretized mesh representation. They noted the benefit of using a more complex geometry discretization scheme for increased spatial deviation analysis accuracy, as opposed to using the least complex geometric representations often featured in BIM geometry models.

- Wang *et al.* [268] used deviation analysis for comparing an *as-is* point cloud captured using a remote Unmanned Aerial Vehicle (UAV) and the corresponding *as-designed* BIM model. They recommend the use of different registration methods for alignment of comparative data, particularly the zone fitting algorithms described by Choi and Kurfess [50] - assuming no matching coordinate information is shared between the two data sets. The deviation analysis is then performed automatically using commercial BIM software.

- Bonduel *et al.* [31] describe the use of *as-is* point cloud and *as-built* BIM comparisons for detecting micro and macro spatial changes. They make use of the CloudCompare

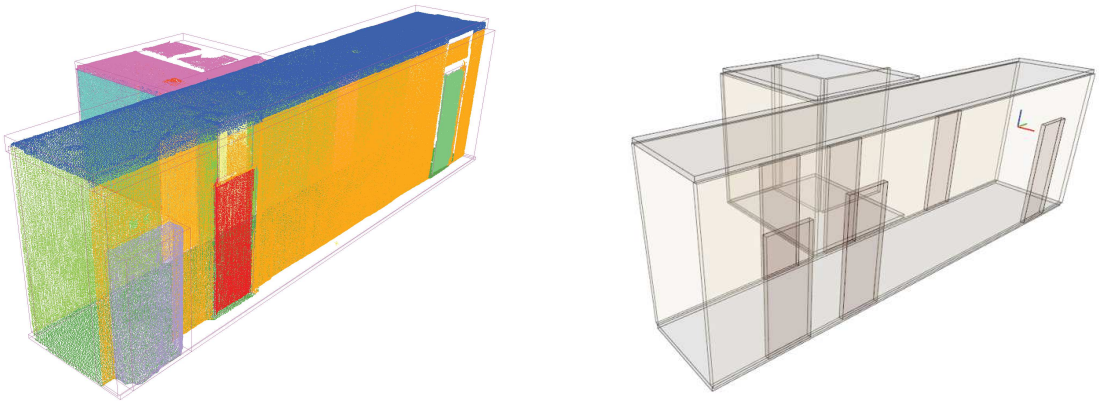
software tool [88], to carry out the primary deviation analysis for detecting macro spatial changes, and use a custom plugin for removing certain components (e.g., windows), from a BIM model prior to detecting micro damage.

- Talebi *et al.* [249] present a processing pipeline for tolerance compliance measurement using point clouds captured with a TLS. The use and visualization of spatial deviation analysis results is deemed as crucial for evaluation of the registration accuracy of captured point clouds.

It can be observed from the cited research that there is a paucity for a flexible and computationally efficient deviation analysis solutions. The two main problems are: (1) Registration of the two data sets that are to be compared in a same coordinate system and transformation frame, and (2) Efficiently calculating the spatial deviations for hundreds of millions of points to the nearest matching geometry surface.

The first problem can in most cases be solved using a selected point registration scheme e.g., Iterative Closest Point (ICP) [164]. The second problem is more complex to solve, as it requires the use of a geometry discretization and point-to-polygon comparison scheme. This may also include the generation of an additional Finite Difference Mesh (FDM) representation of the as-designed or as-built BIM geometry [21, 93] as well as the post-processing of the compared point cloud geometry [232].

2.8 Point Cloud Reconstruction



(a) *Processing and reconstruction of captured point cloud data (e.g., segmentation).*

(b) *Generated as-is BIM representation.*

Figure 2.8: *Example of reconstruction of an as-is point cloud to an as-is BIM (process known as "Scan2BIM"). The captured point cloud (left) is processed using various steps such as normals computation and segmentation as well as semantic-enrichment. The semantically enriched point clusters from the point cloud can then be used to generate as-is BIM geometry components. 2.8(a) shows the initial results of segmentation of an office hallway, and 2.8(b) shows as-is BIM reconstruction result of parts of the same hallway.*

The fundamentals of geometry reconstruction of points are rooted in the field of computational geometry. Important early research by Preparata and Shamos [193] described efforts to implement computational geometry algorithms, in order to enable the generation and rasterization of 2D and 3D geometry on early mainframe and personal computers.

The generation of 3D mesh approximation from segmented point cloud clusters is required for further semantic enrichment operations. Solid-geometry and *watertight* mesh representations, based on point clouds, are particularly useful for generation of *as-is* and *as-built* BIM representations [178, 250]. An approach using the Random Sample Consensus (RANSAC) segmentation method is presented by Liu *et al.* [154] for reconstruction of building models from both complete and incomplete point cloud representations.

Methods for reconstruction vary depending on the intended usage of the generated mesh representation (Fig. 2.8). This process is often referred to as "Scan2BIM", and involves a number of sub-processes that parse, optimize, enrich and reconstruct the point cloud – in order to generate higher-level geometric representations (e.g., BIMs, floorplans, etc.) [169]. Certain methods provide more accurate reconstruction, by better preserving the shape of the reconstructed point cluster.

Reconstruction of point cloud geometry to a triangular mesh representation requires the use of approximation algorithms, which can detect a minimum number of vertex primitives in order to triangulate the required mesh surfaces. The simplest and least accurate method is the approximation of a 3D convex hull shape [17]. This approach is useful for approximating the overall volume dimension of the 3D shape represented by a reconstructed point cluster, but does not preserve any important details e.g., surface cavities and curvature.

A more robust method that can preserve the surface curvature to varying levels of detail include the Poisson surface reconstruction method [130], which is generally more suited towards reconstructing organic shapes due to its nature to smooth hard edges in its 3D shape approximation. The Ball-Pivoting Algorithm (BPA) is a simple surface reconstruction algorithm that approximates a triangular mesh by connecting every three vertices that touch the radius of a rolling sphere [24]. The BPA method is generally able to preserve the hard edges often found in built environment representations, thus it is more suitable for AEC visualization and geometry analysis applications.

A Boundary Representation (B-Rep) of reconstructed point cloud and existing BIM geometry are also useful as parametric-based descriptions of 3D objects. B-Reps preserve the structure of the 3D mesh by distinguishing and linking the vertex, edge and face connections to a given surface – thus forming a boundary between solid and void 3D spaces [244]. B-Reps have the advantage of not requiring the use of triangular tessellation for rasterization and display of geometry – as the main topology (e.g., quadrilateral faces) and connecting geometry (e.g., edge lines and points) are used as the simplest primitive form for display and further evaluation.

In addition to triangular and B-Reps, a voxelized representation of point cloud and BIM geometry can also be used as a parametric surface representation. A voxelized representation of an *as-built* or *as-designed* mesh can be generated by evaluating the

shape and projection properties of the triangular mesh that is used to generate a 3D voxel grid [71]. This voxelized representation of the mesh can be computed, at varying resolutions, most commonly using octrees [110] – and can preserve most of the geometric features of the original polygonal mesh, making it useful for approximating spatial deviations of both regular or irregular (e.g., curved) geometry. The use of a voxelized mesh representation is similar to the style of FDM representations used for FEA.

The use of clustering algorithms also plays an important role in evaluating the topological properties of point sets. Often a 3D point cloud may be of very large complexity and may require subdivision of the point clusters for reconstruction or inspection purposes. The most common and established clustering methods for point data include k-means [155] and Density-based Spatial Clustering of Applications with Noise (DBSCAN) [74]. The k-means algorithm is best suited for approximating clusters that have no clear spatial divisions but are of varying density, while the DBSCAN algorithm is better suited for approximating point sets with uniform density.

The final important element of 3D mesh reconstruction from point clouds is the ability to use the reconstructed meshes for adding, subtracting or merging operations with other existing geometry of the same type. This requires the use of Constructive Solid Geometry (CSG) operations [81]. CSG allows for the approximation of cavities and merged geometry using Boolean operations. The quality of CSG results depends largely on the implemented partitioning and tessellation schemes, but often 3D meshes that have CSG operations performed on them will increase in geometric and visual complexity. The use of CSG also allows for the introduction of explicit 3D geometries into the implicit building component geometry representations using B-Reps [177].

2.9 Semantic-Enrichment of Indoor Point Clouds

Specific meaning in the form of semantics needs to be introduced to point clouds, in order to make them useful for further analysis, visualization and decision making tasks (Fig. 2.9). Reconstruction of indoor environments e.g., offices, houses, factories, storage facilities, etc., poses a particular challenge as reconstructed geometric data is not classified by default, and manual classification is a complex and time consuming process [43]. Methods for semantic enrichment of indoor point clouds are mostly focused on automated detection and classification of point clusters with either statistics or deep-learning-based approaches [270, 13].

Previous approaches for semantic enrichment of point clouds for indoor navigation have been described by Fichtner [76]. Object-specific classification for further enrichment of BIMs has also been researched, with examples including lamp detection [259], and door detection [199]. Additional research focusing on semantic enrichment and segmentation of point clouds includes:

- Research by Bloch and Sacks [29] describes an approach for room classification of standard apartment floorplans, and states the requirements for semantic enrichment of relevant BIM data.

- Research by Ikehata *et al.* [112] describes the implementation of structured indoor modelling for automated segmentation and semantic enrichment of floorplans from point clouds.
- Research by Poux *et al.* [191] presents a framework for semantic enrichment of indoor point clouds, using selected object features and analytic scene information to automatically insert 3D object models in place of classified point clusters.
- Sadeghineko *et al.* [213] describe the generation of semantically rich BIM models from point clouds, where each segmented region of a point cloud is given a unique annotation and used to form an ontology, thus enabling relationships between BIM elements to be captured and queried.

The use of semantics for BIM representations is the key factor behind the development of the Industry Foundation Classes (IFC) file format [36]. The IFC file format is intended to capture the geometric, spatial and semantic relationships between each of the components of a building within a representational hierarchy [261]. While the file format is meant to provide a general interface for all built environment applications, its complex structure and usage paradigms make it particularly challenging to update frequently (time, lack of interoperability between systems, and domain expertise being the main constraints) [133]. Within the scope of building management, the use of IFC data is mostly limited for infrequent digital building documentation generation, usually at the early design and construction stages, and less frequently during the operational phase of a building.

Another file format, which is meant to be more suited towards FM use is the Construction Operations Building Information Exchange (COBie) data format [66]. The COBie data format is essentially a spreadsheet-based file format used for assessing the *as-built* condition of a building, and for asset management. The file format does not include any graphical representations of the building by default, and therefore does not tie any semantics explicitly to any digital representation of a building [147, 252].

File formats for energy consumption and representation of "green buildings" are also becoming more common. Garwood *et al.* [87] present a conceptual framework for Building Energy Modeling (BEM), using as-built BIM geometry reconstructed from as-is point cloud representations of a modern manufacturing facility. The authors make use of the reconstructed as-built BEM envelope (based on the point cloud representation to to assess thermal and energy performance), and convert the as-built BEM model into an Extensible Markup Language (XML) schema-based file format (gbXML), and then use a variety of energy simulation software for validation.

Apart from reconstruction to *as-is* and *as-built* BIM representations, point clouds themselves can contain enough information to make them valuable for domain expertise use within the realm of AEC, and specifically FM. There is a paucity for using frequently captured point clouds for up-to-date representations of the built environment, specifically within a file format such as IFC. Krijnen and Beetz [145] propose an IFC schema extension for including point cloud using a compression method. Poux *et al.* [190] discuss

the need for a "smart point cloud" representation, comparable to an analytics platform in terms of proposed implementation – where point cloud semantics (domain expertise, raw point clouds, processing, sensor data, classification, etc.) are included with the complete point cloud representation for various analytical and stakeholder engagement purposes.

Semantic Enrichment Approaches Semantic enrichment of point clouds are typically based on three approaches: (1) Manual semantic segmentation, (2) Statistics-based and (3) Deep-learning-based approaches [99]. Manual segmentation involves direct input from stakeholders with domain expertise, who typically inspect the point cloud within a 3D software tool [9]. Using such a 3D software tool (e.g., CloudCompare [88]), stakeholders can also manually segment regions of interest and annotate them with additional layers of information (e.g., Uniform Resource Identifier (URI) codes, names of objects of interest, etc.). On the other hand, the use of automated (either fully or semi-automated approaches), based on either statistics or deep learning approaches allows for more flexible processing of larger and visually complex point clouds [255].

The use of statistics-based versus deep learning-based approaches is largely case dependent. For more complex environments with lots of clutter (e.g., indoor environments), the use of deep learning-based approaches provides better results [239]. Statistics-based approaches favour the analysis of the spatial distribution patterns of captured point clouds – in order to separate distinguishing clusters from similarly distributed point regions. Alternative approaches have been described by Ponciano *et al.* [187] for detection of objects in point clouds based on an ontology, where the characteristics such as density, bounding shape, color and distribution of point cloud clusters are used in order to create a knowledge-base for assignment of semantics.

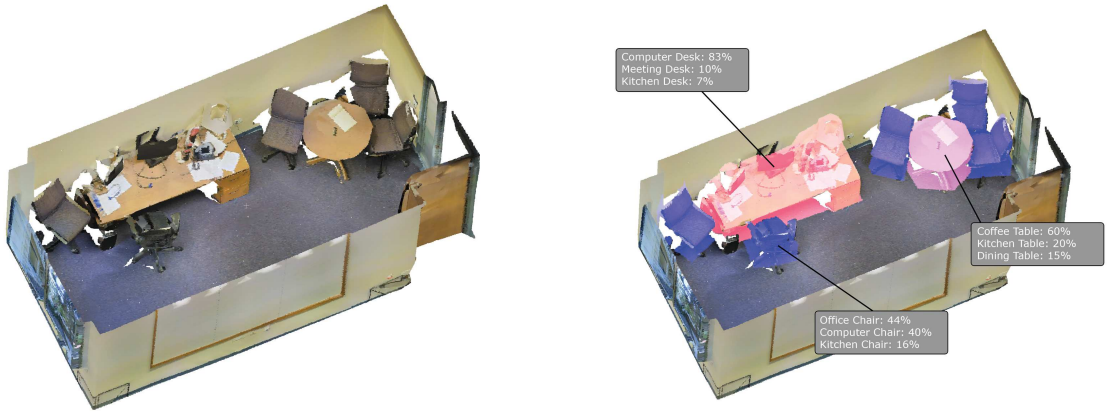
Wolf *et al.* [275] provide an overview of statistics-based classification techniques for semantic-enrichment of outdoor point clouds captured primarily using vehicular mobile mapping. They conclude that statistics-based classification techniques are useful for fast classification of outdoor point clouds, and can further be expanded by adding additional features and dimensions to the point cloud representation (e.g., 4D point clouds that include changes over time). Tran *et al.* [257] propose an approach for reconstruction of 3D floorplans using shape grammars, in order to establish topological relationship between segmented rooms found in typical office layouts.

Semantic enrichment of point clouds with deep-learning is typically based on using a CNN in order to classify point clusters using specific representation features [98, 42, 113]. The CNN can be used to recognize 2D or 3D spatial and visual features of a given point cluster [111]. Classification of 2D raster images of 3D point clusters, known as *multiview classification*, uses sequentially generated images of a point cloud cluster in order to classify them and then assign semantics back to the point clusters. [245]. The multiview classification results can be streamed back and associated with each point cluster of the point cloud using a SOS implementation. [236].

Apart from using images for training multiview classification models (typically 2D CNNs), the use of voxel and actual point data types can be used to train a 3D CNN [246], which can increase the classification accuracy of different objects belonging to similar

categories (e.g., a sofa-chair versus a chair). Specific 3D CNNs used for classifying point clouds e.g., PointNet [194] and PointNet++ [195], have been used successfully for such object detection scenarios as well as for classification of sub-components of objects (e.g., the wheels of a portable office chair, or a computer on top of a desk) [281].

The multiview classification approach requires only the use of 2D images of a 3D point cluster for classification, thus significantly reducing the bandwidth required to transfer data between the client and server for classification. However, the accuracy of the multiview classification approach depends on the quality of the images used for training as well as the viewpoint entropy of the images being classified [236]. Object-based classification provides more accurate results, but requires increased bandwidth transfer times between client and server as well as point clusters for training that are not as readily available as 2D image data [239].



(a) *Input point cloud.* (b) *Labelled point clusters.*

Figure 2.9: *Example of semantic-enrichment of an indoor point cloud 2.9(a), where point clusters representing common office furniture have been semantically-segmented using a deep-learning method e.g., multiview classification 2.9(b). In this example the probability of each of the segmented clusters belonging to a specific office furniture class is presented.*

2.10 Indoor Sensor Data Integration

One of the key challenges in modern FM is visualization of historic and current sensor data within *as-is* built environment representations, e.g., digital floorplans or 3D building models – derived from a building's digital representation (e.g., BIM, point cloud, CAD model, etc.) [251]. Sensors measuring building-related natural and man-made phenomena can provide information and insights about the current operational status of a building or a site (Fig. 2.10).

The linking of captured sensor data to the physical assets (e.g., rooms, MEP components) they refer to, and their combined visualization, and resulting enhancement of decision making and communication tasks, poses two key challenges for integration into SOS-based DT platforms [237]. The first challenge is the capture, processing and

integration of sensor data into a DT platform, especially if the platform is based on SOA principals. Since sensor data can be either real-time or historic, efficient processing of data (e.g., normalization of numerical values within a time series), and mapping this data to a specific building component within its digital representation is required. Previous research has attempted to address this problem:

- Arthur *et al.* [12] simulate the use of integrating numerous sensors within a given building. Their approach simulates the use of temperature and humidity sensors using a *Data as a Service Platform* paradigm, which allows for empirical results to be collected and analyzed for further implementation feasibility studies.
- Research by Kensek [135] focuses on integrating sensor data acquired using affordable sensors and its mapping to *as-built* BIM representations using commercial BIM software and related plugins.
- Pasini *et al.* [184] propose an integrated sensor framework, using remote sensors that are stored in a given BMS, and use it to perform analysis and building control tasks, while sensor data is visualized using the BIM for furthering stakeholder engagement.
- For construction site monitoring, Tezel, Aziz, *et al.* [254] state the advantages of integrating "Site Data Collection" of various IoT sources, and transmitting this data to a database and associated ERP as well as to thin, medium and thick clients for further digital documentation and visualization.
- Chien *et al.* [49] make use of a highly-detailed *as-built* BIM model together with associated digital object identification numbers, and then map location specific sensor data analysis and visualization outputs.
- Scheffer *et al.* [216] describe a prototype system for integrating sensor data readings into simple IFC model representations.
- Santhanavanich *et al.* [214] present a prototypical service-oriented platform for integrated sensor and geospatial information representation.
- Arslan *et al.* [11] describe the development of a prototypical BIM and sensor integration platform for monitoring temperature, activity and water sensor values using a distributed computation platform.
- Kazado *et al.* [129] present an approach for integration of real-time and historical sensor data and BIM, with a focus on utilizing existing CDE software and related APIs.

The second problem is concerned with transforming and visualizing processed sensor data. Since the processed sensor data is numerical and takes into account changes over time, methods for visualization of spatio-temporal data can be used. Such visualization approaches include: thematic color mapping of segmented indoor areas [185], textual

information displayed alongside the 3D model [189], abstracted 3D visualization elements e.g., 3D bar graphs [262], and combined 2D data results with spatially corresponding 3D scene markers for linked visualization of energy-related building data [227].

Combined visualization of digital building representations and sensor data can further be benefited with the use of real-time 3D graphics rendering technologies. Visualization methods enabled by such technologies are typically based on established *visualization idioms* used to convey geometric and textual abstractions of processed data [102].

Khalid *et al.* describes the development of a prototypical BMS using a game engine to enable combined interactive 3D visualization of built environment and sensor data [136]. The authors state the benefit of using a modern 3D graphics engine to visualize the information-rich BIM models in 3D, and to enhance stakeholder engagement through visual communication.

Chang *et al.* [41] describe the design and implementation of a BIM-based sensor data visualization platform for assessing occupant comfort in a classroom, based on temperature data readings and visualization. They note that their approach of sensor data visualization offers FM operators greater insight into occupant comfort and energy usage, in comparison the original 2D analytics and textual-based system they previously used.

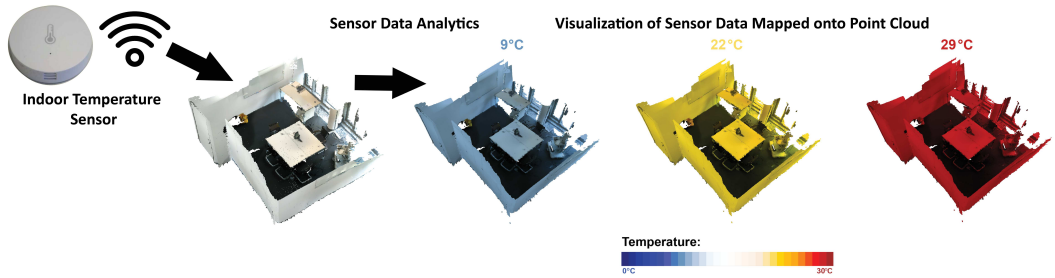


Figure 2.10: *Example of visualization of sensor data, using visual analytics to map processed sensor data onto a point cloud representation of an indoor office. In this example, temperature sensor data is visualized and combined with an indoor point cloud. The color of the point clusters representing the office change with each new temperature reading.*

2.11 Web3D-Based Visualization

Visualization of data provides key insights into problems by enabling communication of messages in way that is understandable and intuitive to the viewer. The types of visualization outputs can either be 2D or 3D. 2D visualization includes static images, videos and graphics. 3D visualization includes the projection of 3D geometry along with various shading computations in order to portray an object or a scene in either an abstracted or natural way (but most importantly the viewer is able to perceive depth in the given scene, and view the scene from any angle or position). 3D visualization can also either be static or animated, and also interactive in real or semi-real time [5].

According to Keim *et al.* [131], the process of visualization can be summarized as:

(1) The collection of data, (2) processing and transformation of data, and (3) mapping of the transformed data to a specified visualization scheme.

This concept, referred to *Visual Analytics* [132]. Interactive 3D visualization is important for presentation and inspection of critical data (e.g., BIM-related data) to stakeholders, particularly when examining such data for decision making [27].

The main objective of interactive visualization within the context of decision making is to provide useful insights, which otherwise are too complex to interpret without visualization, so that it can be used to identify potential concerns or points of interest [233]. For example, layouts of building spaces presented in 3D provide opportunities for FM personnel to obtain insights how features of a building location are associated with particular facility use [117].

The term Web3D was first introduced in order to define a set of loose technologies and standards aimed at delivering real-time 3D content of the World Wide Web [267]. Since then, some of these standards have become widely used and are now supported by most modern web browsers. Campbell notes the immense benefit Web3D can offer for the AEC sector in terms of BIM-related data visualization and stakeholder engagement using web-based platforms [38].

The majority of approaches for 3D visualization in FM rely on desktop-based applications that do not offer the flexibility of service-oriented applications (e.g., streaming to lower hardware specification mobile devices). Integration of interactive 3D visualization with SOSs, especially using Web3D technologies, provides enhanced flexibility to compute visualization results and stream them to all connected clients - thus further benefiting and democratizing the decision making process amongst stakeholders [103].

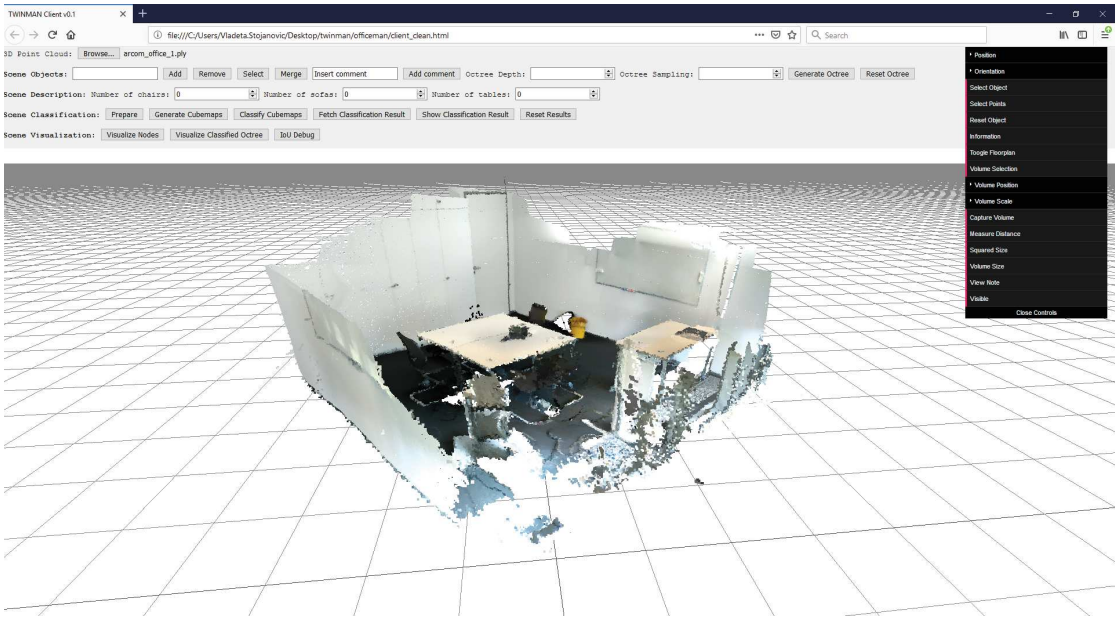


Figure 2.11: *A 3D representation of an indoor point cloud can be viewed and inspected interactively within a modern web-browser, implemented using Web3D frameworks and APIs.*

Interactive 3D Visualization for Facility Management The use of interactive 3D visualization can benefit FM stakeholder engagement, information sharing, and collaboration by allowing real-time display and analysis of 2D and 3D visual outputs generated from the acquired data sources [280]. Using modern computer graphics rendering approaches, particularly those based on real-time 3D rendering, complex visualizations of BIM, point cloud and related digital building data can be visualized using modern web browsers that support Web3D standards such as WebGL [59]. With the support of the WebGL 3D graphics API for most modern Hyper Text Markup Language (HTML) 5 compliant web browsers, it is possible to visualize in real-time 3D various models and generated outputs for AEC applications (Fig. 2.11). Additional examples include:

- Liu *et al.* [153] describe an approach for parsing and visualizing complex IFC-based BIM models within a web-browser using Web3D technologies. They note the need to optimize the BIM data before visualization, and implement this using a service-oriented paradigm.
- Hamza-Lup and Maghiar [106] present a prototypical hardware-software system for processing and Web3D-based visualization of temperate and humidity readings.
- Yan *et al.* [279] describe a multi-agent system for fire evacuation planning and procedure training using a Web3D-based visualization system.
- Rasys *et al.* [200] describe a prototypical Web3D information integration framework for FM applications. Their approach aims to deliver engineering domain-specific information with 3D representations for FM stakeholder decision making.
- Zhou *et al.* [288] present a WebGL-based BIM visualization application. Their application processes and compresses complex BIM geometry and converts it into simpler triangular mesh representations, which can then be displayed in WebGL compatible browsers.

Web3D-based technologies are suitable for processing complex BIM, point cloud and spatio-temporal data (e.g., sensor data), while providing both high and low-level control over the implementation of their rendering functionality. Since most modern web browsers support Web3D technology standards, selecting an appropriate Web3D framework or API is crucial. A number of programming frameworks for Web3D technologies enable the use of real-time 3D features e.g., model loading in different file formats, scene navigation, 3D data structures, and GPU-based rendering [56].

Three.js is a Javascript-based Web3D framework that offers generalized, high-level and specific features suitable for visualizing AEC data (e.g., support for different model file formats including point clouds, scene navigation, 3D data structures, and GPU-based rendering) [37]. According to Goetz *et al.* [92], the use Web3D frameworks for SOS integration is desirable for specific reasons:

- The visualization system does not have to be maintained by the client, as the core processing and computation tasks can be implemented and maintained server-side.

- Since the client front-end is usually tied to a web-based interface, it is hardware and software platform independent (only the clients web-browser needs to support the specified Web3D standard).
- Use of powerful server hardware enables computation of complex visualization, and if the client is capable of running the visualizations then the processing can be scaled and split between the client and server.
- Location independent use of the visualization system, as the client just needs to be connected to the server to receive any visualization results or data.
- Possibility for collaboration in real-time with multiple users having simultaneous access to the same 3D scene for enhanced decision making.

Therefore, the combined use of Web3D technologies with an appropriate SOS implementation is the key combination for enabling processing and visualization of complex data for use with DTs of indoor built environments.

Chapter 3

Service-Oriented Architecture and Systems

A SOS allows for computation and streaming of results from a remote server to various client configurations. In most cases the ability to decouple hardware and software requirements from the client allows for a more flexible approach to the processing of data, as the the users are not restricted to using high-end workstation computers, nor monolithic software implementations, for required computation tasks (e.g., point cloud classification, sensor data visualization, etc.). In this chapter a conceptual reference SOA is presented, along with specific details concerning variant prototypical SOS implementations based on this reference. The implemented SOS prototypes were used to assess the feasibility of key DT software components, and are referenced through ought the rest of the thesis. The conceptual reference SOA and prototypical SOS implementations presented in this chapter are based on previously published work in Stojanovic *et al.* [241, 232, 237].

3.1 Service-Oriented Architecture Overview

According to the SOA reference model provided by the OASIS organization [159], an SOA can be thought of as: *"....a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains".* In practice this translates to the application of a SOA when software requirements state a need for a modular, scalable and interoperable software components. A SOA is scalable as it does not assume any software or hardware platform requirements for implementation. Thus, the use of a SOA paradigm enables the design of software architectures that are able to provide *services* for *clients*, where the SOA is implemented by a *service provider* as a SOS.

An SOS provides access to *services*, which are utilized by clients through a defined API. Services can be thought of as an abstraction layer above the *component* implementation within an SOS. A component is a software implementation that provides a specific processing task for a service. For example this may be a component for segmenting point clouds, which is part of a point cloud processing service.

The implementation of services and components is largely platform and language agnostic, however, for applications that require deployment over networked infrastructure, especially over the Internet, the use of *web services* becomes important. Web services can be defined as the implementation of a messaging protocol coupled with functionality

for transporting data packets e.g., implemented using a data description language such as WSDL [1]. Furthermore, according to Endrei *et al.* [72], the main advantages of web services for implementing SOAs are that they are highly modular, interoperable and self-describing (only the format and content of the exchanged messages using a given protocol are important).

Finally, most SOA implementations follow the principals of *loose coupling* [124], where key services run and communicate asynchronously, and can be updated independently of each other. This allows for the development and implementation of web services using platform and language agnostic software components.

3.2 Service-Oriented System Requirements for Digital Twins

According to Döllner and Hagedorn [61], the use of SOSs enables the integration of complex data sources e.g., visualization of 3D city models. This may include CAD, GIS and BIM data, amongst other digital documentation. In the simplest case, the use of a SOS enables the rendering of complex scenes on a server, with the final result being streamed to a client. The use of such a paradigm is beneficial when the main requirement of a SOS is to provide visualization. However, the need for a SOS such as in the case for a DT platform, which makes use of routinely updated data and relies on integration of semantics with processed data, requires the SOS to accommodate specific functionality [283, 53]. Based on the existing literature review of Real Estate 4.0 and DT platforms and systems integration (Chpt. 2.2 and Chpt. 2.4), as well as general FM guidance [206, 252], the following key requirements for an SOS were identified:

1. The ability to process multiple types of data from various sources.
2. The ability to generate and send results to a variety of client configurations.
3. The ability to interface with existing systems e.g., Database Management System (DBMS), sensor platforms, BAS, IWMS, etc.
4. The ability for key software components to be implemented as services that can receive and send data to clients.
5. The ability for services to be added, replaced and removed in a modular fashion.
6. The ability for the SOS to provide a level of automation for key data processing tasks.

Similar requirements and features for SOS implementations for web-based 3D visualization have been previously described by Repplinger *et al.* [203] and Schiefer *et al.* [217]. Hagedorn and Döllner [103] describe the use of high-level web-services for implementing an SOS for BIM visualization and analysis. The presented BIM visualization web service is designed to stream visualization outputs of complex 3D scenes that are made up of

[1] WSDL: `https://www.w3.org/TR/wsdl.html`

multiple data sources related to BIM and 3D city model visualization. For example, the use of the Web Feature Service (WFS) Open Geospatial Consortium (OGC) standard [2] allows for streaming and modification of the objects attributes (rather then complete object files).

The additional use of Web3D services can be utilized for computation of scene rendering results (e.g., using streamed scene graph descriptions for generation of BIM models [217]). Such requirements have influenced the design decisions of the presented conceptual reference SOA and prototypical SOS implementations.

3.3 A Conceptual Reference Service-Oriented Architecture

A conceptual reference SOA is presented in Fig. 3.1, with a focus on DTs within the realm of Real Estate 4.0. Variants of this architecture are implemented as prototypical SOSs, and used for demonstrating the feasibility of key DT platform software components in the presented case studies. Such a system architecture aims to decouple software and hardware requirements from the client, while being able to communicate with various external system components (e.g., DBMS and FM-specific interfaces), and to enable combined analysis and visualization of related data (e.g., BIMs, point clouds, sensor data, floorplans, digital building documentation, etc.).

The use of SOSs enables scalable integration of key processing components, software frameworks and data sources for varying application requirements. The scalability of a SOS implementation depends on the kind of client the end result is sent to, and/or further processed on - this includes commodity hardware personal computers and mobile devices.

Client Types Clients are distinguished by their hardware specifications and processing capabilities as *thick*, *medium*, and *thin* clients. *Thick clients* are typically workstation computers or servers that have a high hardware specification. Such clients can be used to process data e.g., point clouds, sensor data, etc., and can compute the final result for an analysis or visualization.

Medium clients are defined as commodity personal computers and mobile devices, which feature hardware that is still capable of visualizing and processing the required results. However, they are best suited for receiving the partially or fully processed data, and computing the combined visualization without needing to complete more complex computations.

Thin clients are usually defined as low-end hardware specification devices (e.g., consumer mobile devices), which do not have the sufficient hardware capabilities to compute the visualization result in real-time. Thus they are provided only with the finalized visualization result, usually in the form of an image of the current 3D scene, which is rendered on the server side and streamed to them [62].

[2] WFS: `https://www.ogc.org/standards/wfs`

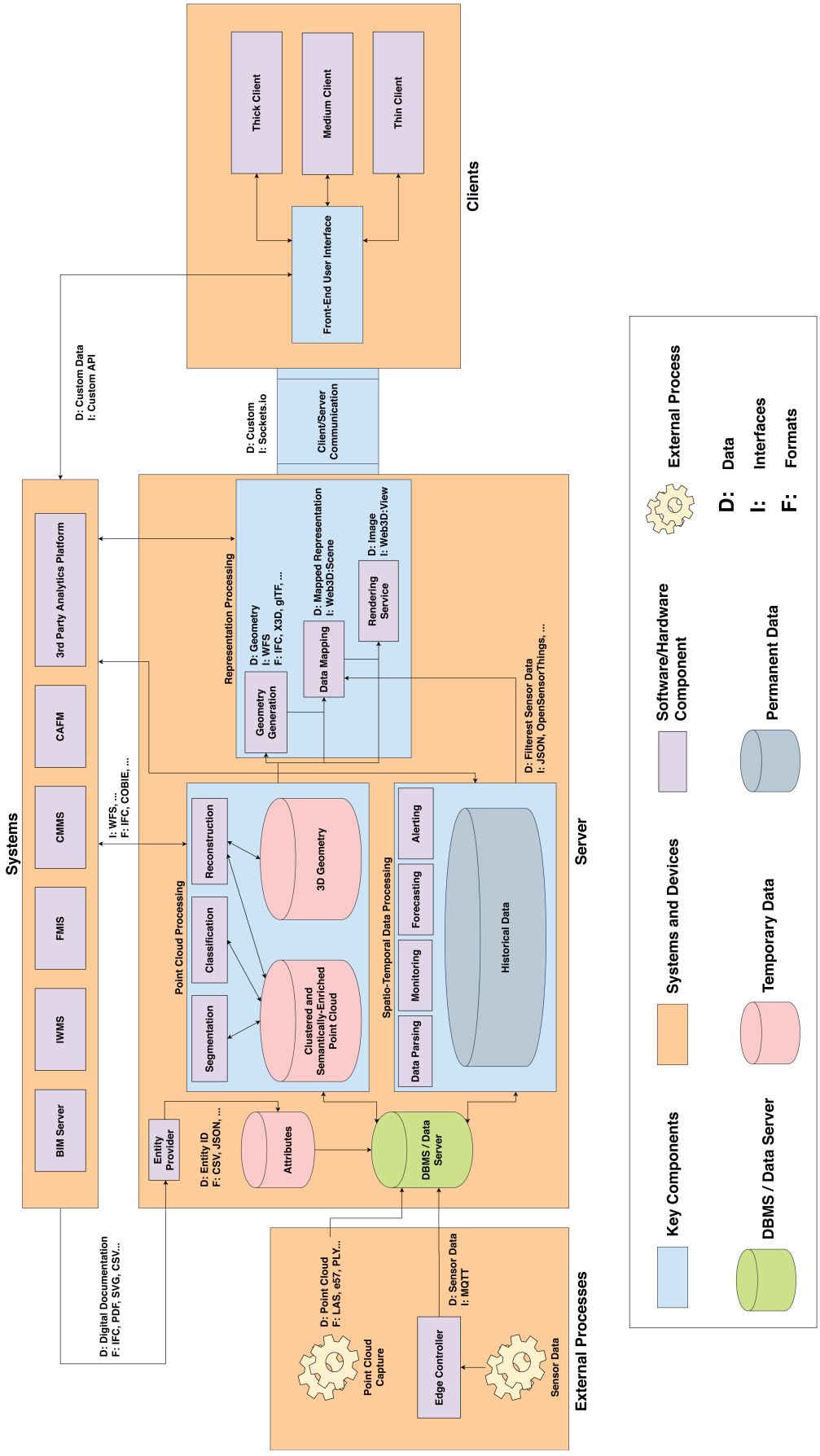


Figure 3.1: *The conceptual reference SOA.*

Client/Server Communication Communication between the server and client requires the use of a bi-directional client/server model. The server and the client should be capable of constantly updating each other about their connection status, and must be able to facilitate appropriate bandwidth for data transfer (in some cases the speed of data transfer should be real-time). With the introduction and standardization of the WebSocket protocol [3], it is possible to accomplish such communication and data transfer tasks. The WebSocket protocol allows bi-directional communication over a single Transmission Control Protocol (TCP) socket, achieving transfer speeds far higher than those available with standard Hypertext Transfer Protocol (HTTP) POST and GET methods. As such, it has been included in various implementations and forms the back-bone of modern full-stack web technologies e.g., Node.js[4].

Node.js is a server-side JavaScript run-time environment, designed to offer cross-platform support for processing tasks that are not always suitable for client-side processing as well as interfacing with various database systems, standards and protocols. The use of Node.js enables the creation of an Express server[5], which uses the Socket.IO[6] JavaScript library (a variant implementation of the WebSocket protocol), in order to establish real-time and uninterrupted communication between the client and the server. With this setup, the server is able to process requests from the client web-application, using real-time bi-directional communication, and enable transfer of data and queries processed by the server.

3.4 Key Architecture Components

The key architecture components are used for accessing, parsing and processing the related data and the user queries sent by the client. In most cases they are implemented as either JavaScript classes and executed by the Node.js run-time environment, or are command-line applications that were implemented using a third-party language (e.g., C++, Python, Java, etc.) – but are invoked and used by Node.js via parsing of command line parameter values and generated results stored on the server.

3.4.1 Point Cloud Processing

The *Point Cloud Processing* component is responsible for the processing and generation of semantics for enriching *as-is* indoor point clouds. It includes three sub-components, which can be used individually or combined – in order to segment, classify and reconstruct a given point cloud. The *Segmentation* sub-component is used to divide and mark similar regions of point clusters, which allows for quicker identification of physical features 4.3.5.

The *Reconstruction* sub-component is used to generate triangular mesh representations of the processed point cloud data. Most of the point cloud reconstruction methods

[3]WebSocket: `https://www.w3.org/TR/websockets/`
[4]Node.js: `https://nodejs.org/en/`
[5]Express Server: `https://expressjs.com/`
[6]Socket.IO: `https://socket.io/`

for BIM applications are based on an automated approach. These are able to detect important structural features e.g., walls, floors, and ceilings [278], along with methods for general 3D mesh reconstruction [152], without additional user input.

The *Classification* sub-component generates *labels* for inputted point clusters, thus enabling the point cloud to be further semantically enriched.

3.4.2 Communication Methods

For querying related data and metadata, an approach following the Representational State Transfer (REST) paradigm, and JSON specification[7] is used. REST is a software architecture used for implementing Web services [77]. The concept of REST relies on the use of an API (known as *RESTful APIs*) between a client and a server based on the CRUD analogy (Create, Read, Update, Destroy) and making use of a URI, the transfer of data (e.g., JavaScript Object Notation (JSON) data), and the API functionality that is usually implemented using standard HTTP methods specification [8] (e.g., common HTTP commands GET, POST, DELETE, etc.). Another alternative to REST is the Simple Object Access Protocol (SOAP) [9], also designed to facilitate communication between a client and a server when implementing web-based services, and using exchange of messages based on the XML schema [10].

3.4.3 Spatio-Temporal Data Processing

The *Spatio-Temporal Data Processing* component is responsible for the storage, analysis, monitoring, querying, and web-based streaming of sensor and related spatio-temporal data. The main function of the component is to interface with sensor hardware and parse the generated sensor data. Sensors can either be connected through the industry standard MQTT[11] protocol for lightweight data transmission, or by issuing HTTP POST requests, with both containing JSON-formatted data as payload.

There are numerous types of communication protocols for sensors based on low-energy, low-bandwidth radio transmission (e.g., ZigBee, Z-Wave, EnOcean, or LoRaWAN), or wired bus connections, and which are suitable for monitoring of environmental phenomena in indoor environments [16]. Low-cost edge controllers can also translate the local sensor data messages sent through low-level protocols into JSON data packets, which can be evaluated by corresponding software components.

Further, live updates of sensor data values can be subscribed using low-latency, bi-directional and non-interrupted communication such as a WebSocket message-based API. Through this, sensor value changes are propagated e.g., to the *Mapping Service* to be visually reflected in the final visualization. Additionally, historical sensor data could be stored in a time series database, which combines efficient access to sensor values as

[7] JSON: https://jsonapi.org
[8] HTTP: https://www.w3.org/Protocols/rfc2616/rfc2616.html
[9] SOAP: https://www.w3.org/TR/soap/
[10] XML: https://www.w3.org/XML/
[11] MQTT 5 OASIS Standard: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html

well as memory efficient data organization. The *Monitoring* sub-component of the SOA evaluates given rules for sensor data and generates events in case of identified matches. For such events, an *Alerting* sub-component can call previously registered notification targets e.g., IWMS or FMIS APIs. These systems would then be able to react to the changes or by triggering automated responses e.g., turning on the air-conditioning if the recorded temperature value is above a certain temperature at a given time of day.

3.4.4 Representation Processing

The *Representation Processing* component provides a set of services for generating the visualization result using point cloud and associated data (e.g., sensor data, floorplans, etc.) – in particular services for *geometry generation*, *mapping*, and *rendering*. The *Geometry Generation Service* is used to create the *as-is* geometry from the semantically-enriched point cloud or reconstructed geometry.

The *Mapping Service* is used to map any additional data values to a given point cloud or reconstructed triangular mesh geometry partition (e.g., sensor data that maps specific attributes such as temperature for a given room). The *Mapping Service* can also be used for assigning specific semantics to data. For example, semantics allow for the association of *labelled* geometry segments with associated data, which is physically located in the real-world counterpart of the 3D representation.

The *Rendering Service* is responsible for generating the final visualization result. The rendering service can make use of low or high-level graphics APIs and frameworks (e.g., Open Graphics Library (OpenGL), WebGL, etc.), in order to generate rendered images (usually in real-time), which can easily be displayed on the client side (either processed and displayed by the client hardware, or streamed from the server).

3.4.5 External Systems and Processes

Any data that is used by the SOA for further processing and semantic enrichment, is given additional attributes via the *Entity-ID Provider* component. This component extracts location and other information from existing building documentation (e.g., longitude and latitude coordinates and room dimensions acquired from *as-built* BIM model, Radio-frequency identification (RFID) or Near-field communication (NFC) tags, etc.), in order to enable automated or manual adding of semantics into the point cloud, and allow for combination with other data sources (e.g, floorplans, sensor data, digital documentation, user annotations, etc.).

These semantics can then be used by stakeholders to identify important physical and location aspects e.g., using the semantically enriched point cloud or other representations (BIM, 3D floorplans, etc). The generated visualization result, and any additional data (e.g., specific location data), are sent to clients. The capture of point clouds is treated as an external process, and can be accomplished using different approaches (Chpt. 4.2). This process is not tied to any specific hardware and software, and in most cases is carried out either by trained FM stakeholders, or is outsourced, with capturing taking place over a given period time with specified frequency.

3.4.6 Data Access and Storage Components

The data access and storage components consist of temporary and permanent data storage (e.g., data stored on the server long term memory), and also a DBMS used for storing and retrieving processed data, semantics, generated results and various attributes used by other software components. The selected DBMS can either be relational or non-relational, but should be oriented towards both spatial and semantics queries. The main difference between relational and non-relational databases is how they store data. Relational databases store data in structured manner, within rows and columns, and each data element is related to a specific row and column. Non-relational databases store data usually in some sort of schematic format e.g., JSON or XML-based schemas.

The main difference between the two database models is how they access data, and how fast this can be accomplished based on a specific user query. Relational databases make use of Structured Query Language (SQL), where data contained in rows and columns is linked and accessed using an indexing scheme (e.g., hash-value keys can be used to represent a pointer to each specific location of data, so access time to data is decreased as the system only has to index the hash-key values, and not the actual data). Non-relational databases are mostly suited towards large amounts of data, as they do not depend on any sort of relational organization of data. This provides the benefit of being able to accommodate complex schema requirements, that may not be clear in the beginning and may change over time (i.e., new attributes are added). Using a non-relational database provides slower data access times though, in favour of flexibility.

A special case of a DBMS that is of particular use to DT applications is the *spatial* DBMS. A spatial DBMS enables querying of specific data types often found in GIS and CAD applications, and allows for more dynamic object-relations to be defined. Specifically they allow the use of spatial data structures (e.g., octrees, *k-D* trees, etc.), in order to spatially discretize complex geometric data and allow for faster indexing. Such databases can either be relational or non-relational, with the main difference being the trade-offs between performance and support for unstructured and structured data versus query features [3]. Additionally, a relational DBMS can be easily extended without rewriting interfaces from scratch, though there are still issues concerning the speed of access to data from user queries (especially large volumes of data) – including enriching such data with custom information due to having to parse and convert changing schematics.

Finally, the method of storing data in such databases needs to be determined. Most databases use an indexing scheme that associates data stored on non-volatile, long-term memory (e.g., hard-disks), thus the access time to such data can be slow. An alternative is to cache a portion or complete set of data in volatile Random Access Memory (RAM), thus increasing the access speed of data. In the presented case studies using variant SOS implementations of the proposed SOA, a database is not used explicitly, but rather the implementations make use of JSON and Comma Separated Value (CSV) files stored locally on the server using a specific directory structure, which is then accessed by the Express server based on queries obtained from the client. In such a case, the approach for data access and management can be thought of as a non-relational.

3.5 Case Study Implementations

With the conceptual reference SOA defined, prototypical SOS implementations based on the SOA are presented and discussed. Prior to implementation of a SOS, the following prerequisites need to be defined:

- The type of data that will be processed (e.g., point clouds, sensor data, 2D images, etc.).
- The hardware configuration of the target client system (e.g., *thick*, *medium* or *thin* clients).
- The processing requirements of the data (e.g., point cloud classification, reconstruction, etc.).
- Processes that will be handled server-side versus client-side (e.g., will visualization outputs be rendered on the client or generated and streamed from the server?).

Once these requirements are defined, the specific processing components, along with data, interfaces and formats can be selected and implemented, based on the conceptual reference SOA (Fig. 3.1). Three prototypical SOS implementations are presented and described. These SOSs were implemented for specific case-studies – focusing evaluating the feasibility of the core processing components required for a DT platform (point cloud processing, classification and spatio-temporal data processing and visualization).

3.5.1 Point Cloud Processing, Analysis and Visualization

For the case study looking at point cloud processing, reconstruction and spatial deviation analysis (Chpt. 5 and 6), a prototypical SOS, with a focus on point cloud processing capabilities was designed and implemented (Fig. 3.2). Point cloud post-processing tasks commonly include computation of normal vectors, segmentation, reconstruction and transformation operations. Once post-processed, the point cloud representation of a specific indoor environment can be further semantically enriched and evaluated (Chpt. 7), or e.g., compared against the finite element version of its *as-designed* BIM geometry counterpart for spatial deviations (Chpt. 6).

Methods for reconstruction can also be implemented e.g., generation of a FDM from BIM geometry can be accomplished with the use of voxelization-based geometry processing methods (Chpt. 5.2). While such processes can be implemented as separate software solutions and components, or using existing software, an integrated solution for such processes is crucial for use in larger and more complex enterprise systems and data analysis platforms i.e., DT platforms.

The main processing components are implemented as command line tools that are executed on the server. The command line tools include segmentation and voxelization tools. The *Segmentation* sub-component makes use of the Point Cloud Library (PCL) framework [211], in order to perform segmentation operations on a given point cloud. The

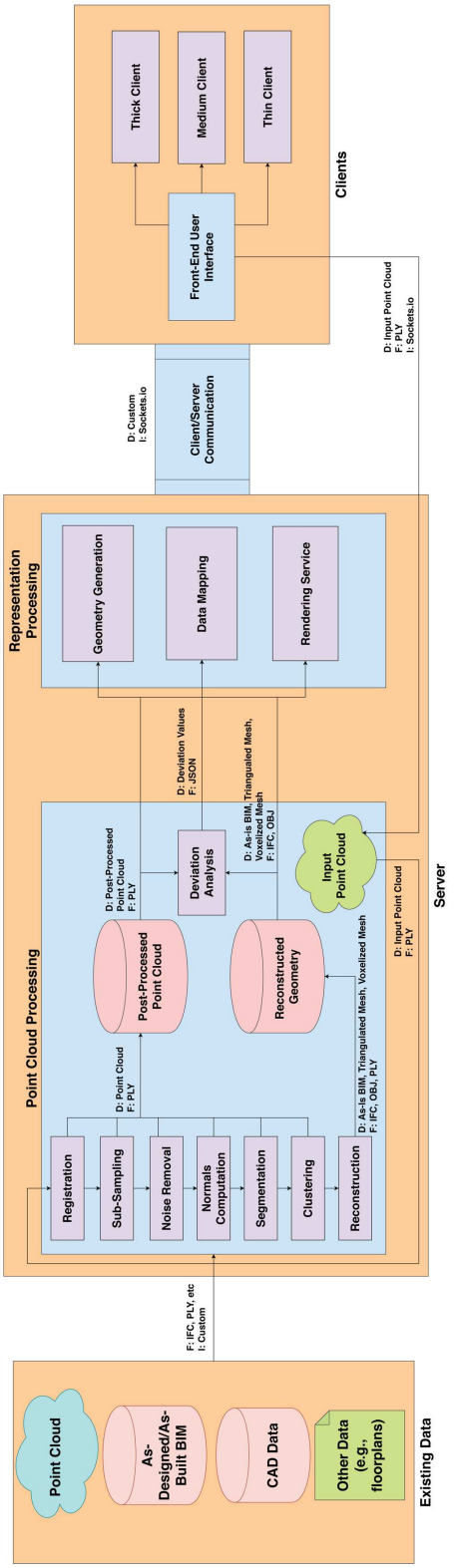


Figure 3.2: *High-level design of the implemented SOS used for processing and visualization of point clouds.*

Binvox voxelization command line tool is used by the *Reconstruction* sub-component to generate a voxelized representation either of reconstructed *as-is*, or *as-built* or *as-designed* triangular mesh geometry [166]. This voxelized mesh representation can then be aligned and compared against the *as-is* point cloud representation in order to highlight any spatial deviations (Chpt. 6).

A custom command-line tool used for segmentation and bounding-box reconstruction based on PCL was implemented in C++, with the ability to directly pass user-defined parameters when invoked. Since compiled C++ programs run faster than interpreted JavaScript on native clients [229], process-intensive tasks e.g., segmentation, can be accomplished faster and have better memory management options (thus being able to process larger point cloud scenes).

3.5.2 Point Cloud Classification and Semantic Enrichment

Service-oriented computation also allows for computation and streaming of classification results from remote servers to clients. The benefit of this approach is that classification results can be viewed on *thin clients*, which otherwise would not be suitable for performing classification tasks due to hardware limitations. The process used in the classification case study (Chpt. 7.8), is implemented as three different system components that capture user control and data input, perform processing both on the client and server-side, and stream the results back to the client (Fig. 3.3).

The client-side data generation system is used to load in the point cloud, generate the 3D partitioning and the 2D images used for classification, and send this to the server for processing. The server-side classification system performs the classification and streams the results back to the *Data Mapping* and *Rendering Service* components. The *Rendering Service* is used to display the classification results to the user as well as to enable further inspection, segmentation, reconstruction and annotation of the classified point cloud.

Using the client-side interface, a user is able to load in a point cloud for classification. The web application then automatically partitions the point cloud using a given scene partitioning approach. These scene partitioning approaches can make use of octree-based partitioning for cubemap image generation (Chpt. 7.5.1), or clustering methods for generating point clusters for optimal viewpoint entropy computation and multiview-image generation (Chpt. 7.5.2), and object-based classification (Chpt. 7.6). The scene partitioning parameters are set through the client-side user interface.

The *Semantic Labelling* component is used to associate the generated classification results with a given point cluster. Prior to classification, each of the point clusters is assigned a ID, which is sent as an additional data attribute along with the point clusters, to the classification components. The generated classification results are then sent back to the *Semantic Labelling* component, along with the original assigned cluster ID, and the component generates the final classification value of the specific point cluster (Chpt. 7.7).

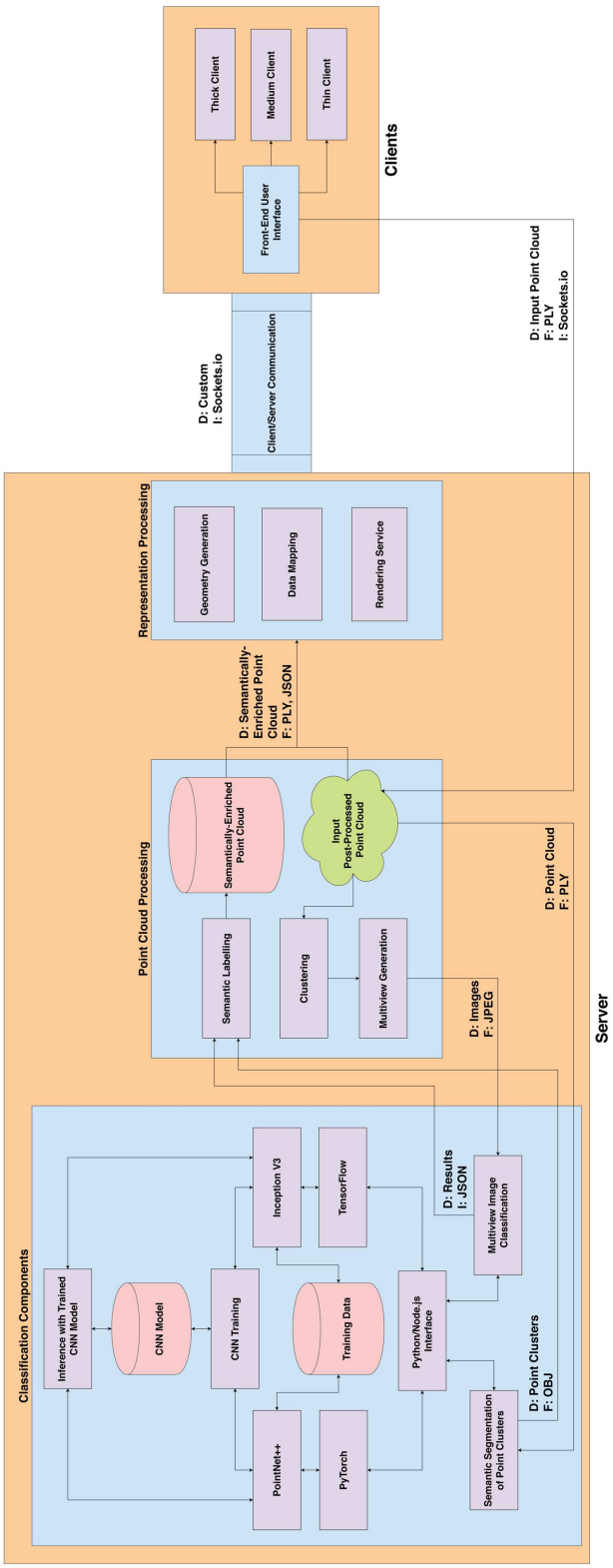


Figure 3.3: *High-level design of the point cloud classification and semantic enrichment SOS implementation.*

3.5.3 Spatio-Temporal Data Processing and Visualization

The prototypical SOS used for the case study of spatio-temporal data processing and visualization was designed and implemented (Chpt. 8.5), using the main processing components (Fig. 3.4). The target client specification for such an architecture is a *medium* client, though provisions could be made to target lower-specification client devices (e.g., if all of the point cloud visualization was computed server-side).

For this specific case-study implementation, JSON files are stored on the server, and used for encoding the processed sensor values obtained for a given time period using the *Data Parsing* sub-component of the *Spatio-Temporal Data Processing* component. These values obtained from the JSON data are then mapped and transformed using the *Data Mapping* sub-component of *Representation Processing* component.

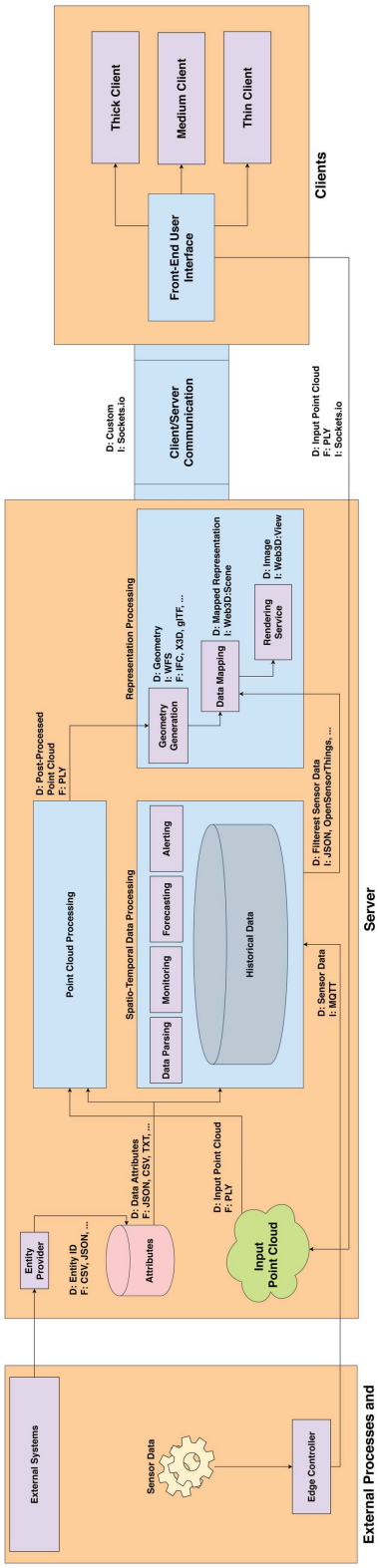


Figure 3.4: *High-level design of the spatio-temporal data processing and visualization SOS, used for processing and visualization of indoor sensor data.*

Chapter 4

Capture and Processing of Indoor Point Clouds using Commodity Mobile Devices

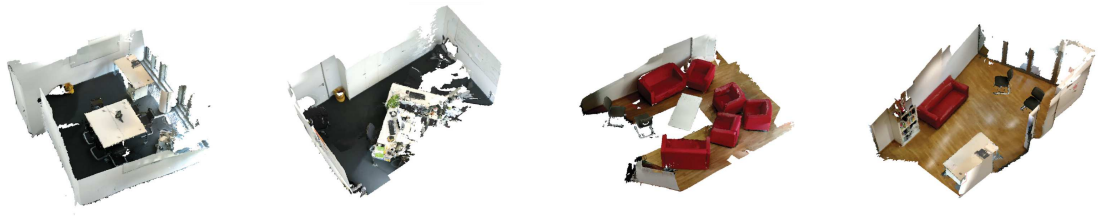


Figure 4.1: *Examples of indoor point clouds captured using a commodity mobile device. These scans contain between 200 000 to 500 000 points, and feature RGB color values in addition to spatial attributes.*

This chapter provides details concerning the selected approaches for capture and processing of indoor point clouds (Fig. 4.1). While there is a wealth of different methods and algorithms that can be used for analyzing point clouds, the selected approaches and algorithms described in this section have been evaluated as being the most appropriate for indoor point clouds. The majority of the point cloud processing algorithms described in this section have been implemented as SOS software components (Chpt. 3.5.1), and tested in the presented case studies.

4.1 Representational Structure and Attributes

A point can be thought of as a vertex in terms of representation within the field of computer graphics. The vertex is the simplest primitive representation in 2D and 3D computer graphics. Formally, it can be defined as a data structure containing at least the position in 2D or 3D Euclidean space, and referenced using a defined coordinate system (i.e., Cartesian coordinates). Additionally, a point can also contain a directional vector (called a *normal*), color information (Red-Green-Blue-Alpha (RGBA) color channels), and also scalar and intensity information (usually one dimensional). The intensity information is commonly associated with point clouds captured using laser scanning,

where the reflectivity of a given surface is encoded rather than its color value. The scalar value can be thought of as auxiliary feature, and can be used to assign additional information to a point cloud post-capture (e.g., estimated surface roughness mapped to given color range). Therefore the point P as is defined as $P := \{XYZ \in \mathbb{R}^3, \vec{N} \in [-1,1]^3, RGB \in [0,1]^3, S \in [0,1], I \in [0,1]\}$.

In terms of computer data representation, a 3D point cloud can be stored in a number of different file formats. While the organization of point cloud data varies amongst different file format specifications, in most cases point cloud data representation can be generalized (Tbl. 4.1).

Attribute	**Data Type**	**Description**
Position Pos_{xyz}	Float32	1
Normal $\vec{N}$	Float32	2
Color C_{rgb}	UByte	3
Intensity I	UByte	4
Scalar S	Float32	5

Table 4.1: *Example data structure for a point cloud. The specific attributes have the features: (1) Position in 3D space e.g., XYZ Cartesian coordinates. Values are usually stored with 32-bit floating point precision can be both negative and positive, (2) Normal vector computed for each point. The direction of the vector is represented by normalized XYZ coordinates and stored with 32-bit floating point precision in the same space as the position attribute, (3) Color values for each of the RGB channels, with an optional opacity channel (A). The color values are stored in the range from 0-255 for each channel, represented as by an unsigned byte, (4) The intensity value of a point of captured. Usually represented as a grayscale value in the range of 0-255 and stored as an unsigned byte, and (5) Scalar value for the point cloud that can be used to add additional numeric information, in most cases using a normalized value range from 0 to 1.*

Common point cloud file formats include PLY[1], e57[2] and LAS[3]. The PLY file format is most commonly used by the SOS implementations for the presented case studies, as it is able to encode positional, normal and color information. It is also supported as a default file format by the Three.js and PCL frameworks (used both for processing and representation of point cloud data for the presented case studies). Specific point cloud data formats can store additional longitudinal and latitudinal coordinates (e.g., the LAS format). Overall, the representational structure and attribute specification of a point cloud depend on use case requirements. The presented processing SOS components are designed to work with indoor point clouds captured using ToF or based on Red-Green-Blue and Depth (RGB-D) data.

[1]PLY: `https://wiki.fileformat.com/3d/ply/`
[2]e57: `http://www.libe57.org/`
[3]LAS: `https://www.asprs.org/divisions-committees/lidar-division/laser-las-file-format-exchange-activities`

4.2 Point Cloud Capture using Commodity Mobile Devices

The primary method for capturing the current physical state of the built environment, advocated in this thesis, is with the use of point clouds. Point cloud representations can be thought of as "snapshots of reality", in a sense that they encapsulate the physical and spatial properties of the space they capture as non-homogeneously distributed points in 3D space [173].

Advantages of using a commodity PCCT-enabled mobile device with depth perception capabilities, including low costs and practicality, has been described by Froehlich *et al.* [85]. Depth Perception allows for devices to determine distances to objects in the real world. Mobile devices with depth perception technology can usually detect objects at distances between 0.5 to 4 meters. A device equipped with such senors can capture an environment with a sensor that casts an infrared dot pattern, mapping and illuminating any nearby surfaces. The sensors measure the time taken for these dots to reach the surface and reflect back (larger dots are farther away) [183]. The size of all the dots are then interpreted as a depth measurement. Combined with an RGB camera (e.g., Structured-Light Stereo (SLS) [120]), the depth and color of each dot can be approximated, and thus an RGB point cloud of the environment can be captured.

Alternatively, point clouds can also be derived from the already triangulated meshes of the captured environment that are generated by the same process such a device. Key technologies used to enable depth perception include RGB-D stereo cameras, and Infrared (IR) sensors and projectors with integrated Structured Light (SL) and ToF technology [170, 47]. Recent consumer mobile devices also included integrated LiDAR sensors [221].

However, there are also certain considerations and limitations to take into account when using PCCT-enabled commodity mobile devices that do not have explicit LiDAR capability. In such cases, the main disadvantage is that captured point clouds may feature lower fidelity in terms of visual details, and contain more noise in comparison to using high-end 3D scanners for capturing.

Investigations into the capturing process presented by Froehlich *et al.* [85] describe how the operator of a PCCT-enabled commodity mobile device (e.g., tablet, smartphone) has to walk around a given indoor space, and sequentially scan each of the surfaces in the area of interest. The average recorded time taken to complete a scan of a $20m^2$ room is approximately 15 minutes. In most cases, the operator would typically start at the entry point to the room and walk around the room in a clockwise or anti-clockwise manner, while scanning items of interest. This would typically first involve scanning the walls, floor and ceiling, than all of the furniture items. Depending on the memory capacity of a given mobile device and the size of a given room/area, multiple scans may be required.

Froehlich *et al.* [85] also noted that a common limitation to this capture approach is that lighting conditions can affect the quality of the capture (e.g., environment is too dark, too reflective or has non-distinctive edge features).

Following this capturing approach, the software application that was selected and used for capturing was the DotProduct Dot3D mobile app[4]. Using the Dot3D app on an compatible mobile device, the user has to point the mobile device camera in the direction of a given area and wait, while the software processes the given frame (Fig. 4.2). The captured 3D point cloud also needs to be filtered for overlapping duplicate points (using the Dot3D app). As multiple scans are required for large areas, the final point cloud representation, in most common use cases, needs to be merged into one single presentation using the partial post-capture scans that have been registered and aligned.

This can be accomplished using specific software tools (e.g., CloudCompare [88]), where multiple partial scans of a given indoor area are parsed and aligned manually with both floorplan images as references as well as visual anchors within the point clouds themselves (i.e., an item that is distinguishable and featured in all related scans, which is used as a reference to transform and correctly align the partial point cloud scans). In most cases, the point cloud can also be sub-sampled to around 100 000 points per $20m^2$, in order to decrease processing time, while preserving the visual fidelity of the original point cloud (Sec. 4.3.2).

Figure 4.2: *Example of the capture process of indoor point cloud featuring common office furniture, using a Google Tango compatible mobile phone with the Dot3D.*

4.3 Point Cloud Processing

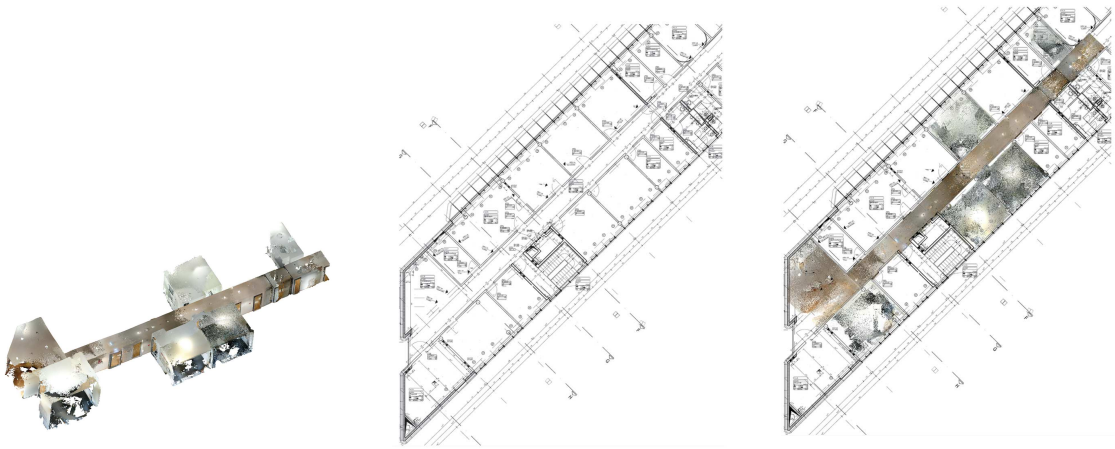
Once captured, point clouds generally need to be processed further before they can be considered useful. This most commonly includes manual removal of visual noise, clutter and other artefacts that may have been captured unintentionally by the device operator, or due to environmental conditions. At the most basic level, a point cloud should be inspected visually, and manually edited prior to any further processing and semantic enrichment. Using such a manual approach, the point cloud is first imported into a point cloud processing tool. Any remaining visual artefacts can then be manually selected and removed by the user when inspecting the point cloud. This step would generally require some domain knowledge of the represented indoor space for the intended function and representation. Domain knowledge would also be required to perform the alignment of the point cloud clusters without introducing spatial errors above a given threshold.

However, point cloud processing can be partially or fully automated, giving the user

[4]Dot3D App: `https://www.dotproduct3d.com/dot3dedit.html`

control over specific processing parameters that would be accessible via a client-side Graphical User Interface (GUI). The user in this case is able to set specific parameters once they load in a point cloud, and send it to the server to for processing. The following sections discuss the most essential point cloud processing tasks. These include: *Registration*, *Sub-Sampling*, *Noise and Outlier Removal*, *Normals Computation*, *Segmentation*, *Clustering* and the representation of point clouds using a spatial data Structure for increased processing performance. Each of these tasks are in turn implemented as SOS software components (Chpt. 3.5.1).

4.3.1 Registration of Point Clouds



(a) *The captured point cloud transformed to undefined coordinates relative to its own coordinate frame.*

(b) *The 2D floorplan featuring the hallway area represented by the point cloud.*

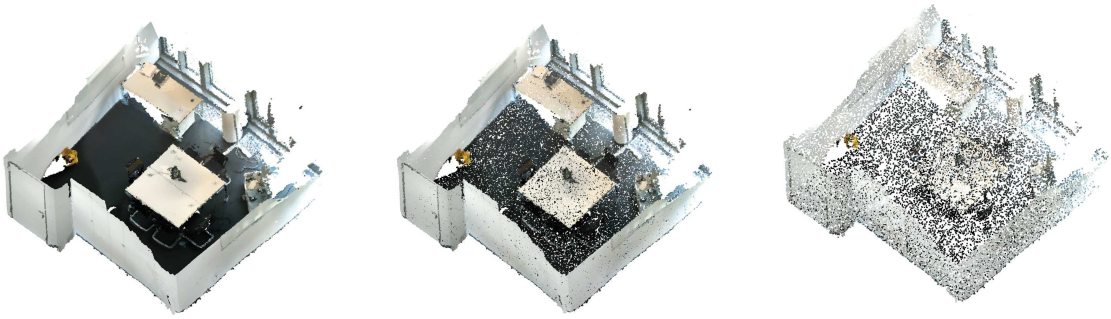
(c) *The post-transformation result of the point cloud registered to the corresponding floorplan layout.*

Figure 4.3: *Alignment of a captured point cloud with an existing floorplan.*

Registration is the process of transforming and aligning two sets of points in 3D space, using their frame of reference to describe their position and orientation. The resulting transformation would use one or more secondary *reference* frames to act as anchors, which in turn would be used to align the original point cloud in the most optimal way. If a given indoor space was captured using multiple scans, then the multiple point clusters can be manually aligned to form the complete indoor space.

Automated methods for alignment are based on an iterative approach, where a number of sequential transformation operations are applied to a given point set in order to optimally align with the reference point set. A common algorithm for point set registration is ICP, using the *Least Squares* regression analysis in order to match the input point set to the referenced one [46].

Another example would be registration of a point cloud to a 2D floorplan obtained from digitised documents or CAD data. In such a case points from a 2D projected point cloud could be iteratively compared with the pixels contained in the line segments of a binary 2D floorplan or image using e.g., edge comparisons [128], or methods based



(a) *Original: 222 084 points.* (b) *Sub-sampled: 108 378 points.* (c) *Sub-sampled: 30 797 points.*

Figure 4.4: *Examples of consecutive sub-sampling of an indoor point cloud using the Random Point Selection (RPS) method.*

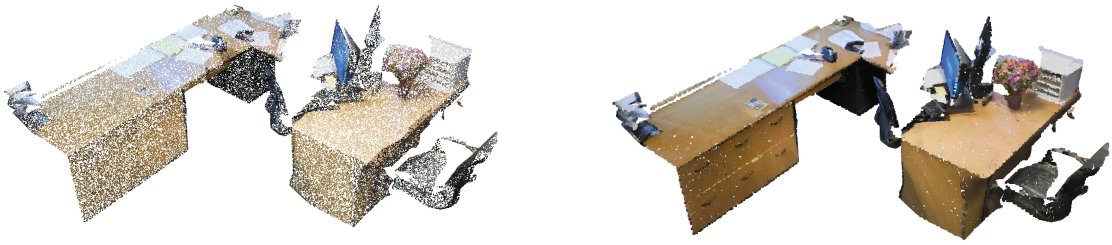
on detection of line segments using the generalized version of Hough Transform [15]. Additionally, it is possible to align a point cloud with existing BIM geometry using an automated comparative approach. Such an approach can either be used for deviation analysis comparisons (based on ICP) [8], comparison between voxelized BIM geometry and point clouds [241] as well as evaluation of graph-based spatial relationships between *as-is* and *as-build* point cloud and BIM components [86].

For the case studies presented in this thesis, a manual registration approach was selected, due to not dealing with very large or complex indoor point cloud scans. Thus, it was possible to align the point cloud by projecting it onto a 2D plane, and using manually entered rotation, scaling and position transformations to align it to the 2D floorplan image projected onto the same plane (Fig. 4.3).

4.3.2 Sub-sampling of Point Clouds

Since point clouds can contain a large amounts of points, it can be beneficial in terms of increasing computational performance to sub-sample them. Sub-sampling in such a case refers to removing points from the original point set, while attempting to preserve the shape of the original point cloud in subsequent versions of the sub-sampled point cloud (Fig. 4.4). This reduction amount depends on the density of the original point cloud, and the distribution of captured points as well as the intended application (e.g., point clouds used for floorplan generation would typically have less points than point clouds used for spatial deviation analysis [240]).

There are three main methods for sub-sampling of point clouds. These include *RPS*, *space* and *octree*-based point cloud sub-sampling methods. Each of the sub-sampling methods are included in the CloudCompare software tool [88], and were used for testing prior to selecting an appropriate method for default use by the prototypical SOS implementation. Variations of the space and octree-based methods were also researched and presented by Moenning and Dodgson [168] and Palomer *et al.* [181].



(a) *Point size 4 - too small, too much white space.* (b) *Point size 8 - optimal.*

Figure 4.5: *Examples of different point sizes for rendering of point clouds.*

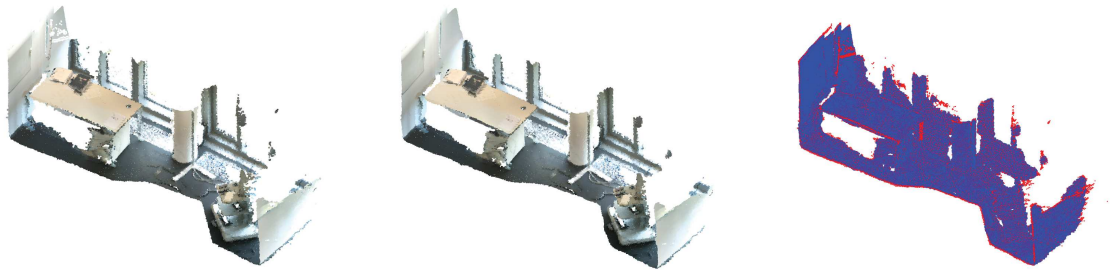
The RPS method takes in the desired percentage of points to keep from the original point set, after which it randomly selects and removes points until the approximate number of points based on the selected input percentage are remaining in the point subset. The space-based method takes in the minimal distance value between two points (as a value between 1.0 and 0.0), before creating a new point subset where each sub-sampled point is not within the minimal distance value to the neighbouring point. The octree-based method generates an octree of a given density, where each point closest to the center of the octree node is kept in the new point subset. Therefore the coarser the octree is, the more points that will be removed for the new subset.

Another important artefact of the sub-sampling process to consider is the introduction of more space between points due to the newly acquired coarseness of the sub-sampled point set. The RPS sub-sampling approach was chosen as the main sub-sampling component in the prototypical SOS implementation, as it allows for the approximation of remaining points through simple heuristics.

Selecting Point Size Post Sub-Sampling Certain point cloud processes require point clouds to have an optimal visual density e.g., multiview classification (Chpt. 7.5). Therefore, the selection of an optimal point size (as well as the "shape" - i.e., the texture used represent the shape of a point), for the representation of the point cloud is required post sub-sampling. This means that the selected size and shape of the points does not introduce too much space between points, so that the scene background color (e.g., "whitespace") is not dominant in the generated multiview images (Fig. 4.5).

4.3.3 Noise and Outlier Removal from Point Clouds

Point clouds often contain noise, visual clutter and artefacts as a result of the capturing process, which need to be removed prior to further processing. Methods for removing noisy areas of the point cloud are mostly based on statistical inference. They include common pass-through filtering methods e.g., detecting points whose average distance from neighbouring points is above or below in comparison to partial neighboring, or complete set, of other points. All three of the methods are implemented in the PCL library [211], namely as bounding box filtering, Radius Outlier Removal (ROR) and



(a) *Original point cloud.*

(b) *Same point cloud with SOR filtering applied.*

(c) *Difference between points in original (red), and filtered (blue) point clouds.*

Figure 4.6: *SOR filtering applied to an indoor point cloud cluster. In this example, 100 points are sampled for the filter, with the difference between the original (4.6(a)) and filtered (4.6(c)) point cloud illustrated.*

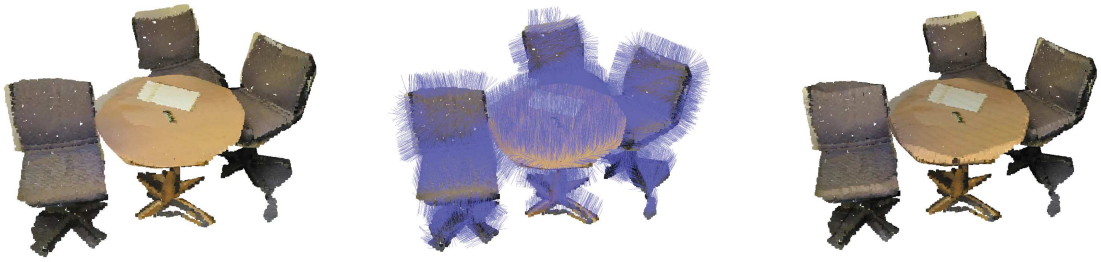
Statistical Outlier Removal (SOR) (Fig. 4.6). Furthermore, research by Skinner *et al.* [228] evaluated all three methods for filtering a point cloud representation of an exhumed water pipe. They note how the ROR and SOR methods both rely on KNN searching in order to detect neighbouring points for evaluation.

Apart from removing clutter and spatially deviating points, captured noisy point clusters can also be denoised using variants of the PCA algorithm. Narváez and Narváez [171] use such an approach for noise reduction in complex point clouds, noting that the use of a variant of PCA is capable of preserving sharp features of post-processed point clusters. Finally, the use of clustering methods can be used to group together with similar spatial distribution of points (usually groups of points forming smaller clusters in comparison to other larger clusters), and remove them. Both the k-means and DBSCAN algorithms can be used to accomplish this (Sec. 4.3.6).

4.3.4 Normal Vector Computation of Point Clouds

Per-point normal vectors approximate the surface of the local point proximity. This allows the point to have an orientation in 3D space (Fig. 4.7), and is thus a very useful attribute for further feature evaluation (e.g., clustering, reconstruction, classification, visualization, etc.). Normal vectors can be computed efficiently by analyzing the local neighborhood of a point. The neighborhood of a point can be defined based on a sphere or a number of nearest neighbors with the KNN algorithm [167]. It is computed using the covariance matrix of the neighbors and corresponding eigenvectors and eigenvalues [109].

The crucial part of normal estimation is fitting a plane for a given sample of neighbouring points, using the *Least Squares* method [69]. From the orientation of the plane, the tangent vector can be computed and used as a normal vector. This is also the standard method used by the PCL framework, and has been further described in detail by Rusu [210].



(a) *Original point cloud without normals.*

(b) *Point cloud with shading applied based on computed normals.*

(c) *Visualization of surface shading using the computed normal vectors of each point.*

Figure 4.7: *Computation of normals for a point cloud cluster. With computed normals, the use of various shading techniques can also be applied to point clouds.*

4.3.5 Segmentation of Point Clouds

Segmentation of point clouds is the process of clustering together similarly distributed point regions. This way, neighbouring points that have similar positional, normal or surface color attributes can be grouped together. Segmentation is crucial for further semantic enrichment, and optimal segmentation of point clouds is important for correct classification, reconstruction and visualization of point clouds.

Horizontal-Slicing Segmentation The first and simplest segmentation method is based on the spatial distribution of points along the projected height axis (e.g., Y-axis), and is referred to as *horizontal-slice segmentation*. It is assumed that the point cloud of an indoor area contains planar walls, which are used to partition the interior space. The horizontal-slicing method generates the 2D horizontal point cloud layers by segmenting the point cloud into three vertical levels – based on the minimum, average, and maximum height of the points in the 3D reference coordinate system (Fig. 4.8).

The vertical levels used for segmentation are selected by computing an Axis-Aligned Bounding Box (AABB) for each of the three levels, and selecting the level with the highest number of points in the AABB for floorplan generation. This method is suitable for floorplan boundary extraction and evaluation (Chpt. 5.3) Any of the segmented layers can be used to generate the floorplan, but usually the mid or top layers are used as they do not necessarily feature any gaps in the wall segments (e.g., door frames). In most cases segmented layers with higher point density are preferable, because they can be used to better approximate possible concave attributes of the floorplan boundary.

Additionally, outlier points are removed from each segmented layer, since they are not part of dense point clusters (with respect to the overall distribution of point clusters). This is typically done using an SOR filter pass on each segmented layer. The maximum height variable used by the horizontal-slicing segmentation method is typically based on the maximum wall height in a given interior space.

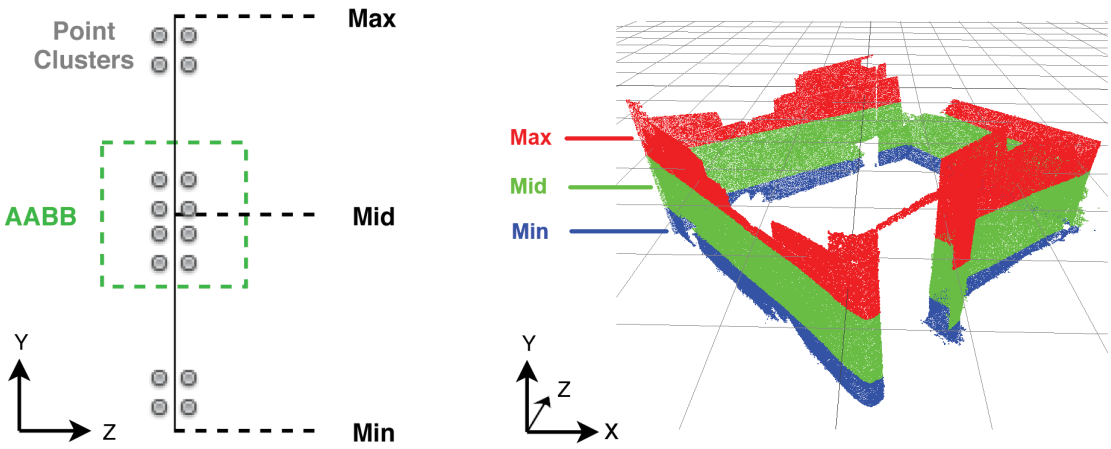


Figure 4.8: *Example of horizonal slicing of a layer for e.g., floorplan generation, using an AABB (left), sampled from the height-based segmentation of an office room point cloud (right).*

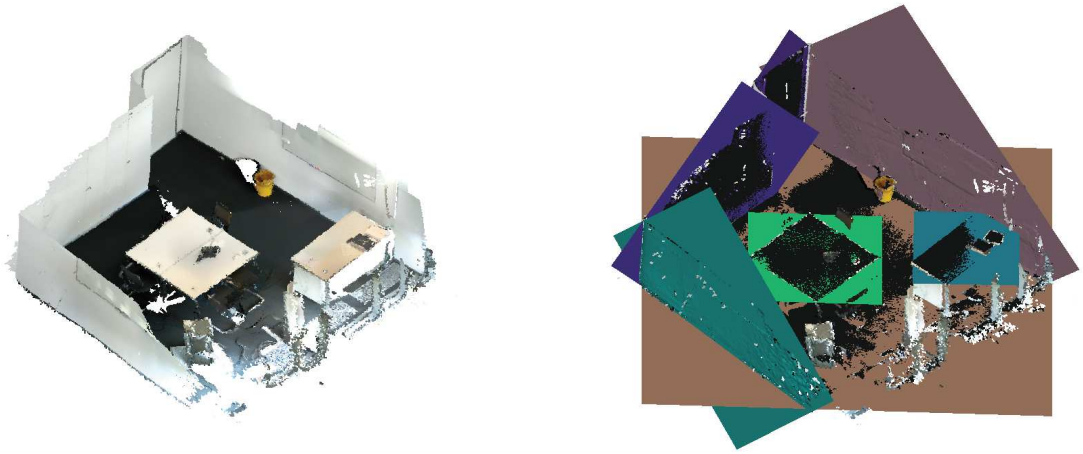
RANSAC Segmentation The RANSAC method is a segmentation method that iteratively attempts fit clusters of spatially distributed points to a given geometric primitive [79]. In the most simplest case, the RANSAC method is able to detect clusters of points that fit to a planar geometric primitive, thus it is useful for detecting walls, floor and ceilings.

The RANSAC method works by sampling a subset of a points from a complete point set (either in 2D or 3D), and attempting to find the best model to fit a geometric primitive to a given number of *inlier* points. This model is then tested against all other points in the set – with the points fitting the model within a given error threshold (e.g., calculated using the *Least Squares* method), and being considered as part of the *consensus set*.

This model generation and fitting is performed a given number of times, until the model is able to obtain enough fitting inlier points as part of the consensus set. Each model-fitting iteration can be repeated with adjusted parameters to determine the "best" model. The RANSAC method can also be expanded to include point fitting comparison tests to other 3D geometric primitives e.g., cubes, spheres and cylinders [218] (Fig. 4.9).

Region Growing *Region Growing* relies on iterative sampling of neighbouring points for determining the segmented point clusters based on their similarity (Fig. 4.10). The measure of point similarity is obtained using either pre-computed point normals [25] (specifically the *surface curvature* value of each point [285]), or color properties [266] of sampled neighbouring points, which are iteratively searched within a voxelized data structure.

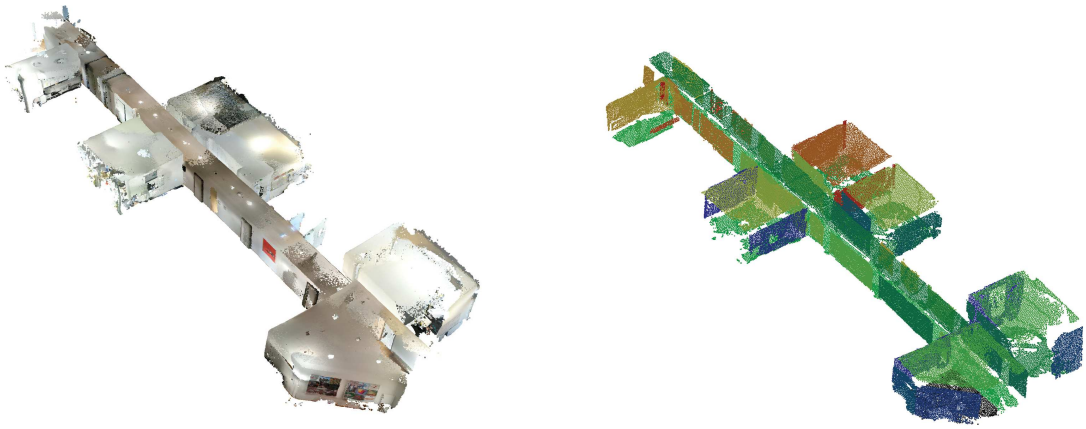
If the region growing is based on point normals, the algorithm first sorts all the points based on their surface curvature, after which the algorithm selects a point with the minimum curvature (e.g., closest to belonging to a flat region), and compares this point normal to the neighbouring one. If the normal angle between the two points is less than the inputted threshold value, the point is added to the region growing point cluster.



(a) *Original point cloud.*

(b) *Computed plane primitives for each of the closest matching point clusters.*

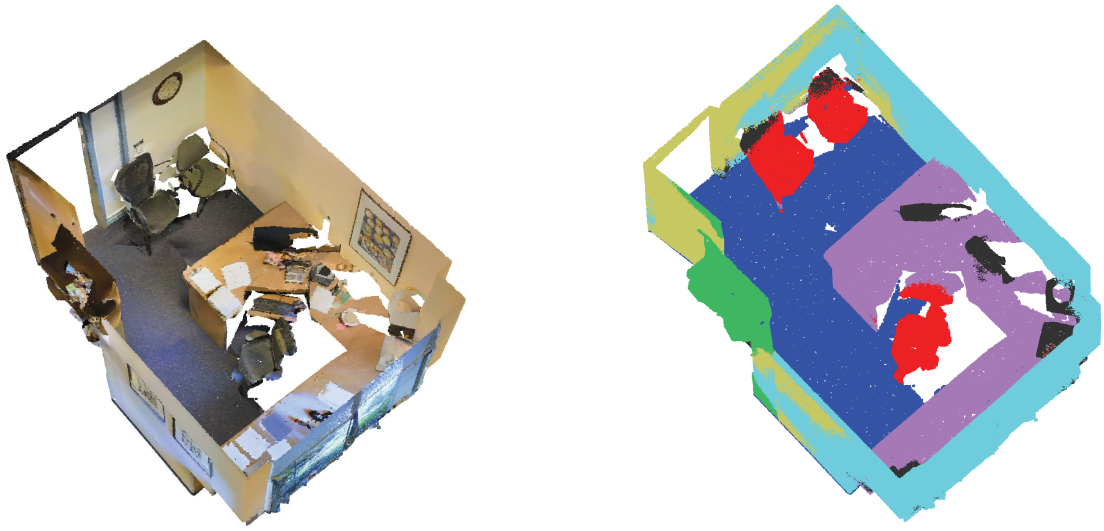
Figure 4.9: *Segmentation results of an indoor point cloud using a variation of the RANSAC-based method described by Schnabel* et al. *[218].*



(a) *Original point cloud.*

(b) *Resulting segmented point clusters.*

Figure 4.10: *Example of Region Growing segmentation applied to an indoor point cloud.*



(a) *Original point cloud.* (b) *Semantically segmented point cloud.*

Figure 4.11: *Example of semantic segmentation applied to an indoor point cloud, where all objects are assigned to a unique point cluster and color – based on the probability of belonging to the class representing the actual object label.*

Semantic Segmentation using Deep Learning Semantic segmentation of indoor point clouds is possible to achieve using CNNs (Chpt. 7.4). CNNs can be trained using a supervised learning approach with example point cloud data that contains semantic labels (Fig. 4.11).

In this way, the CNN learns to detect features specific to point clusters that are labelled as specific objects (e.g., common furniture items – office chairs, walls, doors, clutter, etc.). Once the CNN has been trained using the example point cloud data, it can detect within a given degree of probability point clusters in out-of-core data (i.e., point clouds which have not been used for training), and segment them.

There is however need to use point cloud datasets for training that have high enough visual fidelity and density, in order to extract useful features for training a CNN or other supervised learning models [272]. The use of semantic segmentation based on 3D CNNs is discussed further in Chpt. 7.

4.3.6 Clustering of Point Clouds

Clustering methods originating from data science and statistics have found common use in big data analytics and machine learning, but are also useful for analysis of point clouds. Clustering enables the detection of points with similar spatial distribution and neighbourhood density.

The two main clustering methods implemented as software components in the prototypical SOS implementations are the k-means and DBSCAN clustering algorithms.

(a) *Horizontally sliced floorplan segment (using only 2D X and Y coordinates of the points for clustering).*

(b) k-*means applied to a point cloud segment using 3D coordinates of the points.*

(c) *Use of* k-*means clustering to generate clusters for classification, with the post-classification result shown.*

Figure 4.12: *Examples of* k-*means clustering applied to different point cloud data.*

These clustering methods are further evaluated for semantic enrichment applications based on deep-learning in Chpt. 7.6.

***k*-means Clustering** The *k*-means clustering algorithm is an unsupervised machine learning method used to partition 2D or 3D point sets based on each points proximity to a given cluster *mean* point (Fig. 4.12). The number of means is based on the number of clusters that are required to be generated (usually defined as an input parameter). The method is iterative and begins by calculating the Euclidean distance of each point to each of the means, and assigns each point to a mean that it is closest to, thus forming a cluster [155].

A second update step is also performed where the means of each cluster are re-calculated, based on the amount of new points assigned to it. The algorithm converges when new mean values are no longer assigned, since no new points are assigned to the clusters whose centroids are obtained by computing the mean value. It is expected for the algorithm to converge to a solution after a given number of iterations (size of iterations depends on the density of the point cloud and number of desired clusters).

DBSCAN Clustering The DBSCAN algorithm clusters points together that belong to areas of similar spatial distribution. The algorithm uses the measure of a radius of a given point in a set to determine if any other points are within this radius, before moving onto the next point and checking the same condition. Points that are contained in the radii of neighbouring points are clustered together (Fig. 4.13).

A minimum number of points needs to be set when initializing the algorithm, and this defines the minimum number of points that must be within the radius of an initial point. The minimum number of points that are *reachable* or within the radius of the sampling point are called *core* points. Points that are reachable but themselves cannot reach any other minimum number of points within their radius are defined as *non-core* points and form the edge of a given cluster. All other points are considered outliers, and are not included in the given cluster approximation.

(a) *Applied for clustering of common office furniture objects.*

(b) *Applied to a more complex scene indoor scene.*

Figure 4.13: *Examples of DBSCAN clustering of point clusters representing common office furniture. The use of DBSCAN clustering algorithms enables the generation of AABBs for each of the point clusters, which is useful for segmentation applications.*

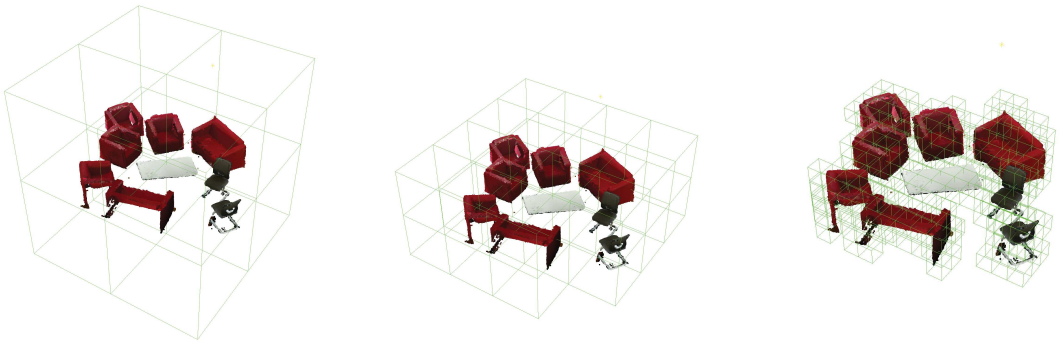
Comparison of Clustering Methods for Indoor Point Clouds The two clustering methods are used as clustering software components in different prototypical SOS implementations e.g., for detection of points for secondary boundary detection in floorplans (Chpt 5.3), and for clustering of objects in point cloud scenes for classification (Chpt 7.5.2).

The DBSCAN algorithm is preferable for point clouds that contain more defined spatial divisions between the furniture objects. Otherwise, the use of k-means clustering is suitable for clustering cluttered scenes or scenes where no clear spatial divisions are present (e.g., if chairs are tucked in partially under the tables).

The DBSCAN algorithm is sensitive to the density of the point cloud, and the number of generated clusters depends on the user parameter inputs that need to be configured for each given scene [7]. For DBSCAN, the minimum number of sampled points is one by default. Alternatively, k-means clustering can be used by setting a default number of clusters and increasing the number of cluster segments to the 3D point cloud with each iteration until a stopping condition is reached [138], or heuristically with the user determining how many clusters are required by visually inspecting the given 3D point cloud. The number of required clusters can also be determined using numerical methods [247].

4.3.7 Data Structures for Point Clouds

The use of space partitioning for point clouds is beneficial for increasing computational performance. For most processing tasks, it is necessary to partition a point cloud into smaller clusters that can be accessed without the need to traverse the entire point set. Typical space partitioning schemes using *quadtree* [78], *octree* [163] and *k-D tree* [23] data structures can be utilized in order to accomplish this.



(a) *Initial octree partitioning of a point cloud.*

(b) *Partitioning with decreased point sampling size, resulting in increased generated octree nodes.*

(c) *A dense octree partitioning scheme.*

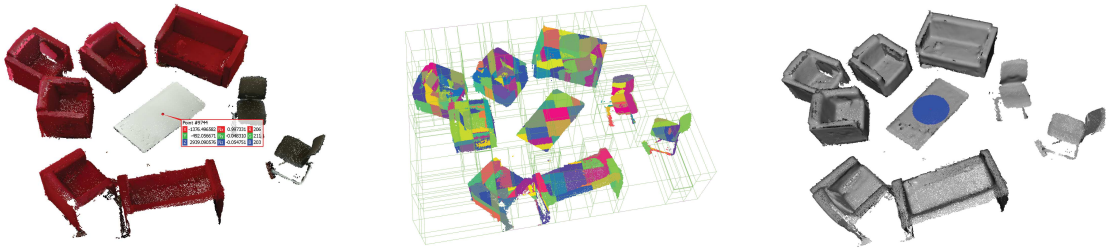
Figure 4.14: *Examples of consecutive octree-based partitioning of an indoor point cloud at varying octree resolutions. For increasing the resolution of the of the octree (e.g., more nodes), a smaller size of points are sampled.*

The use of an octree data structure is particularly useful for partitioning a given point cluster into a balanced discritized representation for processing and classification tasks as well as for optimizing rendering performance for visualizing large point clouds in real-time [219]. Data structures such as *k-D* trees can also be used for acceleration of out-of-core rendering of large point clouds [205], and are particularly suited for nearest-neighbour search operations. The use of an octree data structure was selected for geometry processing (Chpt. 5.2) and multiview classification (Chpt. 7.5), since its density or coarseness can be adjusted depending of the processing needs for a given point cluster as well as providing fast indexing of point clusters for spatial searches.

Quadtrees and Octrees The quadtree algorithm starts by partitioning a given point set into four children nodes, and each of those four children are partitioned further into four more child nodes, until a maximum partitioning depth is reached. The condition for recursive subdivision depends on the application, but for point cloud data the minimum number of points per child node are typically used. In that sense, the quadtree algorithm partitions the point cloud until a minimum number of points are contained in the deepest child node(s) in the tree.

Octrees are an extension of the quadtree data structure, and both methods use a uniform spatial partitioning scheme of recursively dividing a given data set into 2D quads or 3D octants (where each subdivision is composed of eight cube nodes, Fig. 4.14).

While octrees and quadtrees are in most cases surpassed in searching performance by *k-D* trees, they are useful for accelerating the rendering of complex data structures e.g., dense point clouds – by spatially partitioning the point cloud in a way so that each node can be evaluated by the view-frustum of the 3D perspective camera when rendering the scene (nodes containing points outside this frustum are not rendered or can be rendered using sparse representations).



(a) *An initial point used for starting to search for nearest neighbouring points.*

(b) *Illustration of the partitioning scheme using a k-D tree.*

(c) *Result of a nearest-neighbour search, with the search radius highlighted.*

Figure 4.15: *Example of using a k-D tree for nearest-neighbour searching in a point cloud scene.*

***k-D* Trees** *k-D* trees are useful for partitioning 2D and 3D representations of point clouds (Fig. 4.15). Additionally, they can also be used to optimize classification processes using e.g., CNNs [141]. A *k-D* tree splits a given point set as left or right-sided tree branches. Point data is split either left or right side, depending if each value of the chosen attribute (e.g., point coordinate) in the feature set is greater or smaller than the previous value in the tree.

As the algorithm traverses down either the left or right side of the tree, it uses cascading features (e.g., X then Y, in swapping sequence), thus why the *k-D* tree partitioning has both horizontal and vertical lines for 2D representations and hyperplanes for 3D representations of spatial partitioning of point data.

For nearest-neighbour searching with *k-D* trees, the algorithm begins by checking distance from the query point to the root node, and iteratively selecting the "best" (or shortest) distances of points you transverse down the left or right side of the *k-D* tree. If the query point is to the left or right side compared to the root node, the algorithm traverses and searches the side the query point is on. If searching for more than one neighbouring point, a hypersphere can be used. The radius of the hypersphere calculated from the center of the query point, and any points within its radius, are checked for their distances to the query point.

The main benefit of *k-D* trees for nearest neighbour searching is that it uses the tree properties to quickly eliminate large portions of the search space (Fig. 4.15(c)). Specifically, the current "best" point is used to determine if any other areas of the *k-D* tree should be searched - areas where the point could not possibly belong to are pruned.

4.4 Concluding Remarks

This chapter has provided an overview into the most essential point cloud capture, processing and analysis methods. These methods form the foundation for further semantic enrichment, reconstruction, analysis and visualization operations, which are covered in the following chapters. The majority of the processing methods are implemented as prototypical SOS software components (Chpt. 3.5), which run either on the client or the server, and are used to automate otherwise time consuming tasks that require domain expertise.

The presented methods were evaluated and selected after thorough state-of-the-art literature review (Chpt. 2.6). Additionally, the presented methods were also used for the software component prototypes to obtain results for previously published work related to this thesis. The methods provide adequate approximations, and are suitable for processing the complexity of point clouds associated with indoor representations.

Chapter 5

Point Cloud Reconstruction

The process of point cloud reconstruction is used to generate higher-order 2D and 3D geometric mesh approximations of the physical objects based on the shape of a point cloud. Such approximations may include generation of triangulated, voxel or B-Rep geometry. Each of the reconstruction methods generates mesh objects from the point clouds that have use for further analysis and visualization tasks (e.g., deviation analysis, floorplan approximations, *as-is* BIM generation, etc.).

This chapter describes the main point cloud reconstruction methods that were used for specific analysis and visualization tasks – specifically used by software components of the prototypical implementation of the SOS for point cloud analysis and visualization (Chpt. 3.5.1). The presented case studies and results in this chapter are based on the following previously published research: Stojanovic *et al.* [240, 232, 239] and Isailović *et al.* [116].

5.1 Triangular Geometry Reconstruction

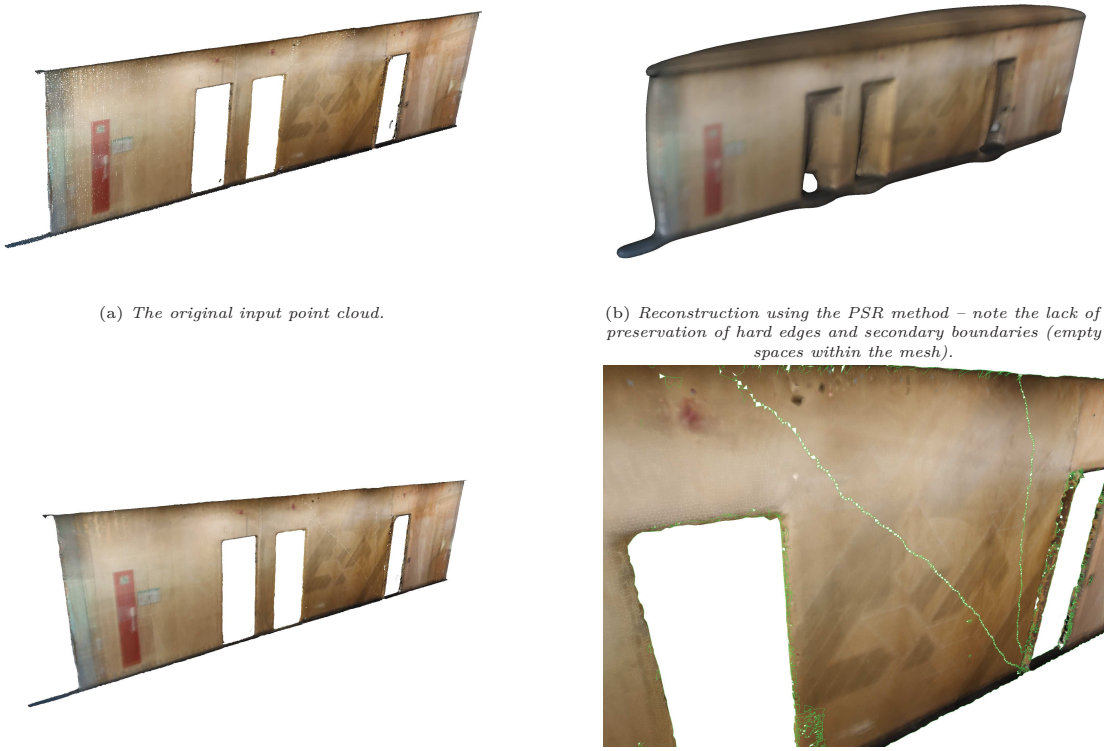
The process of triangular geometry reconstruction from a given point cloud is concerned with approximating and generating triangle primitives for every three vertices found by a selected triangulation algorithm. These vertices are then used to *wind* a triangle primitive, by introducing non-intersecting connecting edges between each of the three selected vertices in a clockwise or counter-clockwise winding direction. The edges of each of the wound triangles are shared between neighbouring triangles, so an in essence a "mesh" network is formed between every $n + 3$ vertices. Normal vectors for each of the point clusters also need to be pre-computed prior to most kinds of reconstructions that require surface approximations.

The two methods used for triangular geometry reconstruction that are discussed and evaluated are the BPA [24] and the Poisson Surface Reconstruction (PSR) [130] algorithms.

PSR was initially evaluated, but it was found not to be suitable for reconstruction for FM and BIM applications, as it does not preserve sharp edges of building elements that are typically found in indoor and outdoor built environments (e.g., non-curved structural features). BPA was in turn selected as the main triangular geometry reconstruction method, as it can preserve the sharper edge features in the reconstructed geometry and can evaluate secondary boundaries (Fig. 5.1(c)). One notable disadvantage of BPA is that

it can also introduce holes into the mesh as a consequence of generating non-manifold edges (Fig. 5.1(d)).

Both reconstruction methods usually require further manual editing of the reconstructed geometry. The resulting triangulated mesh can be exported either in the PLY or OBJ file formats, which allows it to be processed by the *Representation Processing* component of the SOS for further visualization tasks. In most cases, the normals computed as part of the reconstruction process are preserved.



(a) *The original input point cloud.*

(b) *Reconstruction using the PSR method – note the lack of preservation of hard edges and secondary boundaries (empty spaces within the mesh).*

(c) *Reconstruction using the BPA method.*

(d) *Artefacts generated as a result of the BPA method include introduction of holes in the reconstructed triangular mesh, as highlighted by the green edges.*

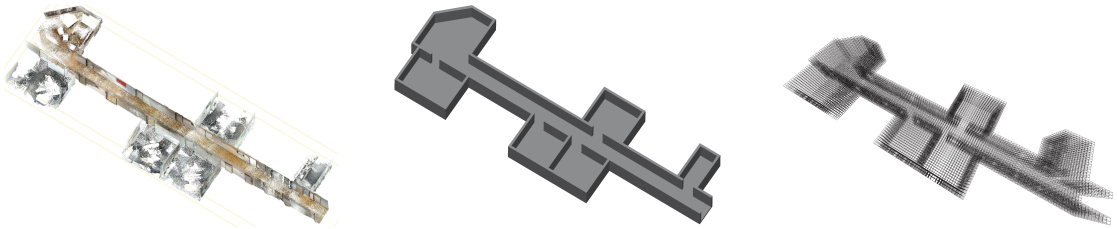
Figure 5.1: *Examples of point cloud reconstruction using the PSR and BPA reconstruction methods.*

5.2 Voxel Geometry Reconstruction

The use of voxelization is important for various geometry analysis tasks e.g., spatial deviation analysis (Chpt. 6). Generally, a voxelized mesh is approximated from a triangulated geometry mesh, where a spatial discretization scheme is used to subdivide the evaluated mesh into smaller connected elements, which are aligned to a 2D or 3D grid structure (i.e., essentially a FDM). This can either be obtained from triangulated mesh reconstruction of point cloud, or from triangulated meshes of existing *as-designed* or *as-built* models extracted from BIM or CAD data.

The *Point Cloud Processing* component is responsible for the generation of the voxelized meshes. The voxelization method implementation used by the *Point Cloud Processing* component is an external command line tool called Binvox [166]. The Binvox voxelization command line tool is used to generate a voxelized representation of either reconstructed *as-is*, or *as-built* or *as-designed* triangular mesh geometry. Voxelization can approximate the general shape of the 3D geometry, including irregular boundaries, by fitting a number of $n \times m \times k$ voxel elements within the given polygonal shape boundaries projected within a 3D grid (usually an octree). The voxel mesh generated by the Binvox tool is exported in the .MSH file format (a simple native file format used by the Gmsh FEA tool[1]), where each center point of a voxel is recorded. The actual size of the voxels is calculated by finding the common difference between XYZ coordinates of each current and next voxel element.

The accuracy of any analysis based on voxles is correlated to the resolution of the voxelized mesh. If the voxelized mesh is too coarse, certain geometry features may be omitted or too simplified to create an accurate enough analysis result. However, the higher the resolution of the voxelized mesh is, the longer it takes to compute the selected analysis result. The resolution for the computed voxel mesh is set by the user, and referred to as the *approximate resolution* - since the voxel fitting algorithm tries to adjust the voxel resolution based on the width, height and depth parameters that are set by the user. A good balance between accuracy and resolution is use-case dependent, but in most cases resolution should be high enough to feature all of the extruded, protruded and non-regular geometry features that are required for specific analysis tasks (Fig. 5.2).



(a) *The captured as-is indoor point cloud.*

(b) *The generated as-is BIM geometry mesh from the point cloud (triangulated).*

(c) *The voxelized mesh result based on the as-is BIM geometry.*

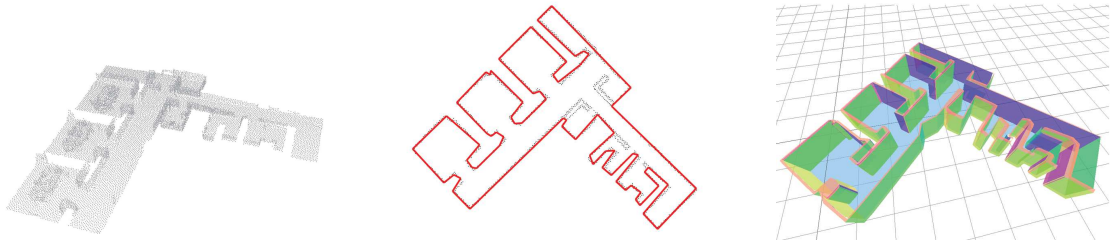
Figure 5.2: *Example of reconstruction of a triangulated as-is BIM geometry to a voxel mesh. The as-is BIM geometry model is in turn generated form an as-is point cloud.*

The voxelization methods use by the Binvox tool are based on the *parity count* and *ray stabbing* methods [176]. The ray stabbing method is preferable for non-organic geometry that has intersecting components (e.g., double walls featured in building models), as the depth sampling only takes into account the initial and final ray sections along a given direction (that in turn is sampled multiple times in different directions for each polygon within a voxel grid). This allows for more accurate generation of complete voxel models (e.g., voxel models where the inside of the model is voxelized, rather than just the *shell* of the model).

[1]Gmsh FEA Tool: `http://gmsh.info/`

The actual visualization of the voxelized mesh typically makes use of geometry data in the form of cube object primitives – where every cube element representing a voxel position is rendered (i.e., a "greedy" rendering method). A more efficient polygon-based geometry rendering method can be used if rendering speed is a requirement for viewing the voxelized mesh interactively [160].

5.3 Approximate Floorplan Generation



(a) *Example 3D point cloud of a building's interior.*

(b) *A 2D vector contour around the horizontal point cloud slice.*

(c) *An extruded 3D mesh representation of the primary 2D floorplan boundaries based on the generated vector contours.*

Figure 5.3: *Example generation of a 2D and 3D floorplan approximation based on a point cloud.*

A floorplan commonly provides an overview used to assess the dimensions of a room and other structural features. Generation of floorplans from existing buildings, especially point cloud representations of the physical environment, pose particular challenges for reconstruction. Reconstruction of 2D floorplans from point clouds requires the use of appropriate shape approximation and line regularization algorithms. Such approximation methods can in turn be used to generate outlines of walls for the approximation of a 2D or 3D floorplan (Fig. 5.3).

Two approaches for reconstruction of floorplans are presented. The first approach extends on the detection of regularized 2D shape boundaries, where the vectorized paths of such boundaries are used to extrude 3D shapes. This approach is known as *Regularized Boundary Approximation*. The second approach makes use of segmentation of planar distributions of point clusters, using the Region Growing algorithm (Chpt. 4.3.5), in order to detect point clusters whose AABB or Object-Oriented Bounding Box (OOBB) elements can be extracted and used as basis dimension and position descriptors for generating higher level geometry representations (e.g., B-Reps used for IFC geometry representation).

5.3.1 Regularized Boundary Approximation

The approach for approximate generation of floorplans from point clouds is based on detection, regularization and extrusion of Primary Boundaries (PB) of an indoor point clouds as well as detection and approximation of Secondary Boundaries (SB) using CSG operations. This approach is based on the previously published research in [240].

The main challenges include selection of appropriate point cloud vertical segmentation layers (referred to as *horizontal slices*), evaluation and regularization of PBs and SBs, and generation of vector paths used for 2D vector image and extruded 3D mesh representations.

Horizontal Slice Segmentation The *horizontal slice segmentation* method described in Chpt. 4.3.5 is used to generate the point cloud segments used for evaluating the PB and SB for approximate floorplans. A selected horizontal slice should be of suitable point cloud density. Outlier points are removed (either manually or using SOR), as they are typically points that are a certain threshold distance away from the main point cluster and are not part of dense point clusters (with respect to the overall distribution of point clusters). The maximum height of the point cloud is typically based on the maximum wall height of the points in a given room scan.

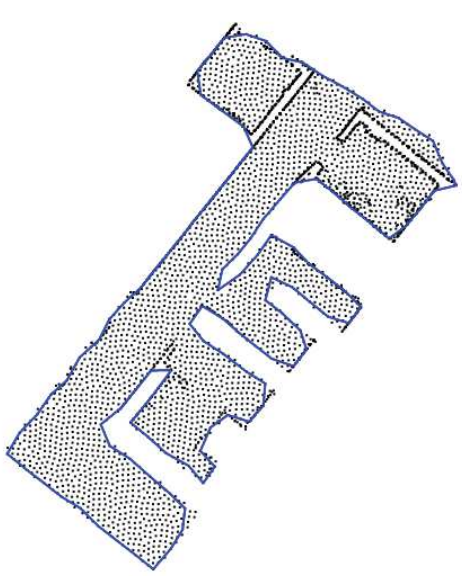
The final step extracts the X and Z Cartesian coordinates of the points from the segmented horizontal layers, and these are exported as a CSV data file, which is then loaded in the by the *Reconstruction* sub-component – for both 2D and 3D PB and SB detection, regularization and approximate floorplan generation.

PB Detection and Regularization The first step for PB detection is the approximation of the concave hull defined by a consecutive distribution of points, which define the general shape of a wall of an indoor space (Fig. 5.4). This data is obtained from parsed values of the CSV file generated in the previous step. The generation of the concave hull is accomplished using the *Gift Opening* algorithm described by Rosen *et al.* [207]. The algorithm works by iterative calculation of the convex hull using the standard *Divide and Conquer* algorithm [142], before converting the best point candidate set into a concave hull (by means of convex hull edge collapsing). In order to capture the accurate concave shape of points that define walls, the Gift Opening algorithm creates usually very noisy convex hull outlines. In order to smooth and rectify these concave hulls for further approximation as floorplan data, the use of line regularization algorithms is required.

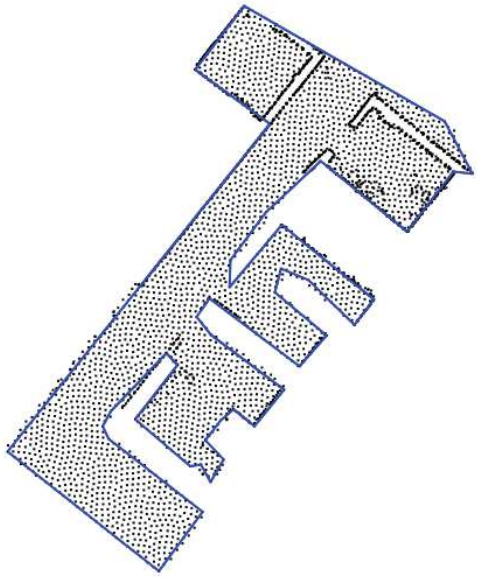
Each of the presented line regularization algorithms have their strengths and weaknesses when attempting to regularize the approximated line elements of the concave hull. In the software component implementation, the regularization of the generated concave hull lines is accomplished by a combination of the *Douglas-Peucker* algorithm [64], the *Visvalingam* algorithm [264], and *radial distance simplification* [143] (Fig. 5.4(d)).

By themselves each of the line regularization algorithms have specific features and constrains (Fig. 5.4(a)-5.4(c)). The *radial distance simplification* method does not smooth out very noisy lines, while the *Douglas-Peucker* algorithm can over-evaluate straight line elements that have breaks in them, thus introducing dents in the line regularization. The *Visvalingam* algorithms preserves the overall shape of the concave hull, but introduces smoother corner evaluations. Selection of the line simplification algorithms was established after reviewing existing implementations [224], and testing the selected algorithms on 2D point cloud slices.

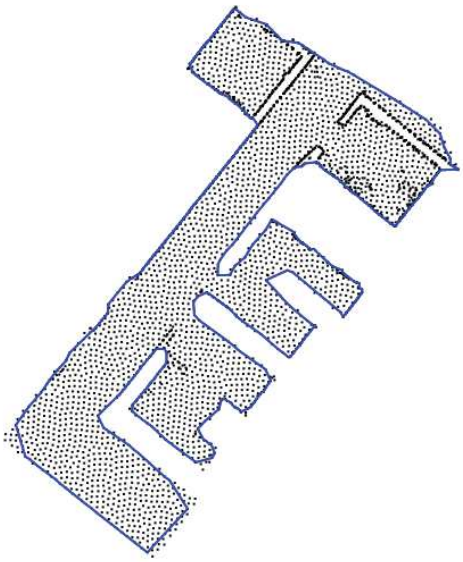
SB Detection and Regularization One particular challenge of generating 2D vector floorplans is detecting smaller closed boundaries inside the PB of a given floorplan partition. For example, the 2D horizontal point cloud slice may contain the walls of a



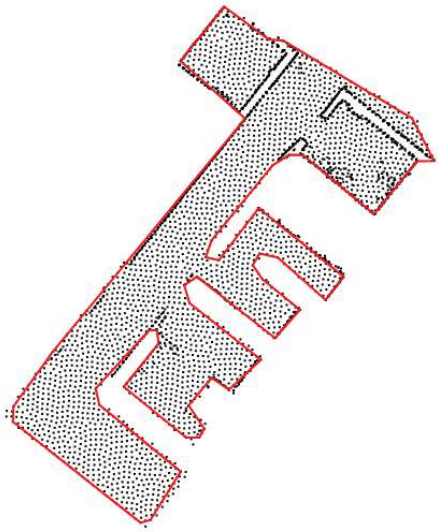
(a) *Regularization generated using the radial distance simplification method.*



(b) *Regularization generated using the Douglas-Peucker algorithm.*



(c) *Regularization generated using the Visvalingam algorithm.*



(d) *Chosen combined method using all three methods to achieve desirable line regularization.*

Figure 5.4: *Examples outputs of three different line regularization methods that were evaluated (radial distance, Douglas-Peucker, Visvalingam and combined). The epsilon values were based on the average distance between the points in the given horizontal slice partition.*

(a) *The SB region defined by a cluster of points, contained within the PB region (blue outline).*

(b) *Using* k*-means clustering the SB region is separated as a unique cluster.*

(c) *The concave boundary is generated and regularized, but includes points from the PB which need to be removed.*

(d) *The correct boundary is generated after all innermost points of the SB are detected.*

Figure 5.5: *Illustration of the SB detection and generation process.*

smaller room or region within a larger room that is not connected to the PB walls (e.g., an elevator or office cubicle), thus the resulting floorplan slice may contain what can be described as SBs. A challenge arises on how to detect these SBs to generate their regularized concave contours (Fig. 5.5).

The *Gift Opening* algorithm applied for concave generation only works on closed point sets that form a PB, so an alternative approach is required. k-Means clustering can be employed in this case as a method to detect point regions in a 2D horizontal slice that have a higher concentration of points, and to cluster them as potential regions for SB generation (Fig. 5.5(b)).

For the selected horizontal slice, the k-means cluster that contains the highest number of points is assumed to be the cluster that contains points representing the SB. The number of k-means clusters is usually set as the number of secondary boundaries plus one (SBs are determined visually prior to approximation, though other methods exist to heuristically determine a number of required clusters [26]).

The algorithm for SB detection is described in Alg. 1 as the *GenerateSecondaryBounds* function, which passes in the point set from the current k-means cluster of a floorplan partition where the SB is located, along with the PB line segments.

Algorithm 1 GenerateSecondaryBounds() function definition, returns a SB polygon.

Require: $Points$, $Polygon$
  $Points_{average}$ **to** AvgDist($Points$)
  $Points_{center}$ **to** CenterPoint($Points$)
  for $i = 0$ **to** $length(Points)$ **do**
    $Dist$ **to** SqrDist($Points_i$, $Point_{center}$
    if PointIsInPolygon($Points_i$, $Polygon$) **then**
      if $Dist < Points_{average}$ **then**
        GenBoundary($Points_i$)
      end if
    end if
  end for

Alg. 1 computes the average distance between all the points in the cluster with the *AvgDist* function call. This function randomly samples a given percentage of the points in the given cluster (taken as 75%, though this is can be adjusted if required), and returns the average distance between them. The 2D coordinates of the current k-means center point from a given cluster are then obtained with the *CenterPoint* function call. This distance value *Dist* is tested to see if it is smaller than the average point distance, so that outlier points from the cluster are not used for the SB computation. This test is performed in addition to testing if the given point is within the previous boundary polygon. In the first instance of the *GenerateSecondaryBounds* function call, this previous boundary polygon is the PB polygon, but with subsequent calls to *GenerateSecondaryBounds* the previous boundary becomes the last SB to be computed. This is used to refine the SB to a satisfactory boundary shape as determined by the user.

The function *PointIsInPolygon* is used for checking if the point is inside or outside of a polygon. This function casts a ray that is projected horizontally from a given point, and the number of intersections with the edges of the boundary polygon are detected as a switchable Boolean statement (Chpt. 6.3.2). To generate the SB contour, the algorithm makes use of the *isPointInStroke* HTML5 Canvas API function to select only the innermost points of the SB (Fig. 5.5(c)). A stroke thickness of approximately ten pixels is used for each of the polygon lines that the points are tested against (Alg. 2). Through experimentation it was determined that this pixel size captures most points in a line segment (minimum three pixels width).

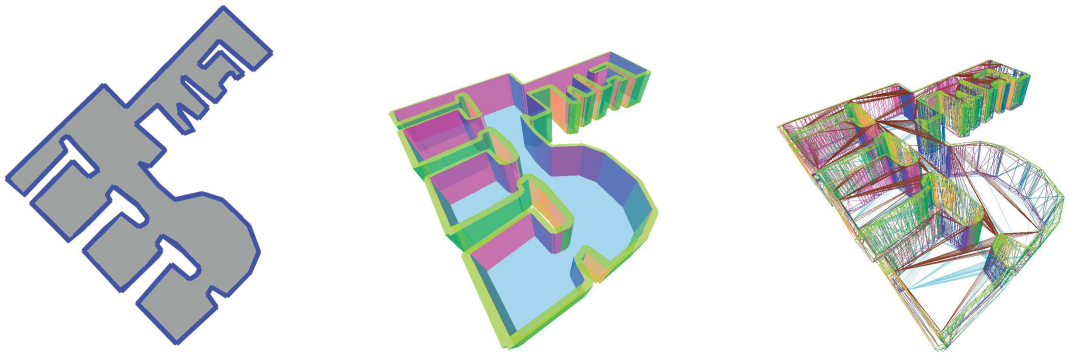
Finally it should be noted that the floor and ceiling parts of a floorplan partition are not selected as horizontal slices nor used for k-means clustering – as such point clusters contain homogeneous and continuous distributions of points, and there would be no distinguishable concentration of point clusters to detect.

Algorithm 2 GenBoundary() function definition, returns a SB points.

Require: $Points$
  $conBoundary$ **to** GenerateConcaveBoundary($Points$)
  $conBoundary$ **to** RegularizeBoundary($conBoundary$)
  $line_{width} = 10$
  for $i = 0$ **to** $length(conBoundary)$ **do**
    $lineTest$ **to** Line($conBoundary_i$, $conBoundary_{i+1}$)
    for $j = 0$ **to** $length(Points)$ **do**
      if isPointInStroke($Points_j$, $lineTest$, $line_{width}$) = FALSE **then**
        $secondaryBoundaryShape$ **to** $Points_j$
      end if
    end for
  end for

Vectorized Paths The vector line paths that form the regularized PBs and SBs are generated as the main 2D floorplan artefact. Additionally, these vector paths are used later on for extrusion and CSG operations of the 3D mesh version of the floorplan

approximation. The HTML5 Canvas API is used to draw line strokes with varying widths, forming a line contour along each of the points. These contours are then used as the basis for the boundary representation. These vector contours are then exported as Scalable Vector Graphics (SVG) vector paths[2]. One disadvantage of using the SVG file format is that any geo-references contained in the original 3D point cloud must be included separately (usually as JSON data), and the corresponding SVG image must be re-projected (if specific geo-referencing is required).



(a) *The regularized PB vector paths (blue).*

(b) *Generated 3D mesh based on the regularized PB vector paths.*

(c) *Wireframe of the generated 3D mesh, showing the geometric complexity.*

Figure 5.6: *Example of a 3D mesh approximated from the regularized PBs.*

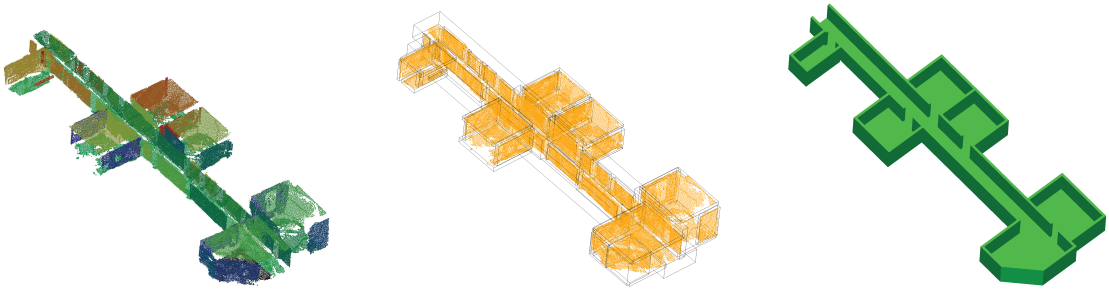
Approximation of the 3D PB and SB Shapes The vectorized line paths from the 2D floorplan approximation are exported into the SVG file format, mainly since it is supported by Three.js for 3D vector path extrusion (Fig. 5.6). Additionally, other popular vector-based image formats that could also be used e.g., Shapefiles[3] and JSON-encoded vector data. The PB is extruded as a 3D mesh based on the height of the walls from the original point cloud.

With the use of CSG operations, the extruded PB 3D volume can be subtracted, added or intersected with any SB 3D volumes using Boolean geometry operations. CSG operations can be implemented using Binary Space Partitioning (BSP), where each triangular mesh used as a primitive is first converted to a BSP node prior to performing constructive operations [192]. The BSP tree is then traversed and the new polygon faces are approximated using polygon splitting – where each vertex belonging to each edge of polygon is tested to see if it is inside, outside, or co-planar to the next polygon in the hierarchy.

The Boolean operations of subtraction, addition, and intersection of elements are represented as a series of function calls, which clip and invert each polygon in the BSP hierarchy to approximate the final CSG result. This allows for the approximation of triangulated 3D floorplan meshes, with variable wall thicknesses and interior SB areas that are not connected to the primary wall boundary geometry.

[2]SVG: `https://www.w3.org/Graphics/SVG/`
[3]Shapefile: `https://www.esri.com/library/whitepapers/pdfs/shapefile.pdf`

5.4 Bounding-Box Approximation



(a) *Point clusters generated using Region Growing segmentation.*

(b) *Resulting OOBBs can then be exported and used to generate e.g., an as-is BIM representation.*

(c) *Example of as-is B-Rep BIM geometry generated from the approximated OOBBs.*

Figure 5.7: *Example bounding-box approximation reconstruction of the main hallway and office areas of an indoor point cloud.*

An OOBB or an AABB representation of point clusters can be extracted and used for generating a 3D floorplan model. These planar clusters are the result of the segmentation process (e.g., Region Growing, RANSAC, semantic segmentation, etc.). The 3D geometry generation involves computing the bounding box for each of the segmented planar point clusters, and exporting them as a text file along with the associated semantic label (Fig. 5.7).

This process is handled by a custom command-line tool implemented using the PCL framework and run by the *Point Cloud Processing* component of the SOS implementation. The resulting clusters are exported and used for further editing, analysis, representation and decision making tasks (e.g., generation of IFC data).

The generated bounding boxes can offer a good approximation of the dimensions of each of the main structural components, but one downside to this approach is that the reconstructed bounding boxes, especially OOBBs, are not always aligned optimally with each other. This requires the use of further geometric evaluation e.g., for example intersecting edges along the vertical axis between two bounding boxes are merged together and a new bounding box is recalculated.

5.5 Reconstruction to Industry Foundation Classes

The use of the IFC file format has in the last 15 years become a standardised exchange format for BIM data within the AEC industries [67]. As a result of this, the generation of *as-is* and *as-built* BIM models has posed a challenge of how to generate up-to-date BIMs without needing to create them manually from scratch, which can be an expensive and time consuming process [265]. The concept of *Scan-To-BIM* has been used in the past decade to describe manual, semi and fully-automated methods for reconstruction of *as-is* and *as-built* BIM data from segmented and semantically-enriched point clusters [250, 178, 104]. The use of semantic enrichment methods e.g., deep-learning based

classification of point clusters, enables the transfer of the clusters dimensions and labels to a corresponding IFC element. The current standard IFC file format specification allows for complete digital description of a building at varying levels of details[4].

Generation of IFC files can either be accomplished manually using specialist software (e.g., Autodesk Revit, Graphisoft ArchiCAD), or can be generated in a programmatic manner. The programmatic approach for generating IFC files is based on existing open-source IFC software libraries. The most notable of this is the xBIM Toolkit, which enables generation of IFC data using C# language bindings [156]. Another custom command-line tool used by the *Reconstruction* sub-component of the SOS is able to parse OOBB dimensions, coordinates and labels (generated by the *Classification* sub-component of the *Point Cloud Processing* component), and use them to generate IFC components at LOD 100-300 representation (using common IFC components e.g., *IfcWall* and *IfcSlab*). The resulting IFC model can then be imported into an existing BIM database of specialist software for further evaluation.

Furthermore, the 3D geometry data stored in IFC files are a major source for comparative representation and clash detection [67]. Since the geometric representations are stored as space partitions, a connectivity graph can be computed to generate a 3D volume of a building partition by linking all of the described nodes as edges [54]. Additional geometric representations besides B-Reps e.g., curves, swept solids, etc., can also be used for geometric and spatial analysis if available in the IFC dataset[5].

5.6 Case Study

The results of the discussed reconstruction methods are presented, with a focus on methods for reconstruction of indoor point clouds containing the core structural features (e.g., walls, floors, and ceilings). The voxelized mesh, approximate floorplan and IFC reconstruction approaches are methods in the *Reconstruction* software component of the prototypical SOS implementation for point cloud classification and semantic enrichment (Chp. 3.5.1). The triangular mesh reconstruction using BPA was obtained separately using the MeshLab software tool [51].

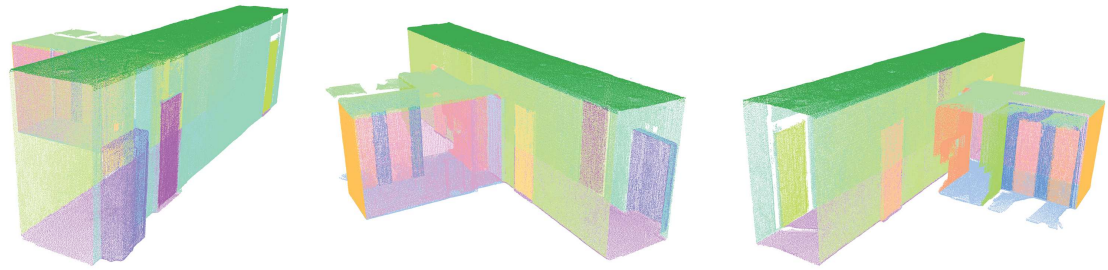
5.6.1 Triangular Mesh Reconstruction

The triangulated mesh reconstruction results are based on the mesh complexity as well as the observed visual fidelity of the reconstructed meshes. The normals of the point cloud were pre-computed prior to reconstruction. The reconstruction results are generated using BPA. The complexity is measured in terms of the number of triangular mesh faces (Fig. 5.8). From the reconstruction results, it can observed that the resulting triangular mesh has considerably high geometric complexity (Fig. 5.8(b)). A possible optimization for this would be to use a sub-sampled point cloud as the input for reconstruction –

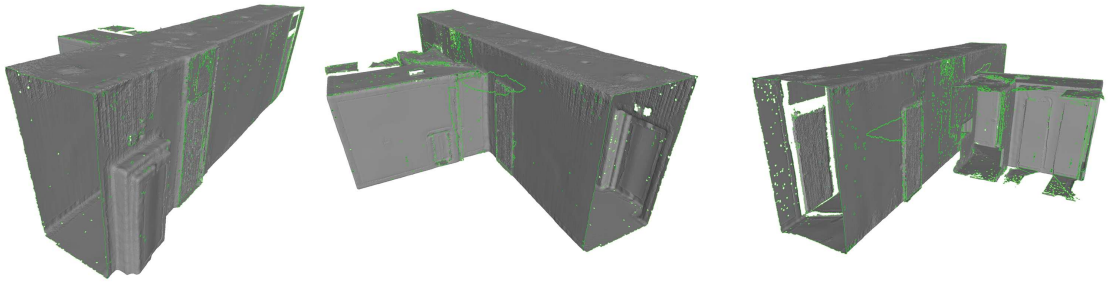
[4]IFC4: https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/

[5]IFC Geometry Representation: https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2/HTML/schema/ifcrepresentationresource/lexical/ifcshaperepresentation.htm

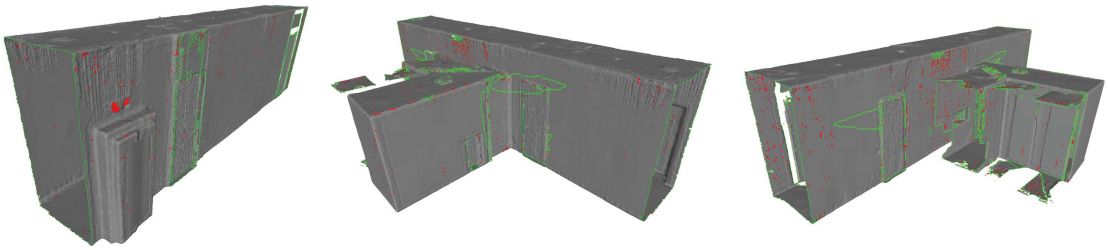
though the preservation of the visual fidelity and original shape (especially the main structural boundaries that are reconstructed as hard edges), would be of importance. Additionally, while the use of a filter to close any holes in the mesh corrects the majority of the artefacts from BPA reconstruction (Fig. 5.8(c)), additional manual editing and reconstruction may still be required for further visualization and analysis applications.



(a) *The input point cloud, containing 464 713 with pre-computed normals.*



(b) *The triangulated geometry mesh reconstructed using the BPA algorithm. Note the holes in the mesh that are generated as using a constant radius of 30.0 units for the BPA sampling (resulting mesh boundaries highlighted in green).*

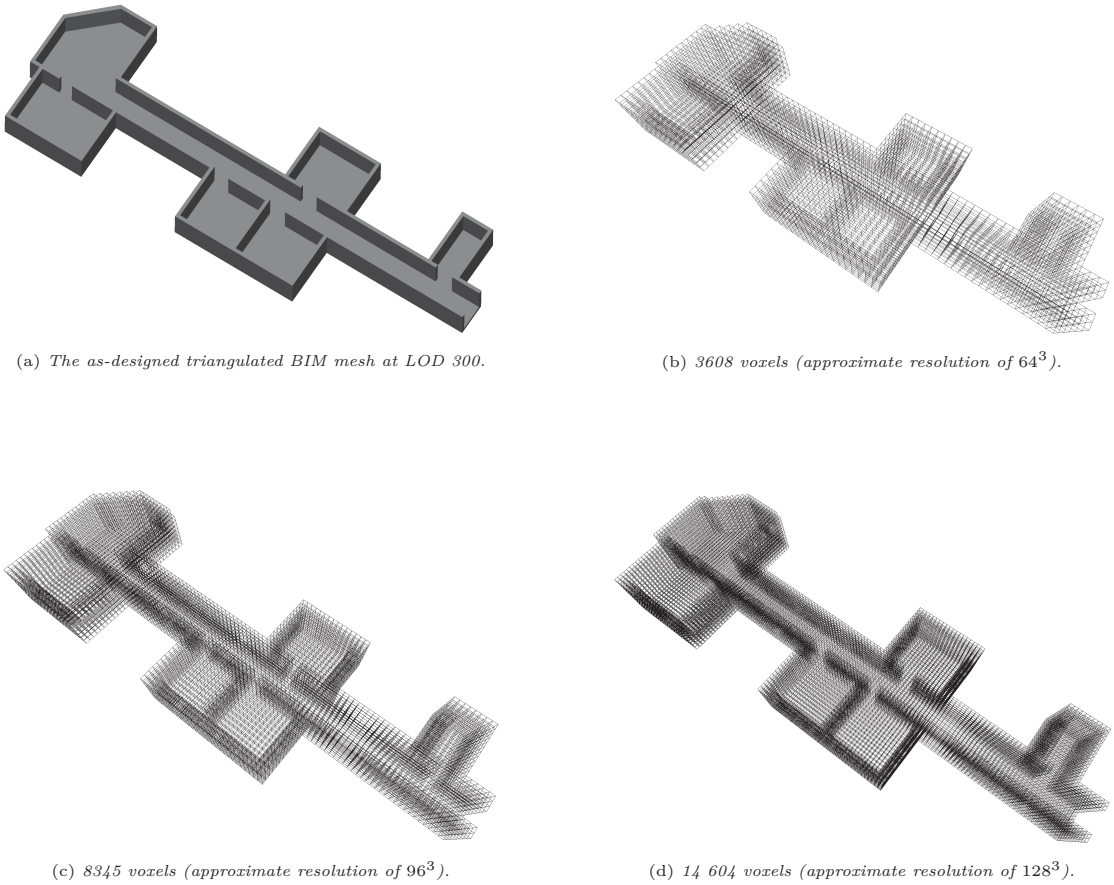


(c) *Post-filtering result of the BPA reconstructed mesh, used to close the majority of the holes in the mesh (faces generated to close the holes in the mesh are highlighted in red).*

Figure 5.8: *Reconstruction of the hallway point cloud using the BPA. The resulting complexity of the reconstructed mesh is 464 713 vertices and 890 416 faces (increased to 895 230 faces post-filtering), with the majority of the mesh holes closed.*

5.6.2 Voxelized Mesh Reconstruction

The comparison of the voxelized mesh reconstruction is based on the results reported in Stojanovic *et al.* [232]. The input mesh used for generating the voxelized meshes is based on triangulated geometry derived from the corresponding as-designed BIM at LOD 300, with three different voxelized meshes being generated at increasing resolutions (Fig. 5.9(b)–5.9(d)). Voxel meshes are also used for spatial deviation analysis (Chpt. 6).



(a) *The as-designed triangulated BIM mesh at LOD 300.*

(b) *3608 voxels (approximate resolution of* 64^3*).*

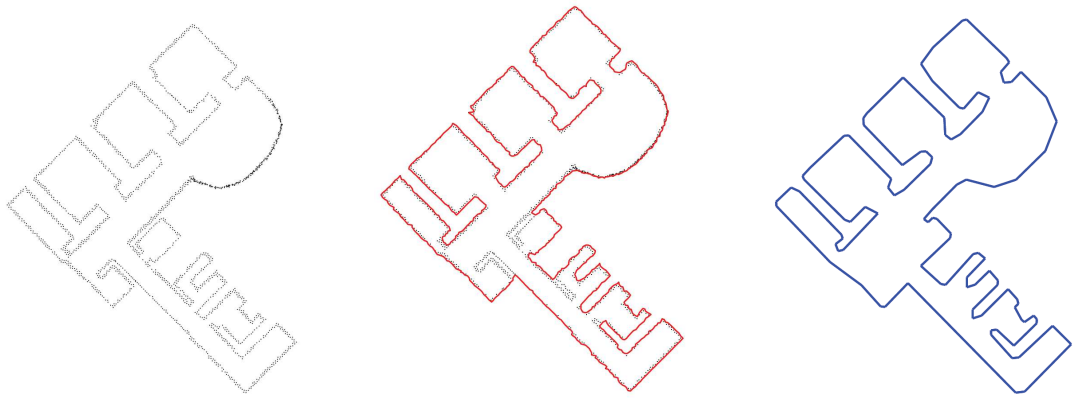
(c) *8345 voxels (approximate resolution of* 96^3*).*

(d) *14 604 voxels (approximate resolution of* 128^3*).*

Figure 5.9: *Example of progressively increasing voxelized mesh, based on the corresponding as-designed BIM geometry.*

5.6.3 Approximate Floorplan Generation

The approximate floorplan generation results are based on the results reported in Stojanovic *et al.* [240]. For the testing of the approximate floorplan generation algorithm, a point cloud featuring a typical office plan was used in order to evaluate the PB and SB approximation both in 2D and 3D. The floorplans and room segments were obtained from an externally obtained 3D point cloud of a building interior, which was scanned using a portable LiDAR scanner. For certain tests (e.g., those for the SB detection and curved surfaces), the point cloud clusters were artificially added. The results from the



(a) *The input horizontal point cloud slice.*

(b) *Generation of the non-regularized concave shape.*

(c) *Regularization of the concave outlines, using a combination of the Douglas-Peucker and Visvalingam algorithms.*

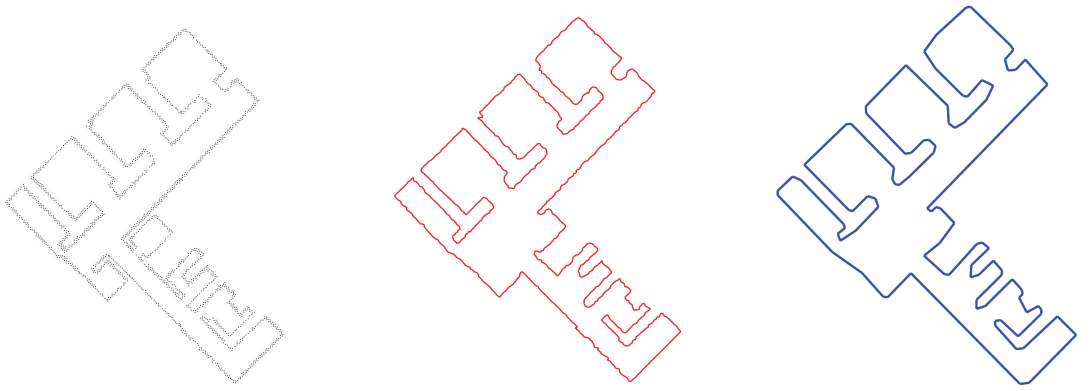
Figure 5.10: *Results of the PB generation for a floorplan featuring a curved wall segment.*

2D PB and SB detection are evaluated against boundaries detected using the α-shape algorithm [70]. The results from the 3D PB and SB detection are evaluated in terms of their geometric complexity (introduced as a result of the applied CSG operations for the detection of SBs).

2D PB Detection The presented results in Fig. 5.10-5.11 show two different versions of a floorplan, one with straight walls and another with an artificially added curved wall section. The results were computed using a non-partitioned floorplan slice, selected from the mid-level height of the point cloud (so no points for the floor or ceiling are contained in the slice). The PB was computed using a two pass filtering combination of *Douglas-Peucker* and *Visvalingam* algorithms. The epsilon value used for the *Douglas-Peucker* generalization parameter was 2.4, while the epsilon value for the *Visvalingam* generalization parameter was 0.2.

2D SB Detection The results for the SB detection presented in Fig. 5.12 were compared against results generated using the α-shape algorithm. The epsilon values for the combined passes of the *Douglas-Peucker* and the *Radial Distance Simplification* algorithms were between 5.0-10.0, while the epsilon value for the α-shape algorithm was set to 150.0. An additional comparison including regularization of the α-shape with a default epsilon value of 10.0 is also included. These metrics were determined using a larger size that was subsequently decreased until suitable approximations were computed.

3D PB and SB Reconstruction Results for the generation of 3D meshes based on detected PB and SB vectorized floorplan paths are presented. The meshes are generated using CSG operations and extrusion, and are not optimized in terms of triangulated mesh complexity. The 3D meshes are generated and visualized using the Three.js 3D web development framework. Results featuring regularized 3D PBs and SBs of a floorplan are presented in Fig. 5.13 and Fig. 5.14.



(a) *The input horizontal point cloud slice.*

(b) *Generation of the non-regularized concave shape.*

(c) *Regularization of the concave outlines, using a combination of the Douglas-Peucker and Visvalingam algorithms.*

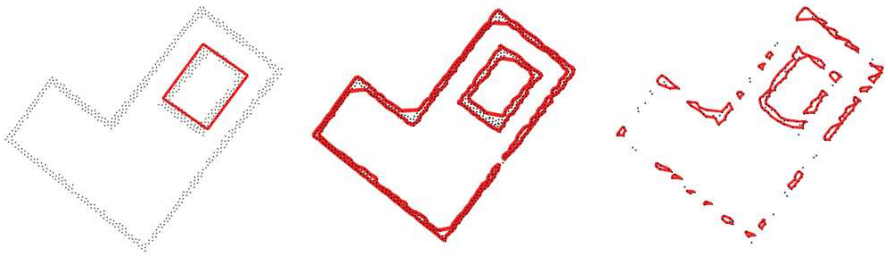
Figure 5.11: *Results of the PB generation for a floorplan featuring straight wall segments.*

5.6.4 Bounding Box-based Reconstruction

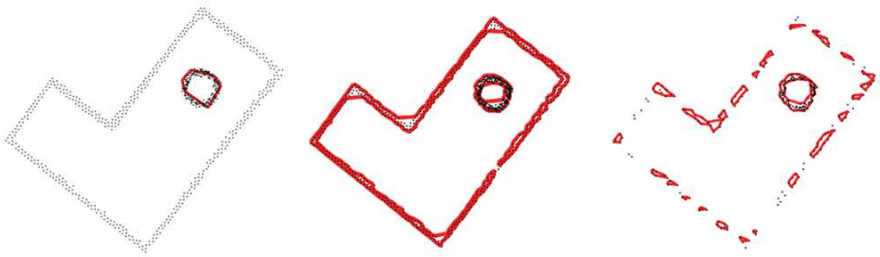
The results of the bounding box-based methods for reconstruction of segmented point clusters are presented in Fig. 5.15. The point clusters were segmented using Region Growing segmentation, and the IFC model was reconstructed using the OOBBs of the segmented point clusters. The results for the IFC reconstruction are based on the results reported in Stojanovic *et al.* [239].

The reconstructed IFC elements (*IfcWall*, *IfcDoor* and *IfcSlab*) are based on the size and position of the bounding box coordinates obtained from the bounding box-based reconstruction method. The resulting IFC can be reconstructed using either AABBs or OOBBs, though AABB representations are only suitable for building layout where all the structural segments are aligned at right angles with each other, otherwise any rotations of elements are not preserved in the AABB representation.

OOBBs can preserve not only the complete volume and position of each point cluster, but also the local rotation related to a given point cluster. That way if the given point cluster (e.g., a wall segment) is intentionally not at right angles to any connecting structural elements, its specific rotation can be preserved. Since the generated OOBBs are generally not correctly aligned, further manual editing is needed in order to create the final IFC model.



(a) *SB method epsilon value: 10.0, α-shape epsilon value: 150.0, α-shape with regularization epsilon value of 10.0.*

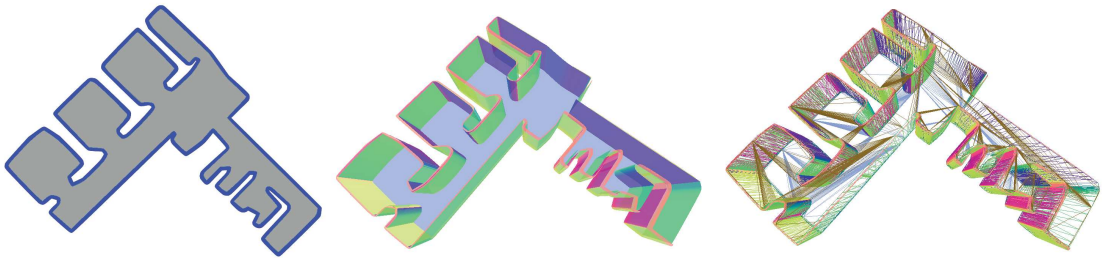


(b) *Uses same parameters as (a).*

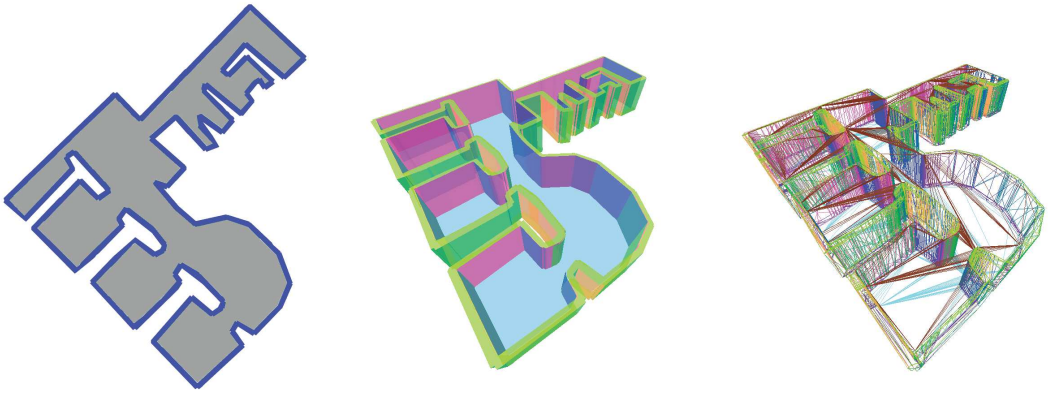
(c) *SB method epsilon value: 5.0, α-shape epsilon value: 150.0, α-shape with regularization epsilon value of 10.0.*

(d) *Uses same parameters as (c).*

Figure 5.12: *Results of SB detection using the presented SB detection method (left), in comparison to SB detection using the α-shape algorithm (center), and the α-shapes with regularization (right).*

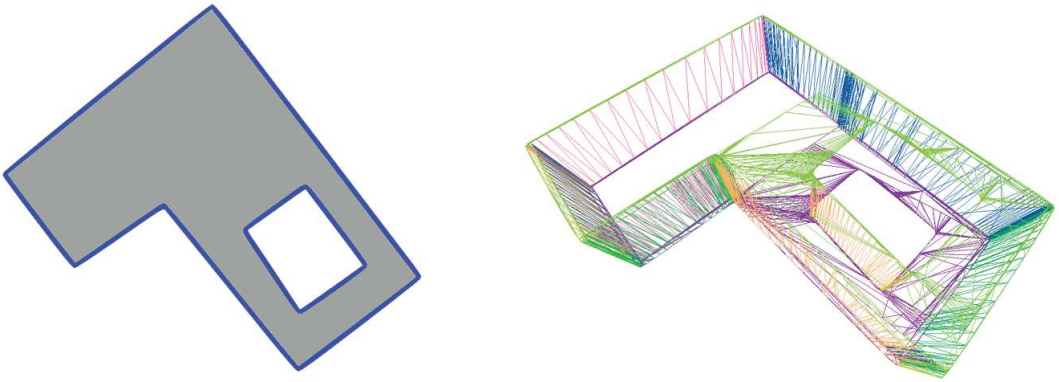


(a) *A generated 3D mesh featuring square elements, with a mesh complexity of 137 106 vertices and 45 702 faces.*

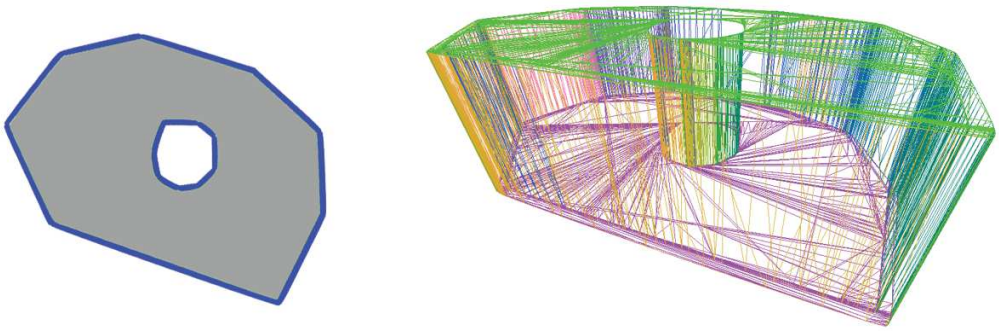


(b) *A Generated 3D mesh featuring a curved wall element, with a mesh complexity of 93 204 vertices and 31 068 faces.*

Figure 5.13: *Examples of 3D meshes of vectorized floorplan paths, generated from regularized PBs.*

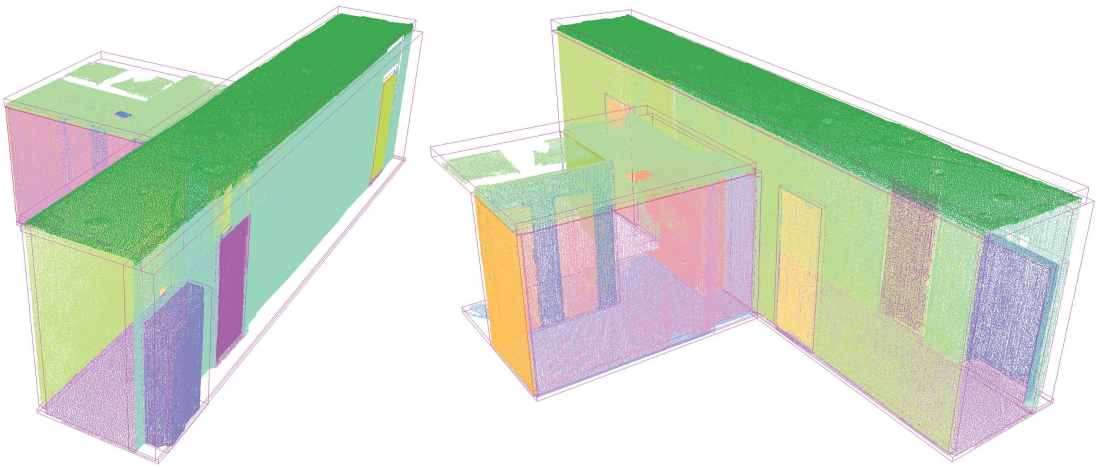


(a) *3D mesh featuring SB in the form of a hollow square, with a mesh complexity of 34 386 vertices and 11 462 faces.*

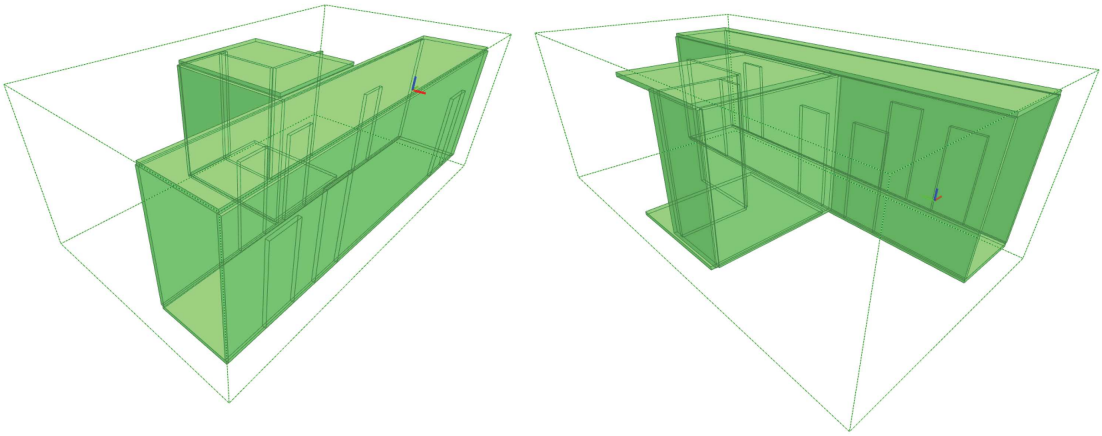


(b) *3D mesh featuring an curved PB with a curved and hollow SB, with a mesh complexity of 45 102 vertices and 15 034 faces.*

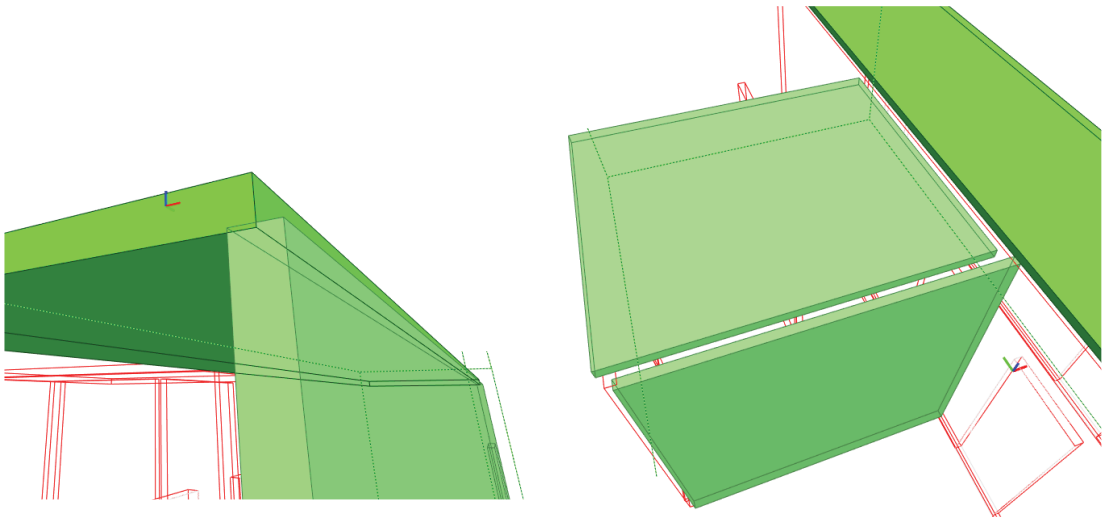
Figure 5.14: *Examples of 3D meshes of vectorized paths from detected and regularized SBs of a floorplan partition.*



(a) *The input point cloud, with segmented point clusters and associated OOBBs for each point cluster.*



(b) *The reconstructed IFC model at LOD 300.*



(c) *Misaligned regions of the IFC model elements that require further manual editing with specialist BIM software.*

Figure 5.15: *Reconstruction result of an indoor point cloud to an IFC using the Bounding Box-based reconstruction method.*

5.7 Concluding Remarks

The presented methods are suitable for reconstruction of point clouds to higher level geometric representations, which can be used for specific applications e.g., spatial deviation analysis (Chpt. 6) and *as-is* BIM model generation. In most cases the accuracy of the reconstruction methods is not accurate enough when compared to manually reconstructed models made using expert CAD or BIM software, but the outputs are suitable for approximating spatial relations and performing visualization tasks of the reconstructed objects.

The triangular geometry reconstruction methods, PSR and BPA, can be used to approximate the triangulated mesh from a point cloud. BPA is usually more suited towards reconstruction of non-organic shapes (or shapes that do not have curved surfaces), while the PSR algorithm is suited towards approximating smooth and curved surfaces. Both methods usually require further post-processing of the mesh (most notably closing any open areas that are featured on the triangulated mesh surface). For most reconstruction requirements of indoor point clouds, the use of the BPA method is preferred as it is able to preserve the hard edges often found in building elements.

The voxel geometry reconstruction method is suited towards discretized spatial approximations (e.g., deviation analysis). As such, it cannot be used to generate an actual as-is mesh representations, but can rather be used as auxiliary data for further assessment alongside an as-is BIM model. Further uses of the voxel geometry reconstruction method for spatial deviation analysis are discussed in Chpt. 6.

The approximate floorplan generation method is suited towards approximating 2D and 3D floorplans of particular indoor building areas. While the method does not produce approximations with the accuracy obtained by traditional CAD or physical drawings, it can be used to quickly approximate spatial dimensions, and create basis visualization data for enhancing communication and decision making for FM operations. Furthermore, it is better suited for evaluating regularized SBs for an airtight floorplan representation, in comparison to the α-shape algorithm (which is more accurate, but is prone to over-evaluation) [240]. Rectification methods for generated 3D floorplan meshes can further be used to make them more visually appealing and possibly increase accuracy [91].

The bounding box approximation method is suited towards generation of as-is BIM models up to LOD 300 [36]. The method is able to capture either AABBs or OOBBs of planar segmented point clusters and reconstruct these as floors, walls and ceilings. While the method can capture most of the planar geometry, further semantics can be added to each of the clusters (either manually or using semantic segmentation methods discussed in Chpt. 7), and the bounding box meshes representing each of the IFC components are usually re-aligned post-reconstruction.

Chapter 6

Spatial Deviation Analysis

One of the core functions of point cloud analysis is being able to compare their representation of the physical environment with *as-designed* or *as-built* BIM data at a given LOD. The captured point clouds represent the building, by default, as a collection of discrete surface points. In contrast, an existing as-designed BIM represents the building typically as a polygonal 3D model with attributes that refer to the semantics of building parts. Point clouds capturing the current state of the built environment can be compared against an existing BIM, and this way they are a key data source for all BIM update processes, particularly those concerning O&M procedures.

The problem of comparing spatial differences between geometric representations of *as-is* or *as-built* versus *as-designed* geometry is a common occurrence, especially when comparing point cloud and BIM geometry. This process is known as *Spatial Deviation Analysis*. The process of spatial deviation analysis may include comparing the inclusion or exclusion of certain building elements captured by point clouds – when compared to the original *as-designed* or *as-built* elements as well as the their spatial alignment corresponding to existing geometric documentation (e.g., floorplans – Fig. 6.1). The use of spatial deviation analysis therefore plays an important role for FM applications e.g., space management and refurbishment or renovation.

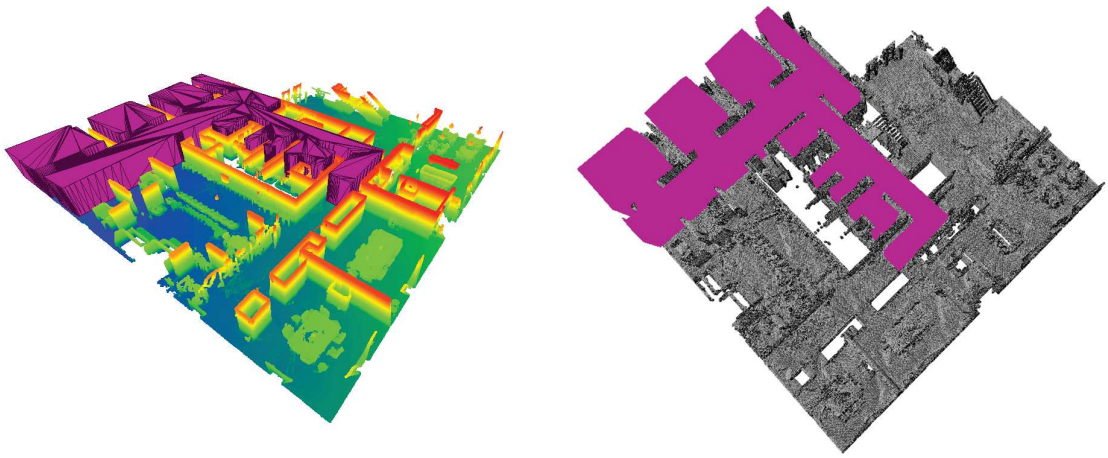


Figure 6.1: *Examples of an aligned and triangulated as-designed BIM element with as-is point cloud geometry. The next step is to compare any spatial deviations between intersecting and non-intersecting points of the as-is point cloud with the as-designed BIM.*

Selected Approaches Three approaches are presented for spatial deviation analysis, two of which are based on previously published work and results in Stojanovic *et al.* [234, 241]. The first approach, *Parametric Shape* comparison, is based on comparing a geometric primitive approximation (e.g., a plane), from as-designed or as-built BIM model elements (e.g., floors, walls and ceiling), to an as-is point cloud representation.

The second approach, *Voxelized Shape* comparison, is based on using the concept of FEA by comparing voxelized elements of as-designed or as-built BIM geometry to the as-is point cloud, and subsequently detecting the points that share the same 3D space as the voxelized geometry.

A third deviation analysis method, *Point to Mesh* comparison, is evaluated using the CloudCompare software tool. The most important aspect of these deviation analysis methods is thus the ability to visually analyze the results that highlight differences between *as-designed*, *as-built*, and *as-is* elements, and to possibly combine it together with other related spatio-temporal data and building documentation. Finally, the integration of spatial deviation analysis approaches as software components within an prototypical SOS implementation are discussed in detail.

6.1 Implementation Overview

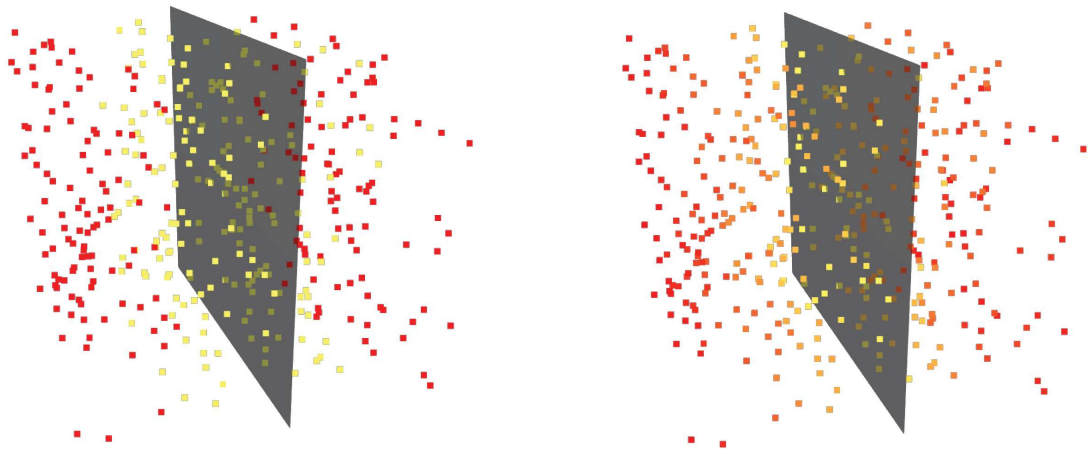
SOS Integration of Deviation Analysis Components The combined use of Web3D-based graphics frameworks allows for processing and visualization of indoor point clouds and associated semantics for spatial deviation analysis tasks, capable of running on different client hardware configurations (including mobile devices) [241]. Therefore, such processes can be implemented as SOS software components and services (Chpt. 3.5.1). Web3D frameworks such as Three.js [37] can be used to visualize important scene elements e.g., deviating points.

The visualization for the deviation analysis is implemented as part of the *Representation Processing* component, specifically the *Data Mapping* sub-component, making use of the programmable graphics pipeline. Using the default shader for point cloud materials provided by Three.js, the color and the opacity of any selected point group can be modified during run-time of the application, thus deviating points ca be highlighted and visualized.

Data Processing for Deviation Analysis The deviation analysis process requires the compared geometry to be processed in order to contain specific attributes. For as-is point clouds, this processing may include computation of normal vectors, segmentation, reconstruction and transformation operations (Chpt. 4).

For as-designed or as-built BIMs, this may include the generation of voxelized FDMs for specific geometric elements. The main processing components used for preparing BIM and point cloud data for deviation analysis are implemented as command line tools that are executed on the server (forming part of the *Deviation Analysis* component). These command line tools makes use of the PCL framework [211] for initial point cloud processing, and the Binvox tool for generating voxelized FDMs [166].

6.2 Visualization Characteristics of Deviation Analysis



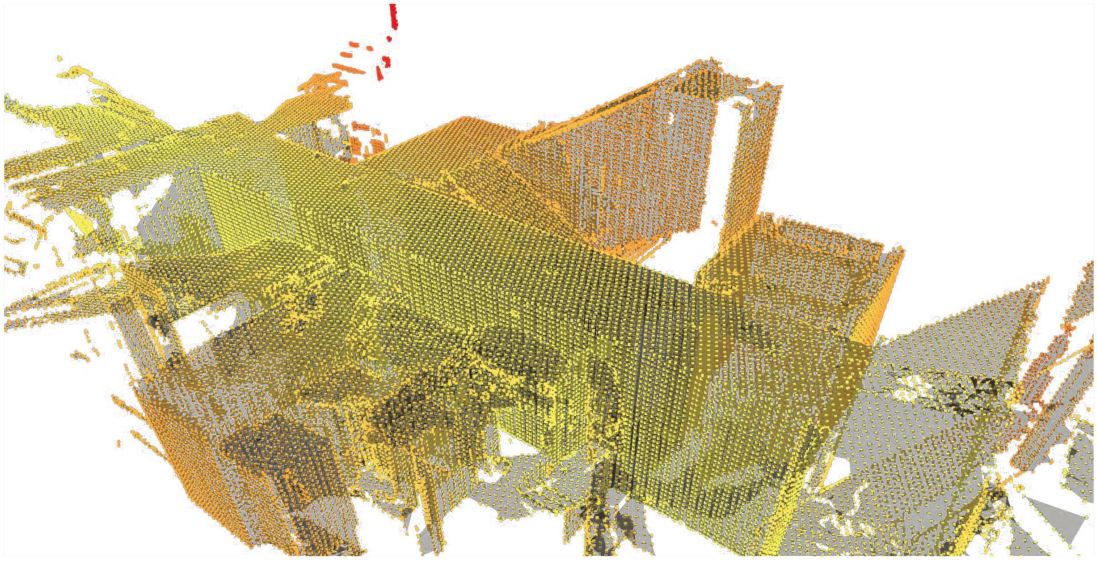
(a) *Binary distance coloring with blending.* (b) *Gradient distance coloring with blending.*

Figure 6.2: *Example of blending with the two different visualization styles for deviation analysis.*

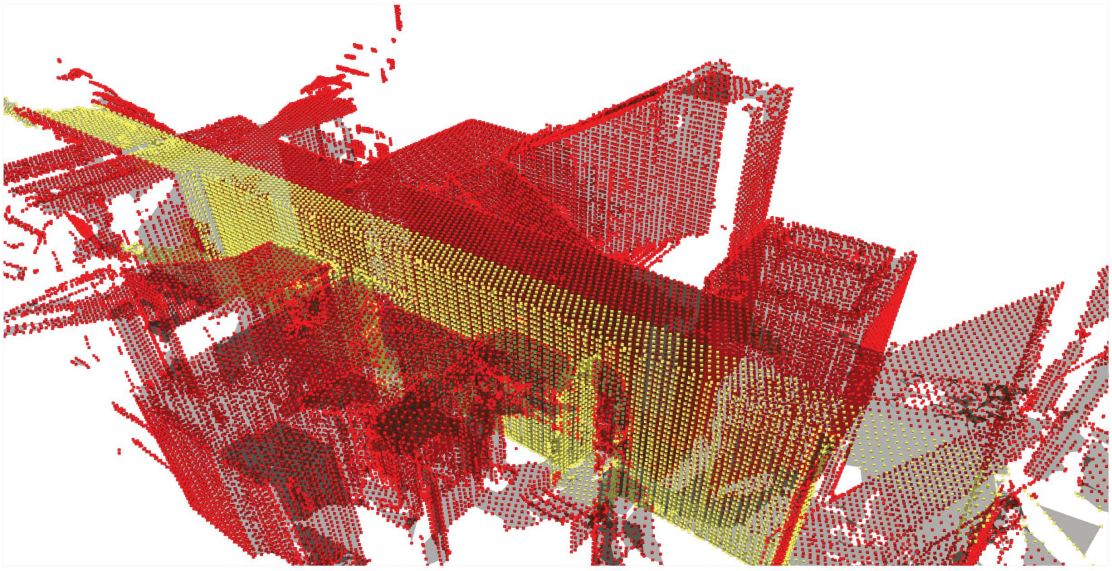
Given the computed spatial deviation values, the deviation analysis visualization can be generated. Two visualization methods used to highlight spatial deviations are presented and discussed: (1) binary distance coloring and (2) gradient distance coloring visualization styles. The binary approach uses a threshold value to determine at what distance the points are deviating. This value can be adjusted by the user. This allows the user to set an acceptable "fault tolerance" for the comparison of as-designed versus as-is built geometry types.

For example, the binary visualization style shades all points beyond a threshold level as red, and all points within the threshold value as blue. The gradient visualization method allows for continuous shading of points from those smaller than the threshold to those greater by linearly interpolating a color scale between them.

Formally, the mapping of the binary and gradient distance coloring can be defined as: the distance between the point and the compared geometry element $D_{relative}$, which is computed as $D_{relative} = \frac{D_{relative} - D_{min}}{D_{max} - D_{min}}$ where D_{min} is the minimum distance from the point to the compared element and D_{max} is the maximum. The gradient color mapping can then be defined as a linearly interpolate value C_{grad}, where $C_{grad} = D_{relative} \cdot C_{end} + (1.0 - D_{relative}) \cdot C_{start}$, and where C_{start} and C_{end} are the starting and ending RGB color values the gradient will be interpolated between. The binary color mapping operation is defined as C_{binary}, which can only have two values assigned to it, and is defined as $C_{binary} = \begin{cases} C_0, D_{relative} < T \\ C_1, D_{relative} >= T \end{cases}$, and where C_0 and C_1 are the two binary RGB color values, and T is the threshold value. Additionally, the use of blending operations can be applied to both points and the compared geometry, and provides better visualization results when dealing with occluded scene elements (Fig. 6.2 and Fig. 6.3).



(a) *Gradient distance coloring with blending between points and reconstructed mesh.*



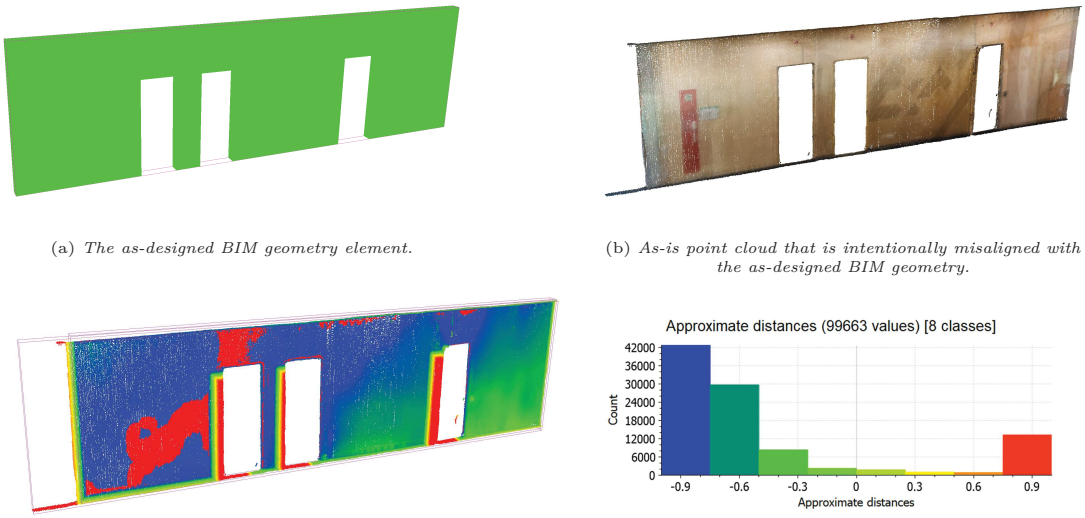
(b) *Binary distance coloring with blending between points and reconstructed mesh.*

Figure 6.3: *Further example of blending with the two different visualization styles for deviation analysis using an actual indoor point cloud and reconstructed triangulated geometry based on it.*

6.3 Deviation Analysis Methods

The two main deviation analysis methods, *Parametric Shape* and *Voxelized Shape* comparison, are presented and discussed. Additionally, the use of the *Point to Mesh* comparison method via the CloudCompare software tool, is used to generate ground truth deviation analysis results to compare with the results obtained using the *Voxelized Shape* comparison method.

6.3.1 Point to Mesh Comparison



(a) *The as-designed BIM geometry element.*

(b) *As-is point cloud that is intentionally misaligned with the as-designed BIM geometry.*

(c) *Resulting point to mesh distance evaluation, using a maximum distance threshold of 1.0 units.*

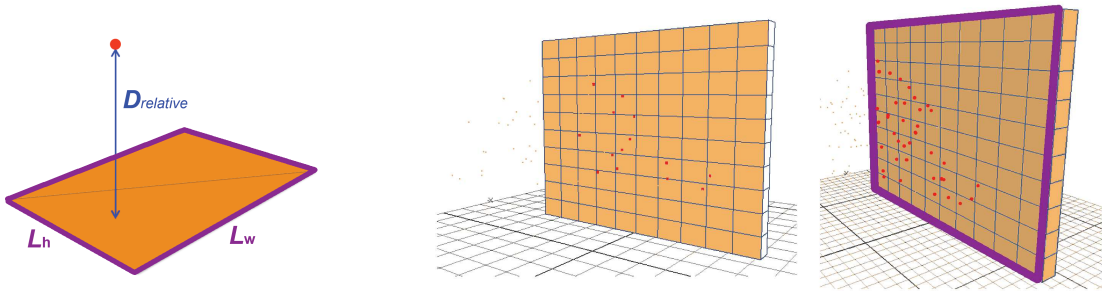
Figure 6.4: *Example of point to mesh distance evaluation using the ColoudCompare.*

The *Point to Mesh* comparison method is implemented as the standard deviation analysis method in the CloudCompare software tool (Fig. 6.4). The method measures the distance of each point to the nearest triangle of a given mesh [68]. The distance from a point to a given triangle is defined as either the orthogonal distance from the point to the triangle plane (assuming that the orthogonal projection of the point on a given plane falls inside the triangle), or the distance from the point to the nearest triangle edge. In order to select what triangle is nearest to a given set of points, an octree data structure is used to partition triangulated mesh, and the distance between points and triangles contained in each octree node is computed [89].

The distances of each point to a nearest triangle are stored as a normalized scalar value (0.0 to 1.0 if unsigned, or from -1.0 to 1.0 if taking into account both sides of a triangle) for each point, and mapped to a given color scale gradient. Additionally, the maximum deviating distance as well as the average distances distributed across the analyzed point clusters are recorded. This allows for approximate visualization and assessment of deviating points to a given triangulated mesh. In most use cases, the parametric mesh would be the extracted and triangulated geometry from a given LOD representation of a BIM element in the IFC file format.

The alignment between the as-is point cloud and the as-designed/as-built mesh can either be performed manually, or using an appropriate point registration method (Chpt. 4.3.1). The longitude and latitude coordinates, if available, can be obtained from either of the models and used to transform both to the same location in 3D space prior to deviation analysis.

6.3.2 Parametric Shape Comparison



(a) *Measurement of the distance value $D_{relative}$ from a point to the plane parametric shape.*

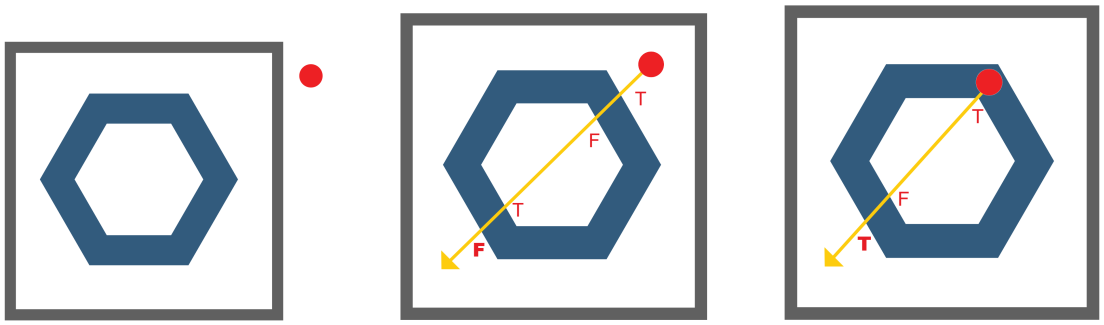
(b) *Example of how the parametric plane shape is projected in 3D space and relative to the 3D BIM geometry and points.*

Figure 6.5: *Illustration of the parametric shape comparison, every point is compared against a plane object derived from the bounding area of BIM geometry element.*

The *Parametric Shape* comparison method for deviation analysis makes use of an approximated plane that is derived from triangles of a given as-built/as-designed model, and compares how far each point is from the given plane (Fig. 6.5). This approach is the simplest and fastest approach for deviation analysis, but also the least flexible and accurate – as each of the as-designed/as-built BIM components needs to be evaluated separately, and the distance from a point to a given BIM component is measured using the center of the components approximated plane.

Additionally, as the plane has no depth, it is geometrically treated as a planar object defined as L, with a given width L_w and height L_h. As such, this approach is mostly suitable for spatial deviation analysis of planar objects (e.g., walls, floors and ceilings).

Extension for Parametric Shape Comparison



(a) *Point outside the shape bounding box.*

(b) *Point inside the bounding box, but outside the shape.*

(c) *Point inside the bounding box and inside the shape.*

Figure 6.6: *Illustration of the three cases when testing for a point outside the shape bounding box, inside the bounding box and inside the shape. Each time a ray intersects with a shape inside the bounding box, the value of the Boolean variable used for intersection testing is switched between true and false. If the value is true, then it can be assumed that the point is inside the shape.*

An extension of this method is to compute the actual bounding boxes (usually OOBBs) of each BIM element, and compare which points intersect the bounding box. However, since a bounding box can enclose both concave and convex manifold shapes that represent various BIM elements, it is not sufficient to just check if a point is inside the bounding box, but if it is inside the shape enclosed by the bounding box (Fig. 6.6).

A function for checking if the point is inside or outside of a polygon can be implemented to achieve this task (as described by Franklin [82]). The idea of such intersection testing is based on the concept of a *Jordan Curve* [105] i.e., the boundary polygon is treated as a simple closed shape that separates the inside region of the curve from the outside region. For such a test, a ray that is projected horizontally from a given point can be cast, and the number of intersections with the edges of the boundary polygon are detected as a switchable Boolean statement. Additionally, methods for testing ray to triangle or ray to volume intersections can also be used [258].

6.3.3 Voxelized Shape Comparison

The *Voxelized Shape* comparison approach is used to evaluate and visualize how close a cluster of points of a given *as-is* point cloud is to the overlapping voxel element of a voxlized *as-designed* BIM mesh element in the same 3D space. The deviation threshold value is used to determine beyond what threshold (measured as distance in Euclidean space), a 3D point is considered to be deviating. This value can be adjusted by the user or based on the required use-case specific parameters. The deviation threshold value allows for setting an acceptable fault tolerance when comparing different geometric and primitive-type representations.

Generally, the generated voxel mesh does not have any void spaces inside it (i.e., every voxel is connected to a neighbouring one). Since the captured point cloud representation is usually projected along a given plane (point clouds are not projected in a volume), the use of a voxelized mesh (i.e., a FDM), without void spaces between the voxels, allows for comparison of points that might otherwise be missed if they were located inside these voids (Fig. 6.7).

If the point is contained in an voxel, it can be marked as *non-deviating*, otherwise the point is marked as *deviating* (Fig. 6.8). The first test is to see what points from the complete point cloud are inside the bounding box of a voxel, which are then copied to a temporary array – along with an additional integer key value to indicate which specific points in the point cloud array are copied. This array is then spliced using these key values in order to obtain a new temporary array containing the deviating points. These deviating points are then added as a new point cluster to the scene and marked visually as being deviating (Fig. 6.8(a)).

Apart from the binary deviation, it also needs to be taken into account if the deviation is present as a surface damage or erosion element (e.g., damaged, missing or eroded elements). This approximation is referred to as the *point sparsity*. In order to assess point sparsity, an average threshold value based on the number of points contained for each voxel element must first be determined.

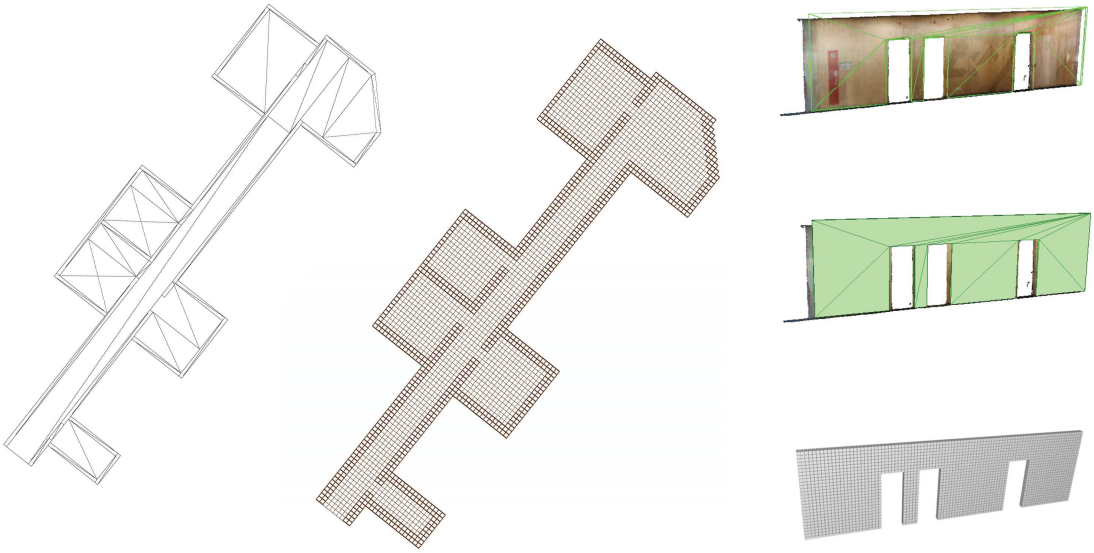


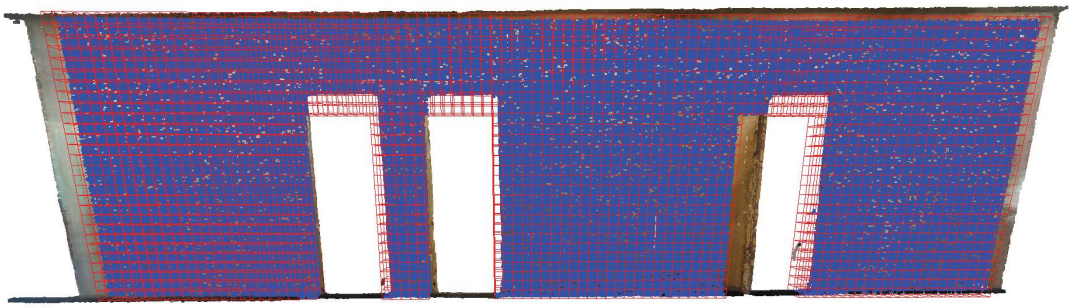
Figure 6.7: *Example of voxelized as-designed BIM geometry.*

The point sparsity computation then checks to see if each voxel contains a number of points equal to or above this average threshold value. If the case is that the voxel contains a number of points below the average threshold, it can be assumed that the missing points present a deviation (Fig. 6.8(b)). The *Voxlized Mesh* comparison method can therefore capture deviations of points when comparing a number of different voxelized forms of BIM elements, making it more flexible, albeit also more computationally expensive, than the *Parametric Shape* comparison method. Due to this flexibility, it is the default method for deviation analysis in the prototypical SOS implementation.

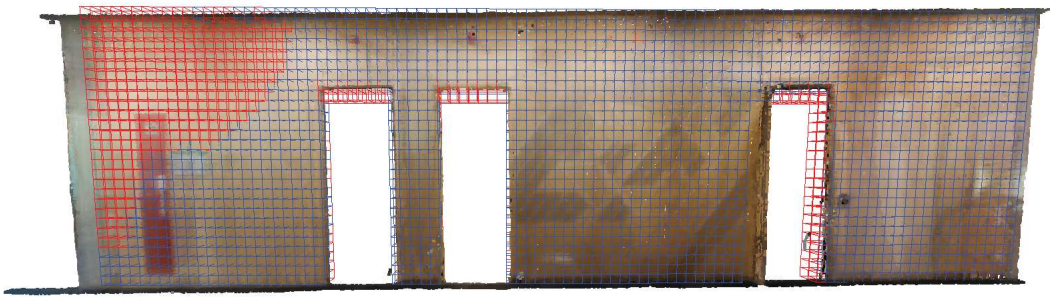
6.4 Case Study

The *Voxelized Shape* comparison method is part of the *Deviation Analysis* software component of the prototypical SOS implementation (Chpt. 3.5.1). The performance and the accuracy of this method is evaluated in this case study, using point clouds and as-designed LOD 300 BIM geometry elements representing typical office environments. The point cloud used for the comparison originally contained 2 506 858 points, but was sub-sampled to 501 372 points to decrease processing time. The voxelized BIM geometry mesh used for comparison contains 14 604 voxel elements (approximate resolution of 128^3).

For generating the final visualization and performance evaluation results, a commodity laptop with an Intel i5 1.8 GHz Central Processing Unit (CPU), 8 GB RAM, and NVidia GeForce MX150 GPU with 2 GB video memory was used. These results are based on the case studies previously published in Stojanovic *et al.* [232].



(a) *Result of the spatial deviation analysis, where all non-deviating points are colored blue.*



(b) *Result of the point sparsity deviation analysis variant, where voxels with too few points are marked red.*

Figure 6.8: *Example of spatial deviation analysis using the Voxelized Shape comparison method.*

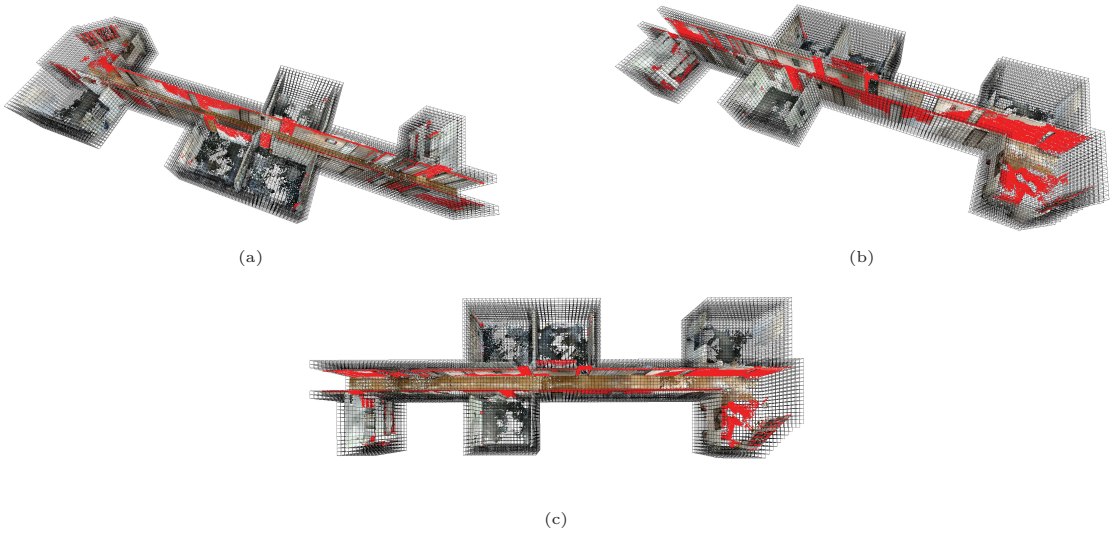
Both binary and point sparsity visualisation methods are used to highlight spatial differences between the compared *as-is* and *as-designed* geometry. The voxelized BIM geometry was compared against the corresponding point cloud for obvious deviations (Fig. 6.9), where deviating points are highlighted in red. Point sparsity deviation analysis is also performed - highlighting voxels with eroded or missing deviations in red, while *healthy* voxels are highlights in blue (Fig. 6.10). Furthermore, for empirical evaluation, the CloudCompare software tool was used to generate the ground truth results, which are then compared visually against the obtained results (Fig. 6.11).

The CloudCompare software tool can compare point cloud versus point cloud as well as point cloud versus mesh spatial deviations – generating both visual and numerical deviation analysis results using an octree-based approach with a linearly interpolated color map to visually highlight any spatial deviations.

A threshold level of 100.0 units was used in order to mark deviating points in the CloudCompare software tool. For both comparison methods, the BIM geometry and the point cloud were aligned manually. The ground truth results show that the majority of the points which are aligned with the as-designed BIM mesh are marked as deviating, as they exceed the threshold value of 100.0 units. It can be observed that the majority of deviating points correspond spatially to the deviating points that were computed using the Voxelized Mesh comparison method (Fig. 6.9).

Finally, the computational performance using Voxelized Shape Comparison method

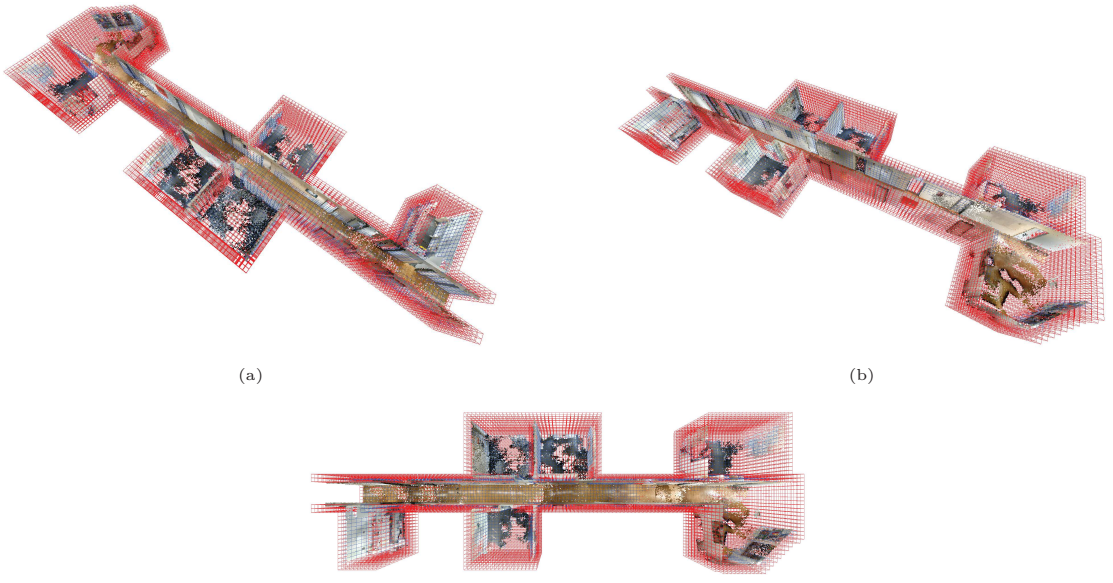
to obtain spatial deviation analysis results is evaluated. The average computation taken (in milliseconds) is measured, for the two different spatial deviation analysis methods (binary and point sparsity deviation analysis). The average time taken to perform the binary deviation analysis was 645 368 milliseconds, while the average time taken to perform the point sparsity deviation analysis was 452 215 milliseconds.



(a) (b)

(c)

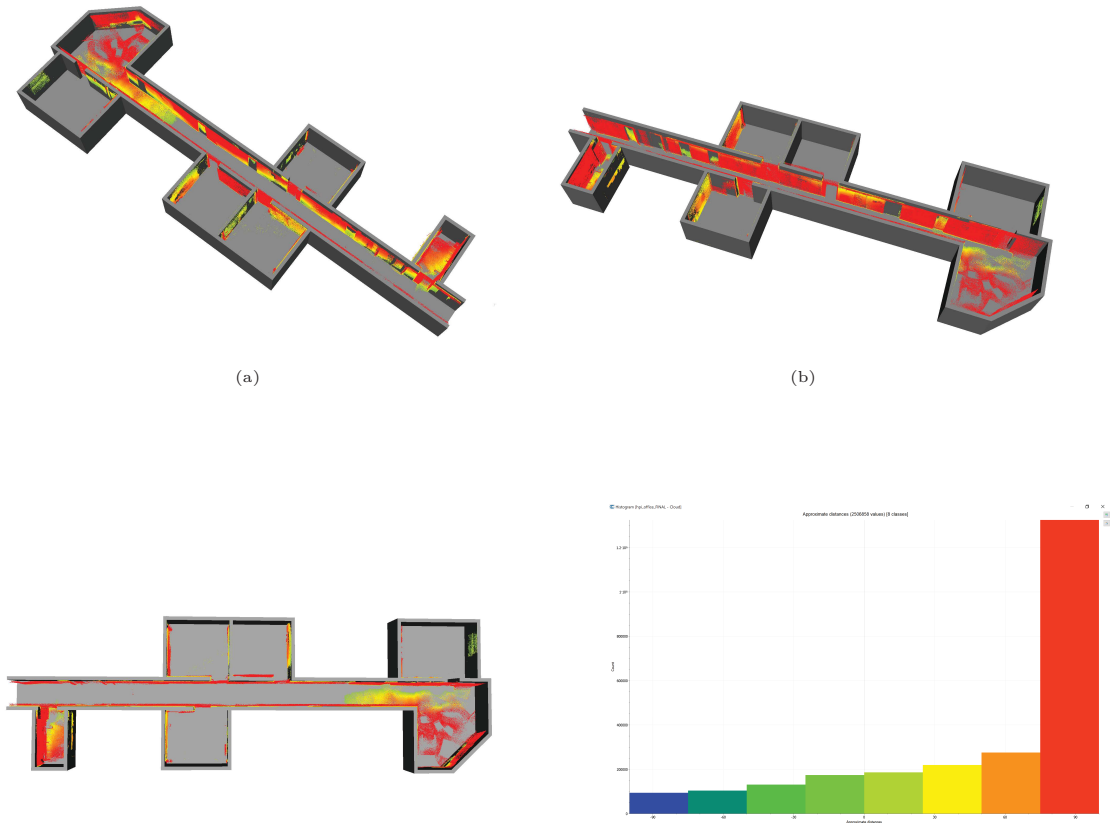
Figure 6.9: *Visualization results for the binary deviation analysis, with detected deviating points colored in red, using the Voxelized Shape comparison method.*



(a) (b)

(c)

Figure 6.10: *Visualization results for point sparsity deviation analysis, using the Voxelized Shape comparison method.*



(a) (b)

(c) (d)

Figure 6.11: *Ground truth deviation analysis results generated using the CloudCompare software tool. The points that exceed the deviation threshold are colored in red.*

6.5 Concluding Remarks

The key advantage of using point clouds for deviation analysis is that they directly provide added values for stakeholder decision making tasks, through visualization and analysis outputs. This is especially beneficial for O&M procedures where changes in interior environments need to be recorded and compared.

The methods presented for spatial deviation analysis provide suitable approximations for measuring the spatial differences between as-is and as-designed/as-built representations. However, they cannot be used to measure spatial deviation distances for AECOO or CQA applications where accuracy is required (in contrast to using precise manual measurements alongside accurate CAD models). As such, the spatial deviation analysis methods are suitable fro providing initial assessment overviews of built environment changes.

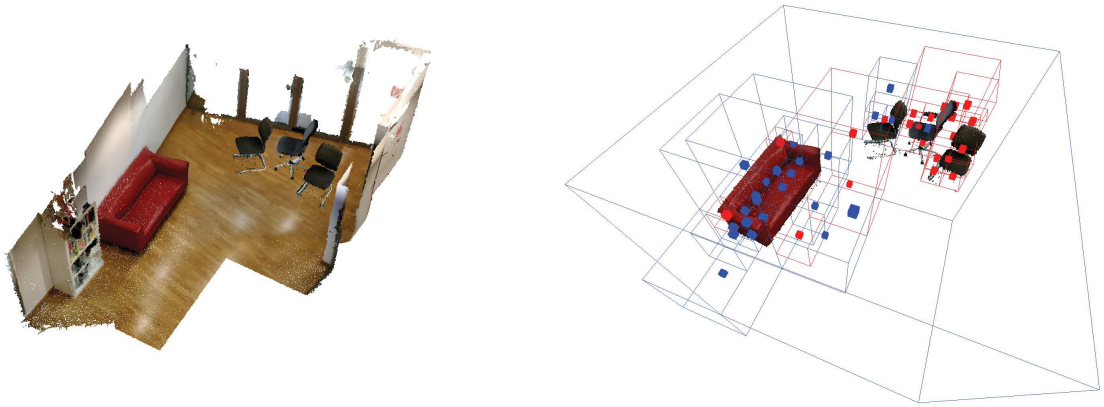
The integration of the presented spatial deviation methods as SOS software components (*Parametric Shape* and *Voxelized Shape* comparison) also removes the dependency on using third party monolithic software tools, though it is still important to compare the deviation analysis results against the the ground truth approximations obtained from such software tools (Fig. 6.11).

The use of discritized parametric surfaces, particularly voxelized geometry i.e., FDMs, is key to enabling the computation of distances between points and corresponding as-designed/as-built BIM elements – such elements are often represented as triangulated or simplified bounding-box or plane geometry. Furthermore, the use of Web3D-based technologies within the *Representation Processing* components of the SOS implementation enables the visualization of the spatial deviation analysis results on both thin, medium and thick client devices.

Finally, the aspect of performance needs to be considered. While the presented approaches are suitable for quick approximations, if higher precision is required then the corresponding FDM needs to be of higher granularity, thus increasing the processing time (usually the deviation analysis performed using the *Voxelized Shape* comparison method corresponds to $O(nlog(n))$ computation time). This can be mediated by finding the right balance between the compared model geometry complexity and the required deviation analysis accuracy – either using a heuristic approach or by evaluation by stakeholders who can adjust the required parameters needed for the analysis.

Chapter 7

Semantic Enrichment of Indoor Point Clouds



(a) *The input RGB point cloud representing a common office area.*

(b) *The segmented and classified output point cloud, where red nodes indicate point clusters belonging to the chair class, and blue to the sofa class.*

Figure 7.1: *Example of octree-based multiview classification.*

A 3D point cloud consists of what can be defined as *non-interpreted data* – data that is open to visual interpretation but does not have any semantics associated with it. While point clouds can be used by themselves to represent the current state of the physical environment for practical needs (e.g., assessment of space usage in a room), for any further representations and assessment the point cloud needs to be processed in order to generate useful semantics for it (Fig. 7.1).

This chapter describes the approaches for semantic enrichment of indoor point clouds, focusing primarily on unsupervised and supervised learning methods for automated classification of point clusters as well as manual user-based semantic enrichment to a lesser extent. The semantic enrichment methods presented in this chapter have been previously published in Stojanovic *et al.* [238, 236, 241, 239].

7.1 Semantic Enrichment

Since raw point clouds do not provide any semantics by default, there is a need to add semantics to them in order to make them more related to an existing domain in a given assessment context (i.e., point clouds used to represent indoor environments would need to differentiate between office furniture and structural elements such as door frames and walls). This may include adding explicit meaning to an object represented within a point cloud cluster, or by adding additional information into the point cloud (e.g., textual annotation), which can be used to derive understanding when assessing it further for decision making. The term *semantic enrichment* is based on the definition given by Sacks *et al.* [212].

Such types of semantics may include human-readable labels (e.g., "chair", "table", "office table", "book shelf", etc.), or it may include specific attributes related to a higher level semantic representation or an ontology (e.g., IFC schema specific attribute names (Chpt. 5.5)). The main reasons for adding semantics to point clouds are: (1) It enables them to be better interpreted for decision making tasks within a specific AECOO domain (e.g., to complement visual inspection), and (2) Enables reconstruction to higher-level representations that require explicit semantics to be associated with geometry (e.g., BIMs represented using the IFC format).

There are three main ways in order to add semantics to a given point cloud: (1) Manual Semantics, (2) Unsupervised Learning Methods, and (3) Supervised Deep Learning Methods. All three methods usually require point clouds that have been processed (e.g., registered with a given floorplan), and include additionally computed attributes that can be used as features (e.g., color and normal vectors).

7.2 Manual Semantics

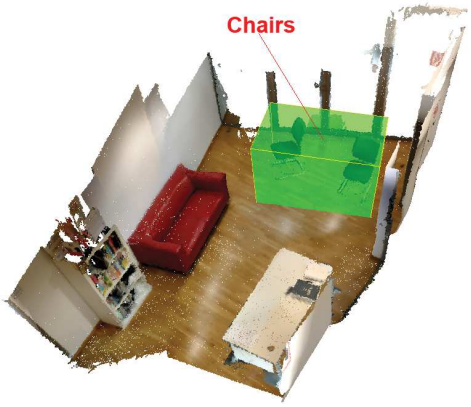
The process of manual annotation of point clouds introduces semantics as a product of the user explicitly adding information for a selected point cloud cluster (Fig. 7.2). Common methods for manual annotation of point clouds include manual selection of regions of interest in 3D space, with the user then adding specific textual or numerical information for the selected clusters via a user input interface (e.g., a GUI dialog). Such methods are based on common computer graphics principals of user interaction within a 3D scene, and the annotations can be displayed using common visualization idioms [243].

Typically, the user input for semantic enrichment tasks is part of the user interface implementation of the client-side processing of the SOS. In such a case, users are able to adjust each of the input parameters, options and information boxes for semantic enrichment tasks, send this to the server for processing, and interactively view and inspect the generated results sent back by the server.

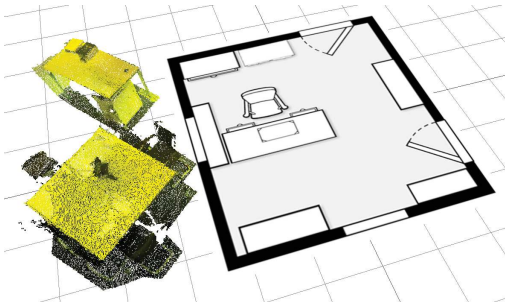
The use of a client-side GUI enables user adjustment of processing parameters, thus making the process of semantic enrichment via user input adaptive to various indoor point cloud representations. This can also encourage users to experiment and learn optimal parameter configurations are for their specific needs.

(a) *Calculation of the square area between four selected points.*

(b) *Example of measurement of distances between two points.*



(c) *Chair object selection using a bounding box volume, and user inputted textual annotation.*



(d) *Alignment of a selected point cluster with corresponding region in a 2D floorplan.*

Figure 7.2: *Example user inputted annotation and distance/area measurements of a selected region of a point cloud.*

However, one key requirement for user inputted semantic enrichment is that it is assumed the user has adequate domain expertise in order to add meaningful, factual and correct semantics to a given point cloud representation.

The two main steps for user inputted annotation are: (1) Point cluster selection and (2) Annotation input. For selection of point clusters, a number of approaches can be adopted. The most common way is to use a *picking* algorithm in order transform a 2D vector, based on the mouse cursor screen coordinates controlled by the user, into a 3D projection and detect if any object is behind the mouse cursor. The selection of the object behind the mouse cursor is performed using a ray-based intersection test, where a ray projected from the position of the mouse cursor is tested against the geometry type of the object it is attempting to pick. This may include intersection tests between OOBBs and AABBs, points, triangles or other geometric primitives [286]. If the point cloud object is already segmented, the point clusters may be individually selected as well by using their OOBB for the object picking intersection test.

Apart from object picking, the other method is based on volume selection (Fig. 7.2(c)). With this method, a 3D volume based on a given primitive (e.g., a box, cylinder, sphere, etc.), is drawn by the user by specifying its width, length and height. Then, the user is able to move this 3D volume shape around the 3D scene and select a location. At this location, any 3D points that are within the volume are selected. Thus the user is able to select a desired region of the point cloud for manual segmentation and further annotation.

There are are also different methods for user inputted textual, numerical and pictorial annotation of 3D point clouds. Firstly, users can measure distances and areas between two selected points, which allows them to obtain measurements of spatial regions in 3D space (Fig. 7.2(b)). Secondly, the users are able to add custom text annotations to specific point cloud clusters, so that any important information (e.g., measurements), can be viewed by other stakeholders who may be inspecting the point cloud (Fig. 7.2(a)). Thirdly, users can also load 2D images (e.g., floorplans), and associate these with a point cloud – so that when a user clicks on a point cloud, additional pictorial information is presented along with the textual and numerical information (Fig. 7.2(d)).

7.3 Use of Unsupervised Learning Methods

ML approaches are deeply rooted in statistics, and as such they usually attempt to distinguish types of data based on specific features, using an *unsupervised learning* approach i.e., where knowledge of what the data represents is *not* known prior to any analysis). A common use of unsupervised ML approaches is for clustering of data (i.e., grouping of data based on the metric of similarity of one of the features of data e.g., color or position in 3D space).

Widely used clustering methods that are suitable for cluster analysis of point clouds include k-means and DBSCAN clustering (Chpt. 4.3.6). Considering the structure of point clouds (Chpt. 4.1), their features can be used to derive relations between similar

regions made up of other points e.g., for applications such as segmentation of rooms using k-Means clustering, etc., (Fig. 7.3).

Typical features of points that can be used for clustering include their spatial distribution, color and orientation (i.e., normal vectors). Such features are used to represent physical surface properties that point clouds attempt to capture e.g., walls of a building with a similar color or surface curvature measured using pre-computed normal vectors.

Additionally, segmentation of point clusters based on Region Growing is discussed in Chpt. 4.3.5.

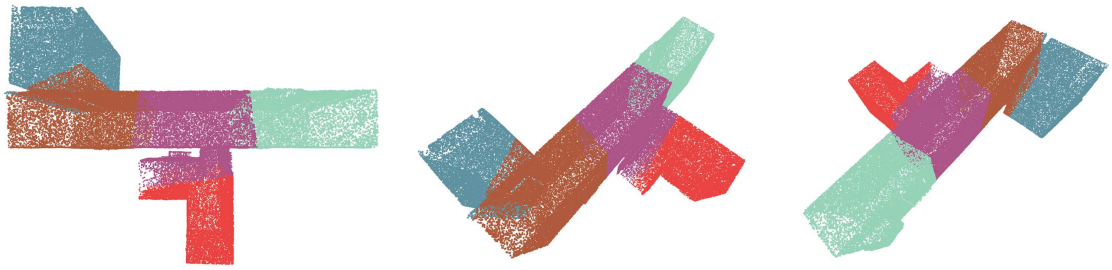


Figure 7.3: *Example of k-Means clustering-based segmentation of an indoor point cloud.*

7.4 Use of Supervised Deep-Learning Methods

Deep-learning is a subset of ML that makes use of *supervised learning methods* i.e., where the knowledge of what the data represents and how it is supposed to be classified *is known* prior to any analysis. Deep-learning methods thus attempt to predict what the data is supposed to represent, based on a model that is trained to detect the same types of existing data.

The use of supervised deep-learning in the presented context relies on using a CNN to predict the probability of input data belonging to one or more classes. In turn, the process of training such a CNN requires the use of example training data and its specific features, in order to train a given CNN model to make correct predictions. The correctness of these predictions depends on increasing the prediction accuracy during training using iterative optimization methods. Applications of deep-learning for point clouds are typically concerned with *semantic segmentation* and *classification*.

Semantic segmentation and classification enables the assignment of points to classes i.e., labels and categories (e.g., "wall", "chair", etc.). Furthermore, semantically enriched point clouds can be used as base-data for BIMs and DT representation [242]. Babacan *et al.* [13] present an approach of semantically enriching indoor point clouds using a combined approach of classification with a 3D CNN and further segmentation of planar point clusters using the RANSAC algorithm. Other recent approaches for semantic segmentation of indoor scenes make combined use of both 2D and 3D CNNs for clustering and semantic labelling of indoor point clouds [48, 271].

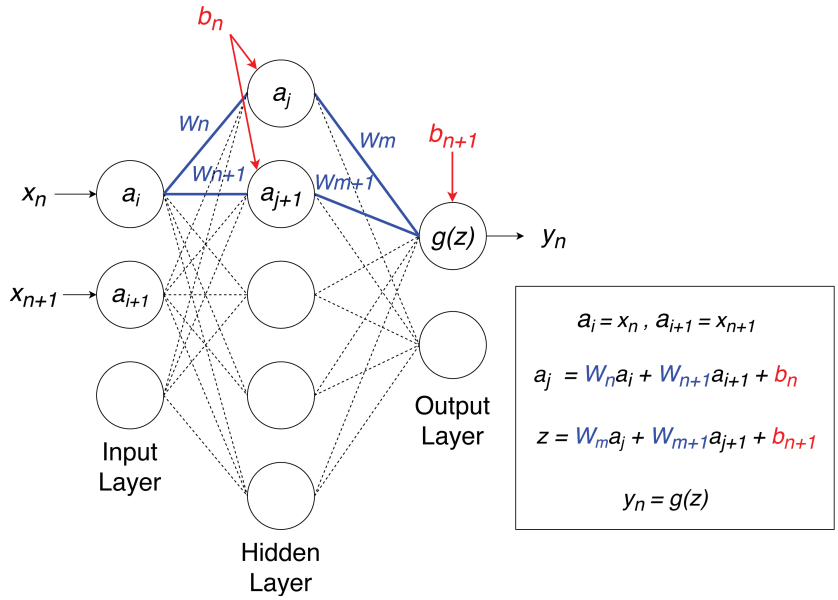


Figure 7.4: *An illustration of a typical neural network with one input layer, one hidden layer, and one output layer. The predicted output y_n is based on the output of the function $g(z)$ that uses linear combination of the weights and biases of the previous input nodes from the hidden layer nodes a_j to a_{j+1}. The hidden layer nodes a_j to a_{j+1} in turn receive their inputs from the linear combination making use of the input layer nodes a_i to a_{i+1} as well as their associated weights and biases.*

Neural Networks Typical *feed-forward* neural networks are based on the model of biological signal processing, where signals are received by neurons, which are usually modelled as *layers* based on mathematical functions. The input signals are processed and sent forward either to another layer or as an output signal [4]. The output signal is then used to create a prediction value. The neurons process the input signal based on weights, biases and activation functions that attempt to evaluate and categorize the input signal. The values used by the biases and weights for each neural layer are adjusted during the training of the neural network.

A *cost function* is used to approximate how well the neural network is fitting the classification of input data with its current parameters. The cost function compares the value of the predicted classification to what the actual classification is supposed to be, and assigns an accuracy score for the current configuration of the neural network.

The objective is to minimize the value of the cost function, thus increasing the predicted accuracy of the current neural network model. The continuous adjustment of neural network parameters during training, in order to minimize the value of the cost function, is achieved using *backpropagation*.

Backpropagation iteratively adjusts the values of the weights and biases for each neural layer in the network. These values are then used to test the sensitivity of the cost function in order to reduce the *error value*. Backpropagation is enabled by differentiating

the activation functions of each of the neural layers, from the last output to the first one, with respect to the current cost function value. This way the sensitivity of the cost function to the weights and bias parameters can be approximated.

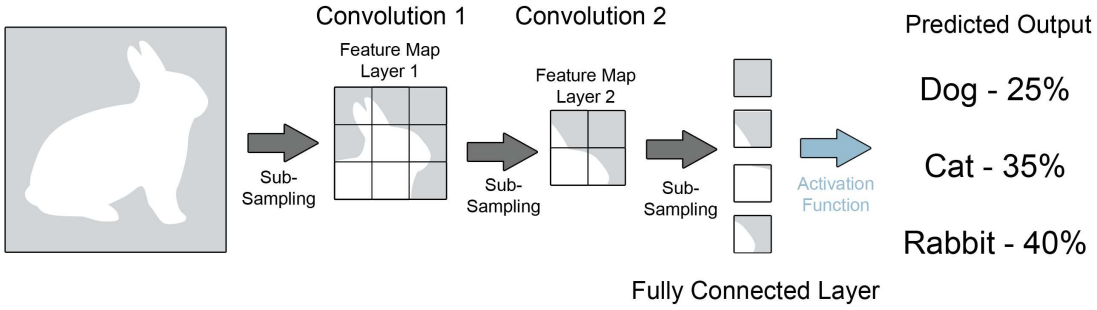
After a number of iterations, the neural network should converge to optimum values of the biases and weights (where the cost function is at the minimum value). The ultimate goal of the neural network is to bring the value of the outputted label classification as close as possible to the value of the data label that the *ground truth* is supposed to represent. This ground truth data, used during training, is known as *training data*. The training data for each class contains a number of different examples of what the entities belonging to it are supposed be i.e., if the neural network is used to classify pictures of cars, then the training data will include various types of cars that can be classified based on specific features (e.g., the color of the car).

Usually a neural network is used for non-binary classification, where there can be more than two types of classes that the data can belong to. In such a case, the input data is usually vector data containing *features* that represent the key distinguishing attributes of the data the neural network is trying to classify. The neural network attempts to assign the inputted data, based on the classification of its features, as a probability measure of belonging to each of the corresponding classes. In the presented context, such classes may include different kinds of office furniture or other indoor built environment features captured by point clouds.

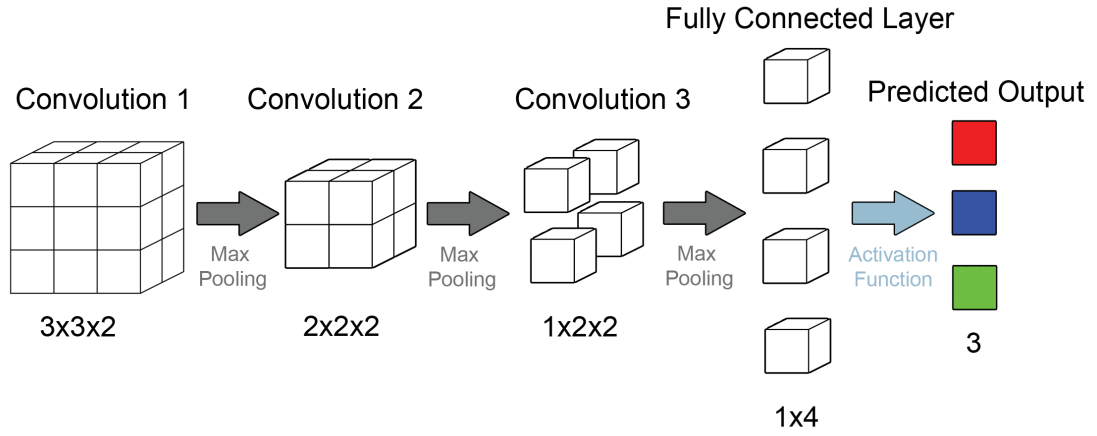
In case of point clouds, the features used for training and classification may include position, normals, color and/or any other semantics, or in some cases this may be data in different dimensions (e.g., 2D images representing ground truth data). These features are usually mapped to a vector, which is then fed through the neural network, and the output is a $1 \times n$ vector, where the size of n is the number of classes, and the n_n component of the vector represents the probability value of the input data belonging to a given class. The term *deep learning* can refer to the use of neural networks that contain more then one neural layer, known as *hidden layers* (Fig. 7.4). The increased number of hidden layers has the potential to increase the accuracy of a neural network for multi-class detection, but requires longer training times.

Convolutional Neural Networks CNNs are a special kind of deep-learning neural networks, designed to work with multidimensional structured data [95], by performing *convolutions* on the data between at least one of the hidden layers (Fig. 7.5).

The convolutions allow the mapping of features in the form of reduced dimensionality (known as *pooling*), onto a *feature map*. A convolution consists of applying a kernel to a 2D or 3D data set, represented either as a $n \times n$ or $n \times n \times n$ input layer (n being the dimension size), where the results of the convolution are used to generate a new output layer. A convolution makes use of the concept of a *sliding window*, where a kernel moves over the defined area of the image or volume, and maps the weighted sum as a reduced feature space output layer. This output layer is subsequently processed by additional convolutional neural layers, each of which further extract specific features with the use of various kernels and generate subsequent feature maps.



(a) *A simple 2D CNN, with two convolution and max pooling layers, and one fully connected layer for classifying an image into three different classes.*



(b) *A simple 3D CNN, with three convolutional and max pooling layers, one fully connected layer, and classification value mapping to three different classes.*

Figure 7.5: *Example of conceptual 2D and 3D CNN architectures to illustrate how CNNs perform classification on 2D and 3D data.*

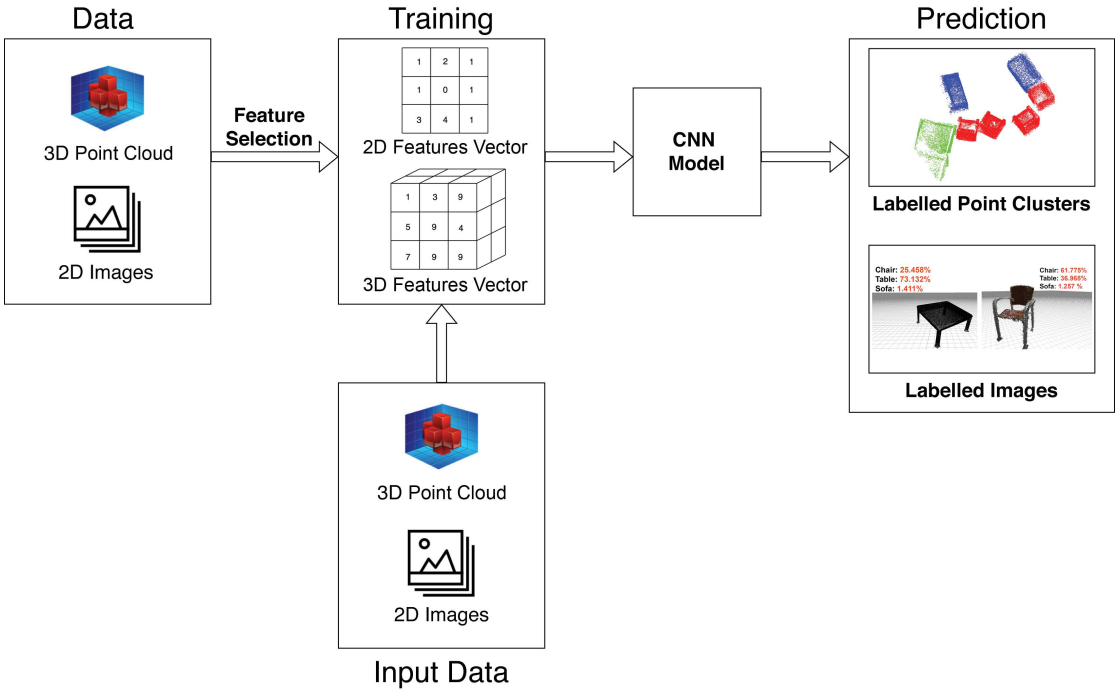


Figure 7.6: *Illustration of the process of feature selection from training data, training the CNN, data input and generation of predicted results for 3D point clusters and 2D images.*

A *fully connected layer* is used in the final stage to compute the output values, which are transformed via an activation function into a probability value, for each of the classes that the CNN is trained to detect.

Segmented regions (point clusters) of point clouds for AECOO applications can be classified using such CNNs [113, 111]. Training data used to train a CNN for classification of indoor point clouds can either be 3D point cloud clusters, geometric approximations of point regions (e.g., voxels), or 2D images of specific 3D objects as well as real-life photographs of their counterparts.

The two presented approaches for supervised deep-learning rely on either 2D image data of point clouds, known as *multiview classification*, and the classification of actual point cloud data, known as *object-based classification*. Fig. 7.6 illustrates the high-level process of classification of 3D point cloud and 2D image data for semantic enrichment of point clouds.

7.5 Multiview-based Classification

The use of 3D CNNs such as PointNet++ impose a considerable amount of overhead in terms of hardware resources and computation times. Additionally, access to existing point cloud training data is required, which can be difficult to obtain. The development and release of Google's Inception V3 CNN model and TensorFlow API allows for more

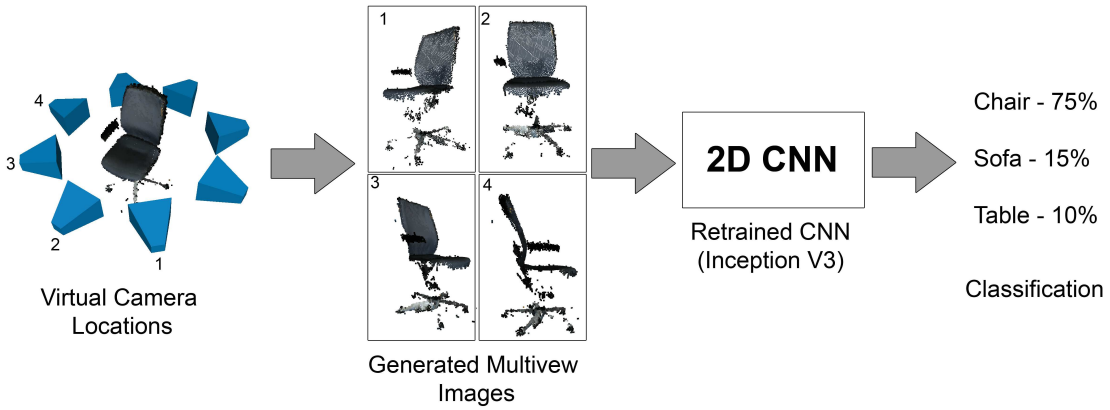


Figure 7.7: *Illustration of the multiview classification process. Captured multiview images of point clusters are classified using a retrained version of the Inception V3 CNN, after which the classified image results are associated with the corresponding point cluster.*

practical implementation and application of deep-learning-based methods for classification of 2D image data [1, 248].

Therefore, an alternative to using 3D CNNs for classification of point clouds is using 2D CNNs instead, which classify images depicting multiple views of point clusters and associating the classification results of such views back with corresponding point clusters (Fig. 7.7). Such an approach is called *multiview classification*, and focuses on classification of 2D raster images of 3D point clusters [245]. Subsequent evaluation of this approach has shown that it can, in certain cases, provide just as accurate or better results as using a 3D CNN – but without the data bandwidth requirements needed for parsing and processing large point cloud datasets [246]. The main requirement is then to classify images of indoor scenes featuring multiple objects (e.g., office furniture), and link the classification result back to the corresponding point clusters, using a multiview classification approach.

This classification result obtained from classified images can be applied to the corresponding point cluster of the partitioned point cloud, and allow each classified cluster to have an associated semantic. The use of multiview classification generally requires the following steps:

1. Discretize the 3D scene into spatial partitions.
2. Generate appropriate images for each partition.
3. Classify the generated images.
4. Associate each partition with the classification result.

The generated multiview images can also be classified in a service-oriented manner, and the classification results can be streamed back and associated with each point cluster of the indoor point cloud [241]. This allows for automated semantic enrichment of indoor

point cloud data, which in turn enables the point clouds to be used as base data for as-is BIM or DT representations. Two main approaches for multiview classification are presented and described. These include the *octree-based* and *viewpoint entropy-based* multiview classification approaches. Both approaches rely on the generation of images of spatially partitioned point clusters, and have been tested for classification of furniture items commonly found in indoor point clouds.

7.5.1 Octree-based Multiview Classification

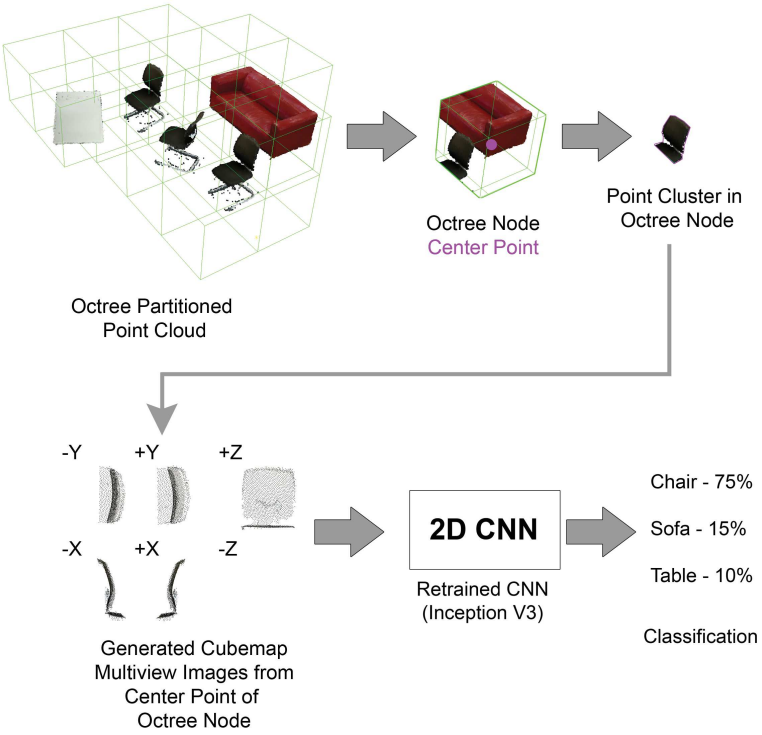


Figure 7.8: *Illustration of the octree-based multiview classification process. Generated cubemap images, captured from the center of an octree node, are sent to a 2D CNN for classification, after which the results are associated with the corresponding point cluster. Not all cubemap faces capture enough information, in which case mostly blank cubemap faces are disregarded prior to classification.*

The octree-based multiview classification algorithm captures multiview projections of point clusters partitioned within octree nodes. These multiview images are then classified, and the classification result is associated back with the corresponding node (Fig. 7.8). While 3D and 2D CNNs can be used successfully to classify single objects, including multiple objects for classification requires partitioning of the 3D scene in order to treat each partition as a unique point cluster belonging to a given object type.

The use of an octree for spatial partitioning provides a reasonable performance in terms of node traversal in comparison to other solutions (e.g., *kd*-trees), and can be used

to partition a point cloud of varying resolutions (e.g., from very dense to very sparse point clouds).

The octree partitioning scheme has to correspond to the density of points in the point cloud e.g., a spare point cloud that is partitioned using a dense octree scheme will generate too many octree nodes containing too few points to make it useful for multiview classification. In most cases, and average of 63 octree nodes are used for a point cloud with an average resolution of 330 000 points for $20m^2$. The use of an octree partitioning is best suited for capturing the macro details of the scene at this resolution.

In case of using cubemap images to train a 2D CNN, the chosen octree resolution has to be similar to the resolution used when generating the training image data for the CNN. If a very sparse octree resolution is used, and the CNN was trained on images containing finer point elements obtained from denser octree nodes, then it can be assumed that the classification will not provide the most accurate results. Therefore, a target for each scene is to have an average of 45 valid octree nodes, from which the cubemap images for classification can be generated. It is also assumed that the point cloud scene is already segmented, so that walls, ceilings and floors are removed (along with any other non-classifiable segments), and not featured in any of the generated multiview images.

Octree-based Multiview Generation Algorithm The formal mathematical definition for the generation of multiviews using octree-based spatial partitioning is described. The octree data structure, defined as O, partitions the point cloud defined as P, into a number of nodes defined as $O_{i,j,k}$. Each of these nodes contain points representing a cluster, defined as P_c, which form part of the whole point cloud representation P. At each node center $O_{i,j,k}$, a center point is defined as O_c. At this center point, a virtual perspective defined as C is placed (described further in Sec. 7.5.2). This process is described in Alg. 3.

Algorithm 3 Octree-based Multiview Generation

Require: O, P
  $P \rightarrow O$ {Partition point cloud using octree}
  for $x = 0$ **to** $length(O_i)$ **do**
    for $y = 0$ **to** $length(O_j)$ **do**
      for $z = 0$ **to** $length(O_k)$ **do**
        $O_c \rightarrow O_{x,y,z}$ {Set node center point to current node}
        Capture Equirectangular Projection of P_c at O_c
      end for
    end for
  end for

The following describe the generation of cubemap images from *equirectangular projected images* [231], based on the procedures described by Reed [201] and Greene [96]. The camera object O_c captures an image using equirectangular projection, from the center point O_c in the current octree node $O_{n_{ijk}}$ (all other point clusters in each of the nodes are set to invisible).

This image, defined as $Im := \{Im_{width,height} \in \mathbb{R}^2, Im_{x,y} \in \mathbb{R}^2, Im_{u,v} \in [0,1]^2, RGB \in [0,1]^3\}$, is then used to generate six different cubemap images that are used for classification.

The definition of each cubemap is defined as $Cm_{face} := \{Cm_{width,height} \in \mathbb{R}^2, Cm_{x,y} \in \mathbb{R}^2, Cm_{u,v} \in [0,1]^2, Cm_{face} \in [0...6], RGB \in [0,1]^3\}$, while the complete cubemap is defined as a unit cube in 3D space with normalized coordinates $C_{cube} := \{C_{cube} \in \mathbb{R}^3, \vec{P} \in [-1,1]^3\}$. Im can be mapped onto a unit sphere object, which is aligned with C_{cube} coordinates in 3D space. This unit sphere object is defined as $\mathbb{B} := \{\vec{M} \in [-1,1]^3, r, \phi, \theta\}$ (Sec. 7.5.2).

The polar coordinates of such a sphere can then be mapped to Cartesian coordinates, in order to enable the sampling of the equirectangular image pixels onto the corresponding cubemap face. These coordinates can be defined as Cr_{xyz}, where $Cr_x = rsin(\theta)cos(\phi)$, $Cr_y = rsin(\theta)in(\phi)$ and $Cr_z = rcos(\theta)$, and where $\theta = u2\pi$ and $\phi = v\pi$. The coordinates from $Im_{u,v}$ are sampled as normalized coordinates $u, v \in [0,1]^2$, where $u = Im_x/Im_{width}$ and $v = Im_y/Im_{height}$, and where $Im_{x,y}$ represent the current x, y pixel coordinates $Im_{x,y}$.

The procedure of mapping a cubemap image to an equirectangular map involves projecting a ray $\vec{R}$ from $\mathbb{B}_{center}$, converting the spherical coordinates of $\vec{R}$ to Cartesian coordinates, recording the pixel the ray intersects on one of the faces Cm_{face} of C_{cube}, and copying the corresponding pixel color value from the current Cm_{face} into $Im_{x,y}$.

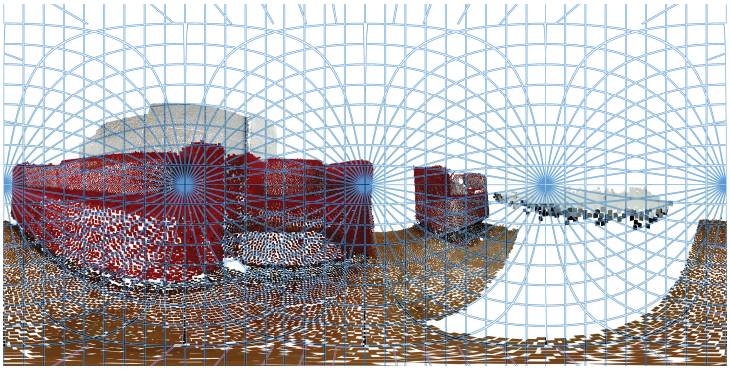
To determine what face of the cube the ray intersects, the absolute value of the sampled pixel position in 3D Cartesian coordinates is computed as $x \rightarrow xx = \frac{x}{maximum}$, $y \rightarrow yy = \frac{y}{maximum}$, $z \rightarrow zz = \frac{z}{maximum}$, and where $maximum = max(abs(x), abs(y), abs(z))$. The reverse of this procedure is therefore the final step in the process to generate cubemap images from an equirectangular image. Each pixel position from each of the faces of C_{cube} (e.g., $Cm_{face_{pos_x}}$, $Cm_{face_{neg_x}}$, $Cm_{face_{neg_z}}$) is mapped back to spherical coordinates in order to sample the pixels in the equirectangular image.

The sampling is accomplished also using a ray that is projected from C_{cube} into $\mathbb{B}$ (onto which the equirectangular image is projected). Additionally, when projecting a ray from each pixel in Cm, the location of the pixel in the relevant cubemap face needs to be normalized between $[-0.5, 0.5]$, and these 2D coordinates are used to form the redefined unit 3D vector $\vec{R}$, where the third coordinate is related to the absolute normalized value for the current face projection.

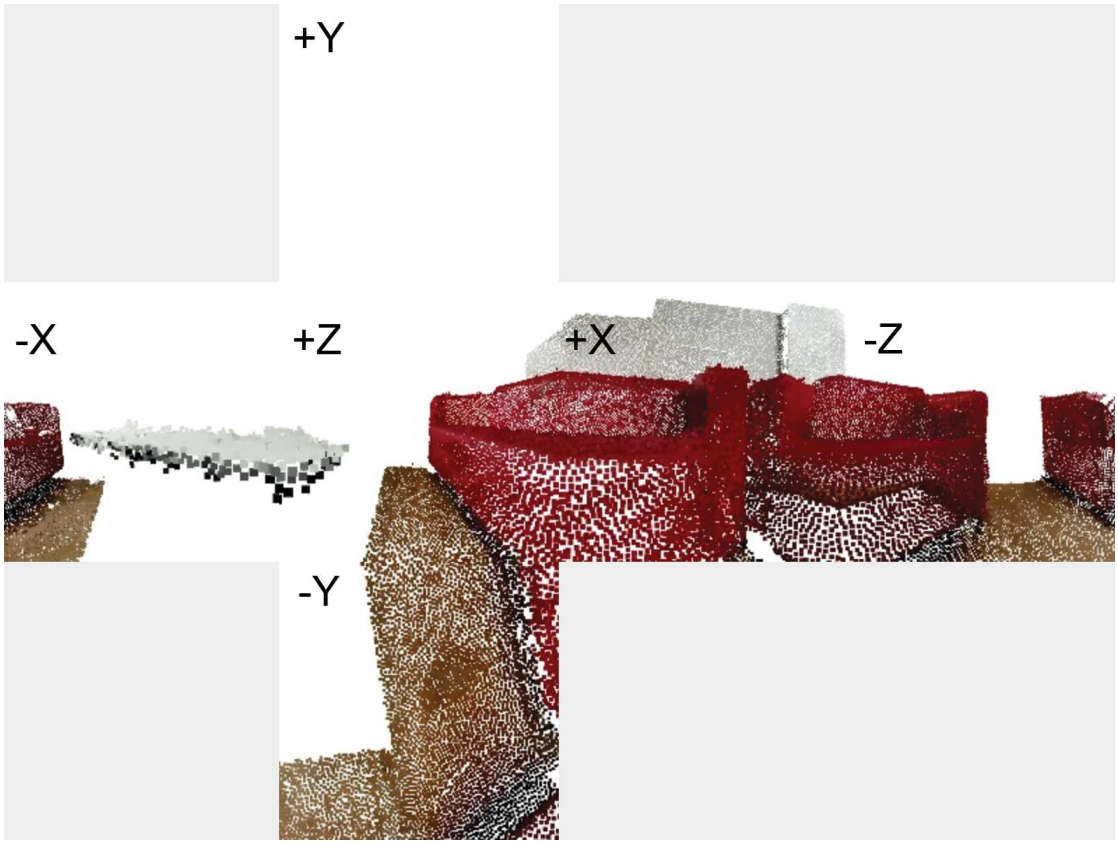
Once the Cartesian coordinates of $\vec{R}$ have been established, they are converted back into spherical coordinates, so that $\vec{R} \rightarrow \vec{R_s}$ can be used as a ray for an intersection test back into $\mathbb{B}$. In this case θ and ϕ are redefined as $\theta = atan2(\frac{\vec{S_x}}{\vec{S_y}})$ and $\phi = acos\frac{\vec{S_z}}{R}$, and where $R = \sqrt{\vec{S_x} \cdot \vec{S_x} + \vec{S_y} \cdot \vec{S_y} + \vec{S_z} \cdot \vec{S_z}}$.

The final step of this reverse mapping procedure is to redefine u and v as coordinates for sampling the pixels in Im. Therefore, u is redefined as $u = \frac{\theta}{\pi}$, and $v = \frac{\phi}{\pi}$. This conversion process from equirectangular projection to cubemap projection is defined in Alg. 4, and further illustrated in Fig. 7.9.

(a) *The indoor point cloud represented in the maps.*



(b) *An equirectangular projection captured from a given center point of an octree node partitioning the indoor point cloud.*



(c) *The resulting cubemap faces generated from the equirectangular projection.*

Figure 7.9: *Example of conversion of an equirectangular projection image to corresponding cubemap faces, representing an area of an indoor point cloud.*

Algorithm 4 Equirectangular to Cubemap

Require: Im, Cm, C_{cube}, $\mathbb{B}$, $\vec{R}$, Cr_{xyz}
  $Im \rightarrow \mathbb{B}$ {Map equirectangular image to unit sphere}
  $Cm \rightarrow C_{cube}$ {Map cubemap faces to unit cube}
  $C_{cube_{\vec{P}}} = \mathbb{B}_{\vec{M}}$ {Set center point of unit cube to the center point of unit sphere}
  for $C_{face} = 0$ **to** $length(C_{cube_{faces}})$ **do**
    for $x = 0$ **to** $length(Cm_{face_{width}})$ **do**
      for $y = 0$ **to** $length(Cm_{face_{height}})$ **do**
        $Cr_{xyz} \in [-0.5, 0.5]^3$ {Normalize Cartesian coordinates projected from unit cube into unit sphere}
        $\vec{R} = Cr_{xyz}$ {Assign projected Cartesian coordinates to unit vector}
        $\vec{R} \rightarrow \vec{R_s}$ {Transform Cartesian coordinates of sampling vector to spherical coordinates}
        $\vec{R_s} \in [0, 1]^3$ {Normalize spherical coordinates to unit vector}
        $\vec{R_s} \rightarrow Im_{u,v}$ {Sample UV coordinates of equirectangular map}
        $Im_{u,v} \in [0, 1]^2$ {Normalize UV coordinates}
        $Cm_{x,y} = Im_{u,v}$ {Map sampled UV coordinates onto relevant cubemap face}
      end for
    end for
  end for

7.5.2 Viewpoint Entropy-based Multiview Classification

The use of optimal viewpoints is based on the concept of *viewpoint entropy* [39]. Entropy is the measure of information conveyed for a given signal, and can be applied to images in order to measure *how much* information is contained in them. This approach is inspired by the concept of Shannon entropy [223]. The viewpoint entropy-based multiview classification algorithm generates images of clustered point cloud segments, using an entropy-based function to select multiview images with the highest amount of visual information.

The entropy value used to select multiviews for image synthesis is obtained as a vector parallelism measure between the randomly sampled cluster bounding sphere vertices (used as virtual camera positions and directions), and point cluster normal vectors. This concept is illustrated in Fig. 7.10.

Viewpoint Entropy-based Multiview Classification Algorithm The formal mathematical definition of selection of viewpoints is described. A point cluster C is defined as $C := \{P \in \mathbb{R}^3, \vec{N} \in [-1, 1]^3, RGB \in [0, 1]^3\}$. For each cluster C representing a finite set of points, a bounding sphere is generated, which is defined as a 3D sphere object, with additional geometric divisions, as $\mathbb{B} := \{\vec{M} \in [-1, 1]^3, r, S_w, S_h, \phi, \theta\}$, with $\vec{M}$ being the center point, r the sphere radius from the center point, S_w and S_h horizontal and vertical sphere divisions, and ϕ and θ the horizontal and vertical sweep angles.

The bounding sphere B generated for each cluster, based on $\mathbb{B}$, can be defined as $B := \mathbb{B}(C)$. Vertices are then randomly sampled from the bounding sphere, defined as subset B_v, where $B_v = \sigma(B) := \{P \in \mathbb{R}^3, \vec{N} \in [-1, 1]^3\}$. Each of the randomly-sampled inverted bounding sphere vertex normal vectors are tested for parallelism with the normal vectors from each of the points in C.

The *Entropy Function* is the main function used to determine the selection of optimal viewpoints, and is defined as $f : \mathbb{R}^3 \to \mathbb{R}$, with $\mathbb{R} \to \{0, 1\}$, where $f(C_p, B_p) = \begin{cases} 1, T \geq C_{\vec{N}} \cdot B_{\vec{N}} \leq 1.0 \\ 0 \end{cases}$, and where the bounding sphere vertices and cluster points that have normal vectors considered close to parallel within threshold level T (e.g., if the *dot* product is between 0.75 - 1.0, and this is within the range of T), are marked as $TRUE$ positions and directions for the virtual camera. The normal vectors of the segmented point cloud can then be pre-computed (Chpt. 4.3.4), with a preferred normal orientation along the positive Y-axis.

The virtual camera object is used to synthesize the multiview 2D images is defined as as $Cm := \{\vec{L_d} \in \mathbb{R}^3, \vec{L_u} \in \mathbb{R}^3, P \in \mathbb{R}^3\}$, where $\vec{L_d}$ represent the camera direction, $\vec{L_u}$ the camera up vector, and P as the camera position. This approach is further summarized in Alg. 5.

Algorithm 5 Viewpoint selection

Require: C, B_v, Cm
  $B \leftarrow C$ {Generate bounding sphere around each point cluster}
  $B_v = \sigma(B)$ {Randomly sample vertices from the bounding sphere}
  for $i = 0$ **to** $length(B_v)$ **do**
    for $j = 0$ **to** $length(C_P)$ **do**
      if $T \geq C_{\vec{N}} \cdot B_{\vec{N}} \leq 1.0$ **then**
        $Cm \leftarrow B_p, B_{\vec{N}}$ {Set camera position and direction to bounding sphere vertex position and inverted normal}
      end if
    end for
  end for

Cluster Generation Prior to generation of the viewpoint entropy-based multiview images, the point clusters used for classification need to be determined. The use of k-means or DBSCAN clustering can be used to cluster regions of points for multiview classification (Fig. 7.11). The desired amount of clusters corresponds to the visual observation of how many are required to sufficiently partition the given point cloud. The spatial distance between each of the points is used as the main attribute for both clustering approaches.

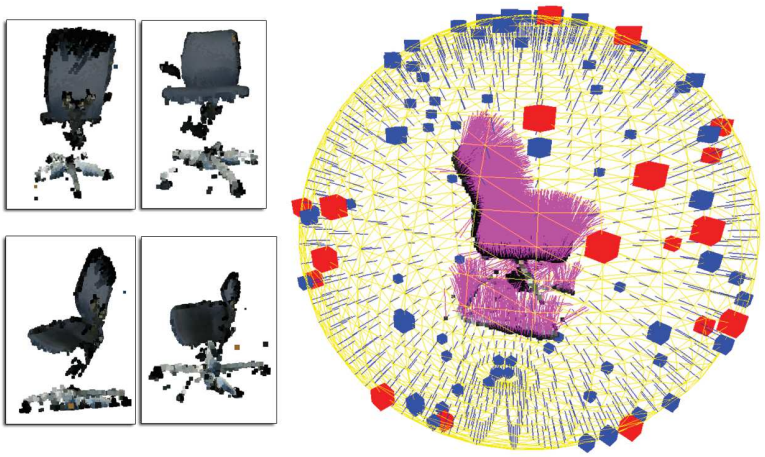


Figure 7.10: *Example of selection and generation of multiviews for a chair object point cluster. Bounding sphere vertices (blue) are selected as multiview camera positions and directions (red), based on the parallelism entropy measure between the inverted bounding vertex normal (blue) and the point cluster normal vectors (purple).*

(a) *The input point cloud representing common office furniture.*

(b) k-*means clustering with nine clusters.*

(c) *DBSCAN clustering result, with seven generated clusters.*

Figure 7.11: *Examples of* k*-Means and DBSCAN clustering.*

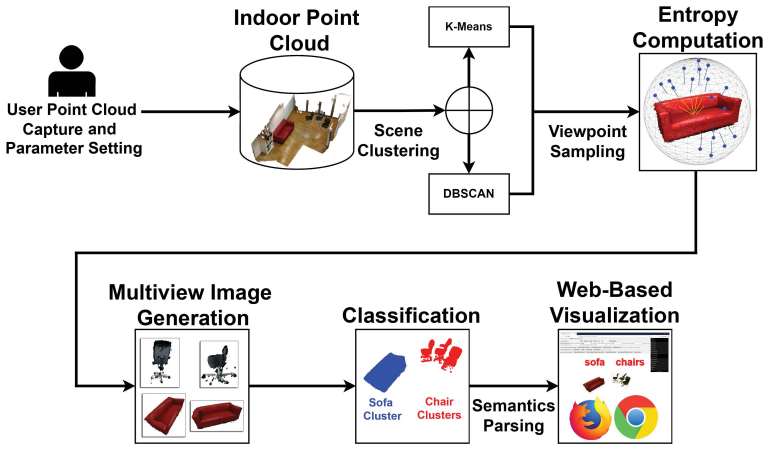


Figure 7.12: *A flowchart illustrating the process of viewpoint entropy-based multiview classification.*

Viewpoint Entropy-based Multiview Image Generation By using optimal viewpoints for taking images of 3D point clusters, the amount of time and data required for classification can be reduced, and such an approach is suitable for use within an SOS implementation alongside Web3D-based visualization (Fig. 7.12). The views captured as 3D perspective projections of point clusters are selected such that the most useful amount of visual information, i.e., those with highest entropy, is shown.

The viewpoint entropy-based multiview classification method takes in an RGB point cloud. It is assumed that the RGB point cloud has planar regions representing walls, floor and ceilings removed using segmentation (e.g., using RANSAC or Region Growing). Once the remaining point clusters representing furniture objects have been generated (using either DBSCAN or k-means clustering), for each cluster a bounding sphere with 994 vertices is generated.

The bounding sphere encloses each furniture object point cluster. A shuffling method based on Fisher, Yates, *et al.* [80] is used to shuffle an array copy of the generated bounding sphere vertices, whose position and inverted normal vectors are then used to randomly sample a portion of as a set. This set is in turn used for sampling potential camera directions and positions.

The inverted sphere vertex normal vectors from this set are compared to each of the corresponding 3D point cluster normal vectors. This process is repeated for each cluster in the 3D point cloud. An average of 11 images per point cluster are generated with this approach. The generated images are then sent for classification.

7.6 Object-based Classification

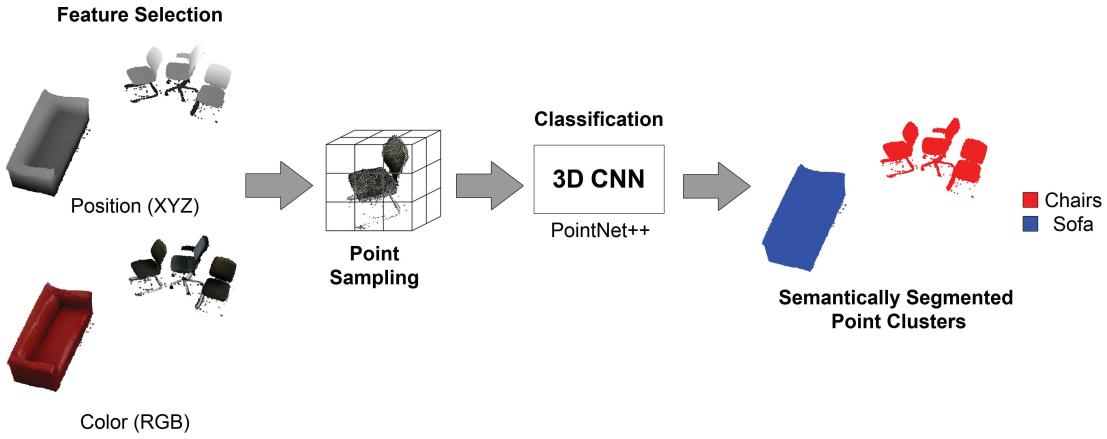


Figure 7.13: *A simplified illustration of semantic-segmentation using a 3D CNN (PointNet++). The two main features of point clouds used for classification are spatial position of points in 3D space, and their color. The input point cloud is then iteratively sub-sampled via convolutional operations, resulting a final classification value that is then mapped back to the relevant point clusters, thus semantically enriching them.*

The use of 3D CNNs for classification tasks has proven to be a viable option for enrichment and segmentation of point clouds (Fig. 7.13). A notable CNN for classification of point clouds is PointNet++ [195]. The PointNet++ CNN architecture is able to learn specific features of point clouds in a supervised manner, and apply this model to semantic enrichment and segmentation of various point cloud models - including point clouds representing the built environment [161] and urban scenes [275].

PointNet++ is able to classify point clouds regardless of their scale, density or orientation, and is invariant to permutations. It uses local distance between neighbouring points projected in Euclidean space to enable direct feature learning from point cloud training data, and recursively sub-samples point regions for evaluating the local features of a point neighbourhood. The training of the network is based on mapping an input point set to a vector representation that is first fed to a multilayer perceptron, then to a max pooling layer. The global feature vector used for generating the output score is obtained from the subsequent local feature vectors of sub-sampled regions.

7.7 Implementation of Supervised Deep-Learning Approaches

Key SOS Component Implementation The supervised deep-learning methods are implemented as SOS software sub-components within the *Point Cloud Processing* and *Classification* components of the prototypical *Point Cloud Classification* SOS implementation (Chpt. 3.5.2). The multiview and object-based classification is implemented with three different system components that capture user control and data input, perform processing both on the client and server-side, and stream the results back to the client.

The main task of the server-side processing includes the classification of 2D image and 3D point cloud data. The client-side system implementation is used to load in the point cloud, generate the 3D partitions using a given clustering method, and send this to the server for processing. Optionally, the client-side could be used instead of the server-side system to generate the multiview images, assuming the client is a higher-specification device (i.e., *Thick Client* or *Medium* client). The server-side classification system performs the classification and streams the results to the *Semantic Labelling* sub-component, which in turn sends the classified point clusters to the *Representation Processing*.

For the prototypical SOS implementation, the final visualization result is generated client-side, using a Web3D-based visualization system (based on the Three.js framework). The visualization system is used to display the classification results to the user as well as to enable further inspection and annotation of the classified 3D point cloud.

2D CNN Retraining and Classification The Inception V3 CNN architecture was used to retrain two CNN models for classifying common office furniture items (chairs, tables and sofas). The retraining of the CNN was implemented using TensorFlow 1.11 and Python 3.6. The version of Tensorflow 1.11 is compiled for AVX2 CPU instructions for cross-system compatibility, and did not have multi-GPU support enabled. Only the last bottleneck layer of the complete CNN model was retrained; Inception V3 was originally trained using ImageNet dataset [208]. For the training data, 9759 different RGB images of chairs, tables and sofas were used (with a 70/30 split for training and validation images). These images were obtained by using a batch image download script for Google Image search written in Python.

Additionally, the images were reviewed and those that featured pictures of furniture items without too much visual clutter were selected and resized to 300×300 pixels (with the aspect ratio preserved). The training data input vector size is $300 \times 300 \times 3$ elements. Random distortion of training data (brightness, scale, and cropping) was not utilized for the retraining. The predicted classification accuracy using the retrained Inception V3 CNN is 92.9%, using 4000 training steps with a learning rate of 0.01.

The second version of the CNN is used for classifying scenes with only chairs and tables, being retrained using the same training parameters, except using only photos of chairs and tables. The second CNN model has a predicted classification accuracy of 94.5%. Initial classification experiments showed that using a CNN with only two object furniture classes for scenes that feature only two types of furniture provides more accurate classification results.

For both the octree and multiview-based classification approaches, the generated multiview images are resized to 300×300 pixels and saved as Joint Photographic Experts Group (JPEG) images prior to classification. The classification probability scores for the input image data are calculated based on a linear softmax function.

3D CNN Training and Classification A PyTorch [1] implementation of the PointNet++ 3D CNN architecture was used for training and testing the CNN for semantic segmentation of indoor point clouds. Two different versions of the PointNet++ model were trained, using the S3DIS dataset [10] (one with and one without additional RGB features of the point data). Both the RGB and non-RGB versions of the PointNet++ CNN model were trained using 112 different scenes from the S3DIS dataset, featuring typical indoor offices (with a 81/31 split for training/testing data).

The CNN was trained for 200 epochs, with a learning rate of 0.001. 8192 points were sampled for each training cycle. The validation accuracy for the RGB version of the CNN was 64.4%, while for the non-RGB version it was 67.2% (this includes the validation accuracy of the whole scenes, not just furniture objects).

Point normals were not used as a training feature, since by default the S3DIS set does not contain any normals. A small batch size of one was chosen due to computational hardware restraints of training the CNN on a commodity computer (though if access to GPU clusters is available, the batch size used for training, and the number of training cycles, can be increased). The S3DIS already featured labelled data, which was used later on to generate ground truth models to test the accuracy for semantic segmentation of the specific scenes (for evaluating the multiview classification approach as well).

Octree-based Multiview Classification Implementation At each octree node that contains a point cluster, a virtual camera position is computed. This virtual camera generates an equirectangular projection from the center of the current node, while the visibility of other octree nodes is disabled (this also prevents occlusion and points from other nodes intruding the current node whose equirectangular map is being generated). The cubemap images are generated from a equirectangular projection (Alg. 4), which captures all the points in a given octree node from the center view from inside the current node. This approach enables the capturing of the complete environment around each node center as a single image. The generation of the cubemaps is handled via a *fragment* shader [5] and saved as a JPEG image.

The *Multiview Generation* sub-component of the SOS implementation (Chpt. 3.5.2) then generates six cubemap images from the single equirectangular projection image. The *Multiview Generation* sub-component also checks the generated cubemap images to determine if they have an average color value of less than 250 for each of the RGB color channels. These faces are then marked as being valid, otherwise they are marked as invalid and are not classified. This prevents images with too much whitespace (the default background color), from being classified. File names of the generated cubemap faces that are marked as valid are given a numerical ID that corresponds to a specific octree node in the scene. Each of the valid cubemap images are then classified as having a probability of representing one of the furniture object classes, with the result mapped to each corresponding octree node (via the *Semantic Labelling* sub-component).

[1] PyTorch: `https://pytorch.org/`

Multiview Classification Implementation For implementation of multiview classification, communication with the server is established via the Sockets.io library using an bi-directional client/server architecture.

The server listens to any communication by the client from a given port, such as incoming SEND responses for receiving data and GET responses for sending classification results. The server calls the image classifier script implemented in Python (using the Tensorflow library). Once the classification results have been generated by calling the image classifier Python script, the server loads in the classification results (saved as text files on the server), parses them and sends the results back to the client as JSON data.

The result for each corresponding node that the valid cubemap faces were generated from is then averaged. The server then removes the generated cubemap faces and results once the classification has been completed.

Object-Based Classification Implementation The object-based classification method is used to generate semantically segmented versions of input point clouds. The point clouds do no need to be segmented to remove wall, floors and ceilings, as these object classes are classified by the retrained CNN, in addition to detecting occluding cluster features (e.g., furniture objects). The object-based classification implementation makes use of both RGB and non-RGB point clouds, without any normals. The classification system is implemented in Python using PyTorch, where the classifications script loads in a point cluster in the Hierarchical Data Format (HDF5) [2].

The HDF5 file format is used to initially generate the ground truth models, which combine the actual point cloud in the PLY file format, along semantic labels for each of the objects classes that the clusters in the point cloud belong to. During classification, the classified cluster regions of the point cloud are compared with the ground truth point clusters in order to predict the classification accuracy.

The point clouds that are to be semantically segmented can either be stored on the server or on the client, before being labelled by the *Classification* sub-component of the the server-side *Point Cloud Processing* component. The labelled point clusters are then sent to the *Representation Processing* component, from where they can be processed further for reconstruction and visualization tasks.

7.8 Case Study

Experimental results for the multiview and object-based classification approaches are presented. These results were previously published and discussed in Stojanovic *et al.* [238, 236, 241, 239]. The results are presented as three different case studies, each one focusing on a particular deep-learning classification approach.

The results for viewpoint entropy-based multiview and object-based classification accuracy were obtained by detecting and finding the average of intersection points between the ground truth and predicted point sets. An intersecting point is defined as having the same location and color as the corresponding point in the ground truth set.

[2]HDF5: `https://www.hdfgroup.org/solutions/hdf5/`

The results for the octree-based multiview classification accuracy were obtained by comparing classified octree nodes between the predicted results and corresponding ground truth data. The point clouds that were used for generating results feature positional and RGB color attributes.

7.8.1 Detection of Common Office Furniture

The two multiview classification approaches were initially designed for detection of common office furniture objects for inventory-related O&M tasks. These methods were evaluated using pre-segmented point clusters, where point clusters representing different furniture objects were used for testing.

Octree-based Multiview Classification Results For the evaluation of the octree-based classification approach, the number of correctly classified nodes are compared against ground-truth models. Each of the octree nodes are color coded to represent each particular furniture class (red for chairs, green for tables and blue for sofas). The accuracy of the classification result is compared against how many nodes are correctly labelled between the predicted model and the ground truth model – using the same octree resolutions for to spatially partition both models.

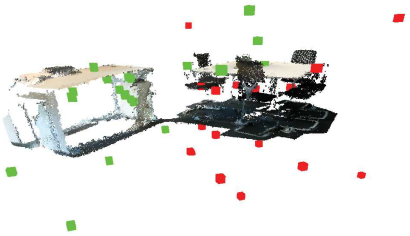
The eight selected test scenes were constructed using point clusters from a custom dataset and from an publicly available one [118]. The test scenes feature RGB point clusters with both simple and complex furniture arrangement as well as different point cloud resolutions. The test data also includes variations of the three different furniture types, with different spatial arrangements and rotations of multiple objects in a given scene. This allowed for testing the presented approach for dealing with versatility of point cloud data, which was captured using different sources.

The presented results show the classified nodes in respect to the spatial arrangement of objects in the scene. Tbl. 7.1 provides the ground truth as an object type composition overview for each test scene. Fig. 7.14-7.15 show the ground truth and predicted results.

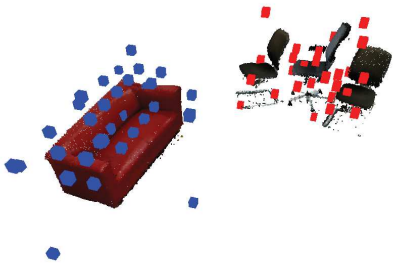
Viewpoint Entropy-based Multiview Classification Results Two different datasets were used for the evaluation of the viewpoint entropy-based multiview classification approach. The first dataset was based on a custom dataset captured using a commodity mobile device, depicting a common indoor office. The other dataset used was Area 1 from the S3DIS dataset, which was chosen as it features a high number of cluttered items in the desired classification categories (e.g., chairs, tables and sofas). The 3D point clusters featuring the furniture objects were extracted using manual segmentation. The k-means clustering method was evaluated as the primary method for generating clusters for multiview evaluation. The selected number of clusters for each scene was determined visually by counting how many objects of each type are present in the scene.

Table 7.1: *Ground truth table showing number of chair, sofa and table objects for each test scene, along with with number of points and the sampling size for the generated octree (maximum number of points sampled per octree node).*

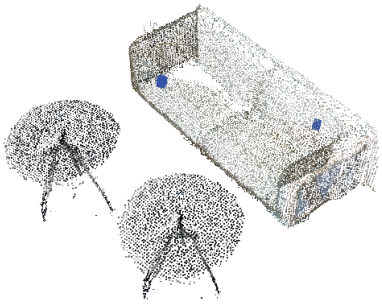
Scene	Points	Chairs	Sofas	Tables	Sampling
Set_1_1	48 783	6	0	2	30
Set_1_2	121 739	2	6	1	50
Set_1_3	36 797	3	1	0	20
Set_1_4	50 962	3	1	1	20
Set_1_5	56 176	3	0	2	50
Set_2_1	350 655	3	2	3	200
Set_2_2	1 097 427	4	2	1	1200
Set_2_3	16 166	0	1	2	30
Set_2_4	845 205	3	2	1	1000
Set_3_1	1 061 387	3	0	1	1200
Set_3_2	691 316	0	3	1	500
Set_3_3	703 299	4	0	1	500
Set_3_4	557 234	0	1	1	500
Set_4_1	10 974	4	0	2	5
Set_4_2	23 936	3	1	1	10
Set_4_3	12 524	3	0	1	7
Set_4_4	18 812	2	1	1	30



(a) *Ground truth for Set_1_1.*

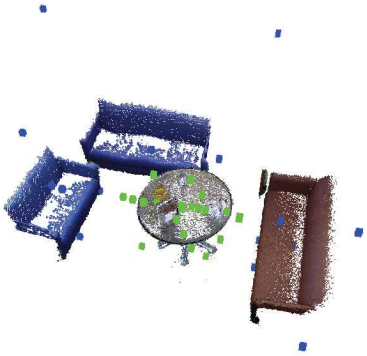


(b) *Ground truth for Set_1_3.*

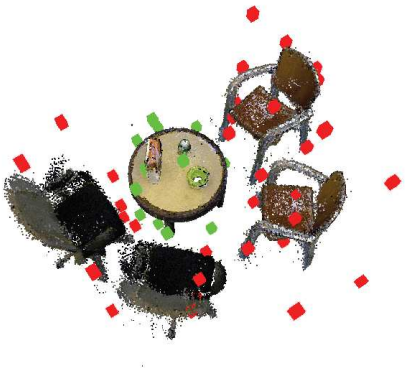


(c) *Ground truth for Set_2_3.*

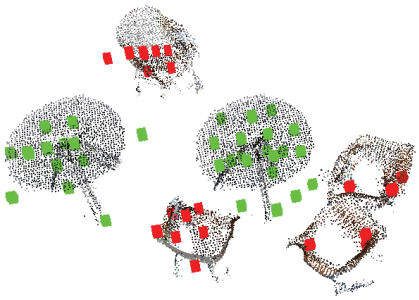
(d) *Ground truth for Set_3_1.*



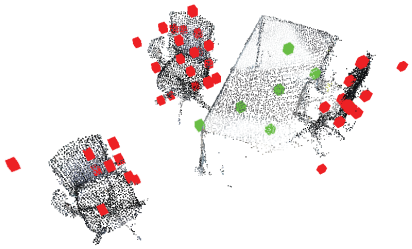
(e) *Ground truth for Set_3_2.*



(f) *Ground truth for Set_3_3.*

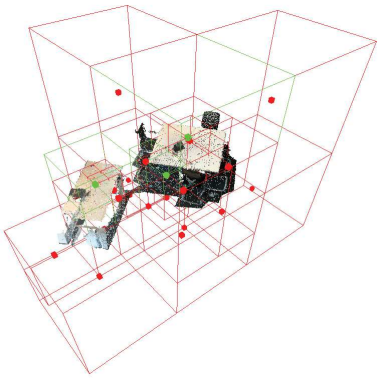


(g) *Ground truth for Set_4_1.*

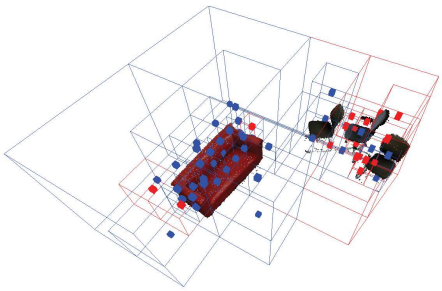


(h) *Ground truth for Set_4_3.*

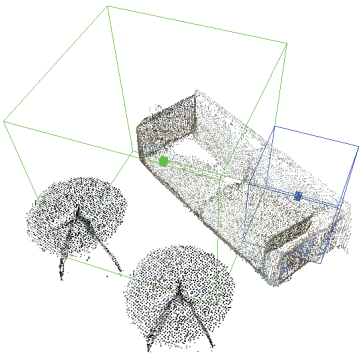
Figure 7.14: *Ground truth images for the test scenes used for the comparison of the octree-based multiview classification results.*



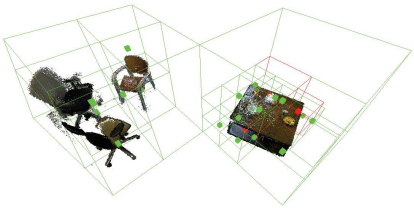
(a) *Set_1_1 - Valid Nodes: 42/67, Chair nodes: 34, Sofa nodes: 8, Table nodes: 8.*



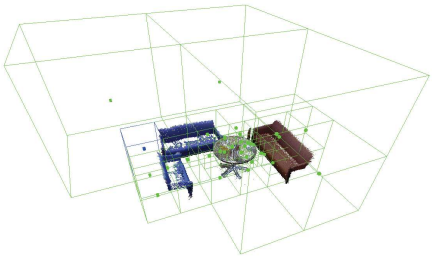
(b) *Set_1_3 - Valid Nodes: 61/84, Chair nodes: 22, Sofa nodes: 39, Table nodes: 0.*



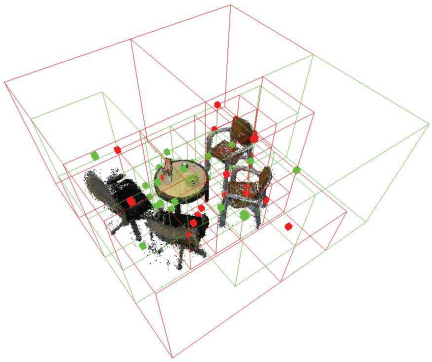
(c) *Set_2_3 - Valid Nodes: 2/20, Chair nodes: 0, Sofa nodes: 1, Table nodes: 1.*



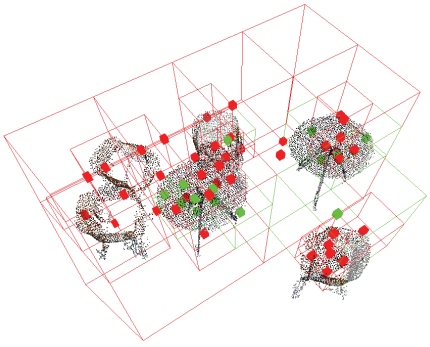
(d) *Set_3_1 - Valid Nodes: 32/39, Chair nodes: 7, Sofa nodes: 0, Table nodes: 25.*



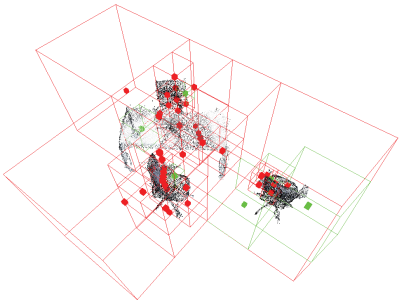
(e) *Set_3_2 - Valid Nodes: 40/64, Chair nodes: 0, Sofa nodes: 6, Table nodes: 34.*



(f) *Set_3_3 - Valid Nodes: 59/67, Chair nodes: 28, Sofa nodes: 0, Table nodes: 31.*



(g) *Set_4_1 - Valid Nodes: 43/87, Chair nodes: 30, Sofa nodes: 0, Table nodes: 13.*



(h) *Set_4_3 - Valid Nodes: 37/76, Chair nodes: 30, Sofa nodes: 0, Table nodes: 1.*

Figure 7.15: *Octree-based multiview classification results for the eight selected point cloud scenes.*

To reduce the number of generated multiview 2D images, while keeping the probability of correct classification at an acceptable percentage, it was decided to sample up to 36 different views per point cluster (3.6% of the bounding sphere vertices). Using this sampling size, an average of 11 multiview images are generated per cluster. The selected parallel threshold level was chosen to be 0.75, as in most cases it was required to sample a majority of parallel normal vectors when calculating the entropy for the veiwpoint selection. The results from the viewpoint entropy-based multiview classification show that the approach has an accuracy of 52.22% for the custom dataset (Fig. 7.16), and an average accuracy of 39.56% for the S3DIS dataset (Fig. 7.17).

7.8.2 Semantic-Segmentation of Indoor Point Clouds

Both RGB and non-RGB based semantic segmentation approaches were evaluated using PointNet++ on the 14 office areas from Area 1 of the S3DIS dataset (same as for the viewpoint entropy-based multiview classification). They contain three of the furniture object categories used to train the multiview CNN to classify the data (tables, chairs and sofas).

The obtained results from the object-based semantic segmentation of the S3DIS dataset show that it has an average classification accuracy of 83.51% for the version of the PoinNet++ CNN trained only on spatial point features, and an average classification accuracy of 81.14% for the version of the PointNet++ CNN trained using both spatial and RGB point features (Fig. 7.18).

7.9 Concluding Remarks

The methods for semantic enrichment are required in order to inject *meaning* into otherwise ambiguous point clusters. Semantically-enriched point clusters can then be used for further reconstruction, visualization and decision making tasks within the realm of FM-based applications – particularly for inventory-based tasks defined within the scope of O&M procedures.

Out of the three different approaches for semantic-enrichment (manual, unsupervised machine learning, and supervised deep-learning), the supervised deep-learning approaches provides the most promising results as they can be scaled to deal with large amounts of data (as manual segmentation and user-based annotation tends to be a very time consuming process, often requiring domain expertise). Additionally, supervised deep-learning can be applied to both classification of 2D and 3D data using different object-based and semantic segmentation approaches based on the use of CNNs.

Three different supervised deep-learning methods were compared, namely two multiview-based classification methods based on a 2D CNN, octree-based partitioning and viewpoint entropy-based, and an object-based approach using a 3D CNN for semantic segmentation. There is a trade-off between practicality and performance versus classification accuracy when deciding which approach to use for deep-learning-based semantic segmentation.

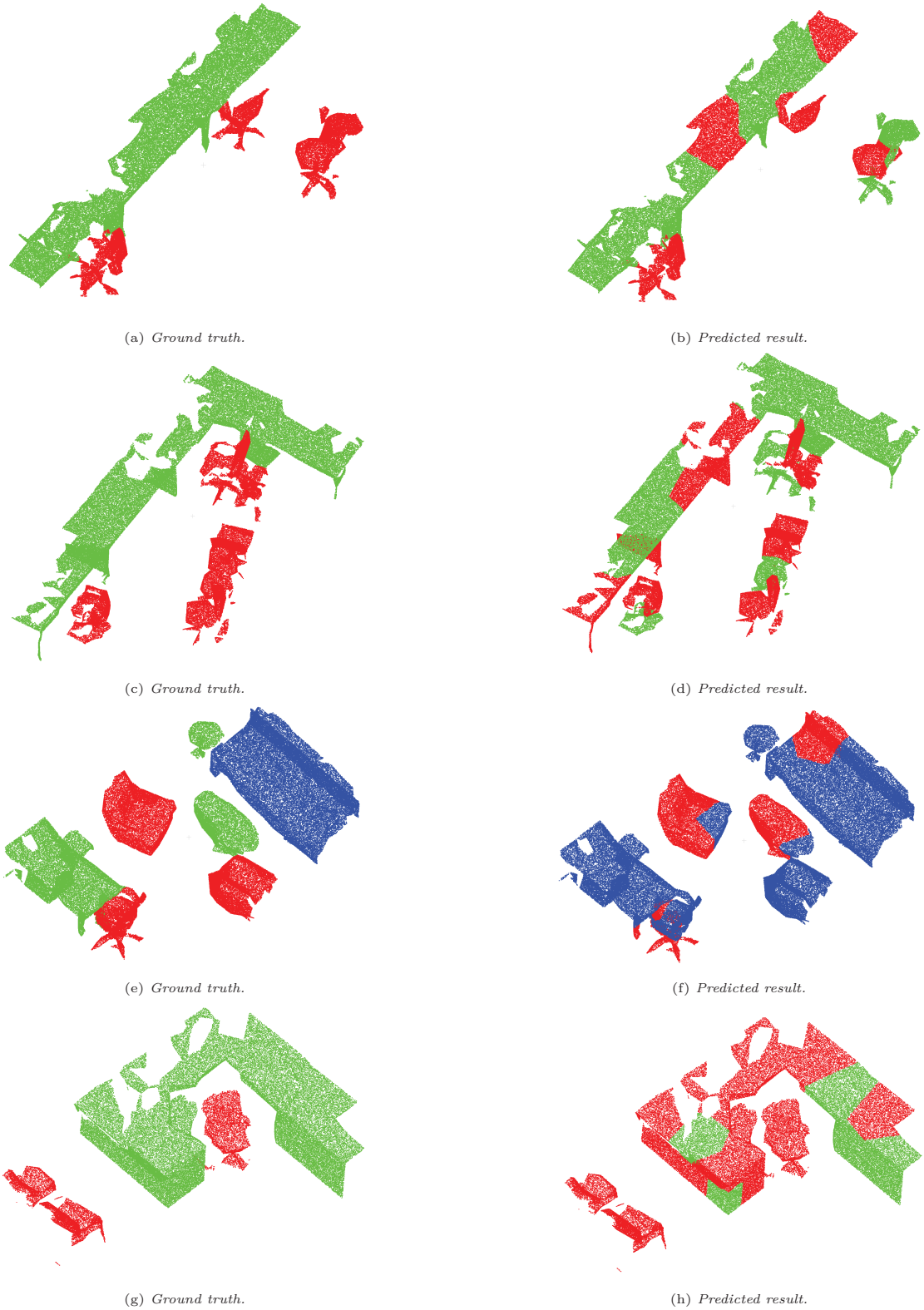
(a) *Ground truth.* (b) *Predicted result.*

(c) *Ground truth.* (d) *Predicted result.*

(e) *Ground truth.* (f) *Predicted result.*

(g) *Ground truth.* (h) *Predicted result.*

Figure 7.16: *Results comparison between ground truth (left), and predicted (right) multiview classification results with the custom dataset. Red point clusters indicate chairs, blue sofas and green table objects.*



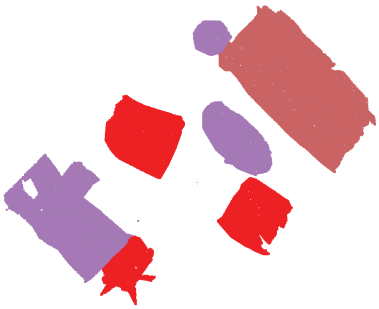
(a) *Ground truth.* (b) *Predicted result.*

(c) *Ground truth.* (d) *Predicted result.*

(e) *Ground truth.* (f) *Predicted result.*

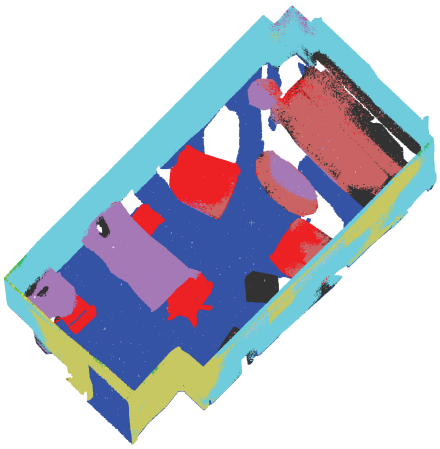
(g) *Ground truth.* (h) *Predicted result.*

Figure 7.17: *Results comparison between ground truth (left), and predicted (right) multiview classification results with the S3DIS dataset. Red point clusters indicate chairs, blue sofas and green table objects.*

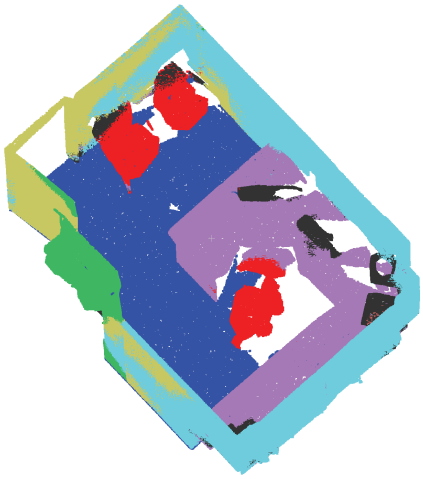


(a) *Ground truth.*

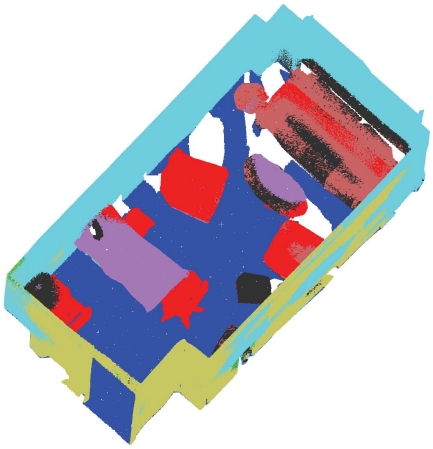
(b) *Ground truth.*



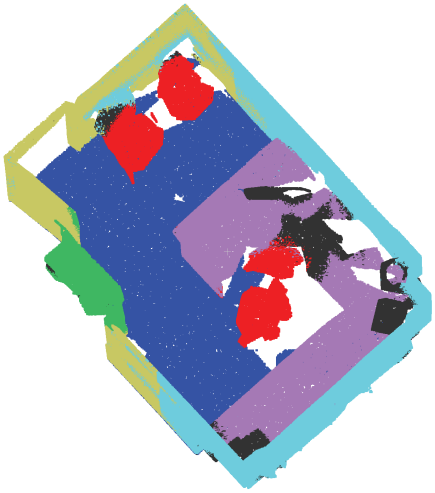
(c) *Predicted semantic-segmentation using non-RGB feature version of CNN.*



(d) *Predicted semantic-segmentation using non-RGB feature version of CNN.*



(e) *Predicted semantic-segmentation using RGB feature version of CNN.*



(f) *Predicted semantic-segmentation using RGB feature version of CNN.*

Figure 7.18: *Results comparison between ground truth (7.18(a)-7.18(b)), non-RGB feature trained version of PointNet++ CNN (7.18(c)-7.18(d)), and the RGB feature trained version of PointNet++ CNN (7.18(e)-7.18(f)). Red point clusters represent chair objects, copper represents sofas and purple represents tables.*

The two most important features for multiview classification are the color attributes of point clusters, and the shape of the clusters depicted in the multiview images (3D spatial features in Cartesian coordinates converted to 2D image space). However, for applications where classification accuracy is more critical, the use of a 3D CNN such as PointNet++ is a better option if high-end computing resources are available. The main challenge when using PointNet++ for semantic segmentation is that it relies on point cloud data for training, and this is impractical for integration with commodity computing hardware compared to the multiview-based classification approaches (as most implementations of PointNet++ are designed to leverage the computing power of parallel multi-GPU architectures). The retraining of the Inception V3 CNN with images of real-life furniture, used for the multiview-based approaches, takes only a fraction of the time compared to training the PointNet++ 3D CNN using the S3DIS dataset for example.

Comparing the the two multiview classification approaches - octree and viewpoint entropy-based, the later provides better classification results and does not generate as many redundant images for classification as the octree-based approach (which generates six cubemap images for each node regardless what type and features of point clusters are contained in the node - disregarding only images which feature a majority of the background color). Furthermore, an additional evaluation was conducted in previously published work [236], where the viewpoint entropy-based multiview method was compared against the octree-based multiview classification method, in order to establish which method is more accurate. The results show that the viewpoint entropy-based method has a higher classification accuracy of up to 25.3% in comparison to the octree-based multiview classification method.

Both multiview classification approaches can be considered appropriate for implementation as *lightweight* classification components within an SOS implementation (meaning they do not require extensive hardware and software resources to perform training and classification). While the multiview-based approaches offer worse classification accuracy in comparison to the object-based approach, they are generally more adaptable for use on commodity hardware, and retraining the 2D CNN (using only the last bottleneck layer), is more convenient with access to potentially millions of images on the internet depicting common office furniture in different categories (in comparison to having to find and use point clouds of such objects, which is often difficult).

One drawback of the viewpoint entropy-based multiview approach is that its multiview image generations relies on correctly calculated point normals that conform to the surface curvature (in terms of viewing the object from a given distance from its center with 3D perspective). For the octree-based multiview approach, an adequate resolution of the octree is required in order to capture macro details for classification (this also relies on using a point cloud with enough density in order to be able to visually depict non-whitespace regions in generated 2D images, which can then be classified).

Selected clustering approaches used for the viewpoint entropy-based multiview classification also require specific clustering parameter adjustment – based on the observed density and distribution of the point cluster.

Chapter 8

Processing and Visualization of Indoor Sensor Data

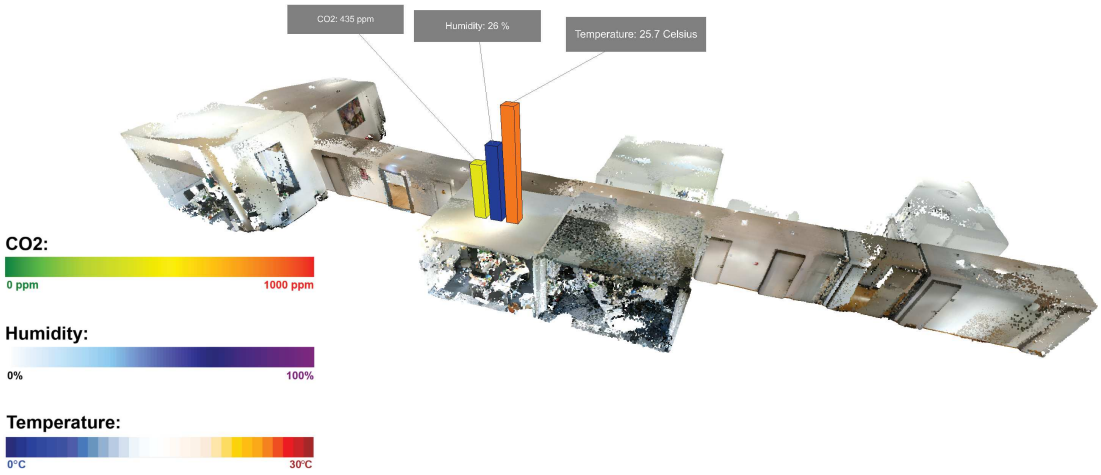


Figure 8.1: *Example of a combined visualization of various sensor data, mapped onto an indoor point cloud, and demonstrating the implemented visualization styles.*

Analysis of real-time and historic sensor data provides important insights into the operational status of buildings. There is a need for the integration of sensor data and digital representations of the built environment for furthering stakeholder engagement within the realms of Real Estate 4.0 and FM, especially in a spatial representation context. This chapter describes a prototypical SOS implementation (Chpt. 3.5.3), which processes sensor data, and generates visualization and analysis outputs. The key requirements and challenges for analyzing sensor data for indoor spaces within the scope of FM are:

1. The use of indoor point clouds for adding spatial context to the visualization.
2. The acquisition and processing of data from multiple sensors.
3. The interactive visualization using implemented SOS software components.

Furthermore, a prototypical implementation using the discussed software components demonstrates the application for integration, analysis and visualization of sensor data

from a typical office building, with the aim to communicate and analyze occupant comfort (Fig. 8.1). The empirical results presented in the case study are used to demonstrate the feasibility of utilizing such software components for the service-oriented integration of sensor data processing, analysis and visualization. The foundations and case study of this chapter are based on previously published work in Stojanovic *et al.* [237].

8.1 Overview of Sensor Data Analysis

Sensor data plays an important role in assessing the current state of the built environment. Once processed, the analysis and visualization results can be used to show important temporal changes in a given FM context. This may include occupancy comfort criteria e.g., room temperature or ambient noise levels, or important energy efficiency data e.g., power usage and carbon emissions.

Additionally, temperature and energy-usage data captured from indoor sensors has been used alongside BIM for the monitoring and analysis of building energy performance [172]. Such real-time or previously collected sensor data (e.g., historic sensor data) can then be visualized in combination with the as-is digital representation of the indoor environment (e.g., BIM geometry or indoor point cloud models), thus adding an important layer of information by associating the sensor data with a physical location.

Once captured, the sensor data can be processed and analyzed using implemented SOS components (notably the *Spatio-Temporal Data Processing* component as well as the *Data Mapping* sub-component of the *Representation Processing* component, Chpt. 3.5.3). The client visualization system can then map these values using different visualization methods for different FM scenarios (e.g., monitoring of temperature and humidity in an office). This allows for the application of visual analytics to enhance decision making and forecasting using quantifiable attributes of the built environment.

8.2 Service-Oriented System Implementation

The challenge of mapping and visualizing sensor data is addressed using a prototypical SOS implementation based on the conceptual reference SOA (Chpt. 3.3). While state-of-the-art methods promote IoT integration with BIM and associated technologies (e.g., IWMSs) [251], in practice the integration of sensor technology is complex due to the heterogeneity of sensor equipment (accuracy, availability, cost) and its interfaces, and the expertise required to combine these two areas in one *as-is* built environment representation. The presented approach therefore focuses on use of affordable hardware for sensing of room properties (e.g., temperature, humidity, and carbon dioxide concentration), and processing, analyzing and visualizing the captured raw sensor data.

The raw sensor data is filtered and stored by the *Spatio-Temporal Data Processing* component, from where it can be queried by the *Representation Provider* and external FM related systems (e.g., FMIS and CMMS). The filtered sensor data, along with the point cloud representation, is given additional attributes via the *Entity Provider*

component. This component extracts location and other information from existing building documentation (e.g., longitude and latitude coordinates and room dimensions acquired from the *as-built* BIM, room information contained in RFID or NFC tags, etc.), in order to enable automated or manual adding of semantics into the combined point cloud and sensor data information model. These semantics can then be used by stakeholders to identify important physical and location aspects, using the *as-is* point cloud representation.

The captured and filtered sensor data is computed server-side and the result is typically sent to a medium client, where the final visualization result is computed. In the presented case study, the processed sensor data is mapped onto the point cloud, using selected visualization techniques that are discussed in detail.

8.3 Processing of Sensor Data

The *Spatio-Temporal Data Processing* component is responsible for the storage, analysis, monitoring, querying, and web-based streaming of sensor data. Sensors can either be connected through the industry standard MQTT[1] protocol for lightweight data transmission, or by issuing HTTP POST requests, with both containing JSON-formatted data as the payload. There are numerous types of communication protocols for sensors based on low-energy, low-bandwidth radio transmission (e.g., ZigBee [2], Z-Wave [3], EnOcean [4], or LoRaWAN [5]), or wired bus connections. Additionally, software solutions e.g., KNX [6] are able to integrate various IoT data sources and perform analysis (e.g., energy usage monitoring). Low-cost edge controllers translate the local sensor data messages sent through such low-level protocols into JSON data packets that can be evaluated by the services of the SOS implementation (Chpt. 3.5.3).

For querying sensor data and corresponding metadata, such a platform can provide an API following the REST paradigm and JSON API specification (Chpt. 3.4.2). Further, live updates of sensor data values can be subscribed using a WebSocket message-based API. Through this, sensor value changes are propagated e.g., to the *Data Mapping* sub-component of the *Representation Processing* component to be visualized. The historical sensor data can be stored in a e.g., time series database [186] – which combines efficient access to sensor values and memory efficient data organization.

Additionally, a *Monitoring* sub-component (Chpt. 3.3) can be used to evaluate given rules for sensor data, and to generate event notifications in case of identified matches. For such generated event notifications, an *Alerting* sub-component can call previously registered notification targets (e.g., using specific IWMS or FMIS APIs).

[1] MQTT 5 OASIS Standard: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html
[2] ZigBee: https://zigbeealliance.org/
[3] Z-Wave: https://z-wavealliance.org/
[4] EnOcean: https://www.enocean.com/
[5] LoRaWAN: https://lora-alliance.org/about-lorawan
[6] KNX: https://www.knx.org/

Finally, the forecasting component can be used to analyse the current sensor data in order to predict the current state of a given physical environment or MEP component. In such a case, Bayesian probability can be used in order to predict the future state based on previously recorded sensor readings and other associated data [115]. Within a *Smart Building* paradigm, such systems would be able to work in cohesion and react to changes by triggering automated responses e.g., turning on the air-conditioning if the recorded temperature value is above a certain temperature at a given time of day.

8.4 Visualization of Sensor Data

8.4.1 Combined Visualization with Point Clouds

Visualizing sensor data with point clouds enables the depiction of the physical *locations* where sensor data readings are measured, and can enhance analysis, intervention, forecasting, and decision making processes within the context of FM (specifically O&M procedures).

In that sense, the combined visualization of sensor data and indoor point clouds forms a key component of representation for DTs. However, the linking of captured sensor data to the physical assets (e.g., rooms, MEP components) they refer to, and their combined visualization and resulting enhancement of decision making and communication, remains a key challenge in today's AECOO domains.

Semantically enriched point clusters are critical to this task, as semantics allow for the association of *labelled* geometry segments with sensor data that is physically located in the real-world counterpart of the 3D representation (e.g., labelled point clusters that distinguish different types indoor environment and building elements).

8.4.2 Use of Visual Analytics

Visual analytics enable key decision making tasks to be carried out through the use of intuitive graphical user interfaces, data processing, visualization, aggregation and analysis. Specifically, according to Keim *et al.* [132] *"visual analytics combines automated analysis with interactive visualisations techniques for an effective understanding, reasoning and decision making on the basis of very large and complex datasets."*

Key concepts of visual analytics can be applied to the processing, visualization and decision making based on analyzing acquired sensor data for indoor environments. The application of visual analytics to sensor data related to building occupancy comfort and performance is based on established methods used for analysis and visualization of spatio-temporal data. The process of applying visual analytics to data, such as sensor data, can be defined as a function, which transforms unprocessed data into visual outputs providing important insights – based on the hypothesis of what the data and visual outputs should provide and be driven by user interaction Keim *et al.* [131].

Furthermore, the application of this visual analytics process expands the visualization *mantra* proposed by Shneiderman [226]: *"Overview first, zoom and filter, then details on demand"* to *"Analyze First - Show the Important - Zoom, Filter and Analyze Further - Details on Demand"* [131] – making it applicable for use in visualizing analytical outputs using multi-variable and multi-dimensional data associated with AECOO domains (e.g., BIM [234] and CAD/GIS/spatio-temporal data [233]). Previous works concerning integration of visual analytics for sensor data related to buildings have successfully combined its use with BIM and FM data, in order to engage relevant stakeholders about the current state of a building [41, 49, 148]. These examples are described further in Chpt. 2.10.

8.4.3 Implementation of Visualization Components

In terms of generating visualization outputs, key software components used for processing the sensor data and generating the visualization outputs are utilized. These include the *Representation Processing* component, along with the *Data Mapping* and *Rendering Service* sub-components (Chpt. 3.5.3).

The *Representation Processing* component provides a set of services for generating the visualization result using the captured and processed point cloud and sensor data – in particular services for *geometry generation*, *mapping*, and *rendering*. The *Geometry Generation* sub-component is used to create the *as-is* geometry from the semantically-enriched point cloud or reconstructed geometry.

The *Data Mapping* sub-component is used to map the captured sensor data values to a given point cloud or reconstructed triangular mesh geometry partition. This may include sensor data that maps specific attributes e.g., temperature for a given room.

The *Rendering Service* sub-component is responsible for generating the final visualization result. The rendering service can make use of low or high-level graphics APIs and frameworks (e.g., OpenGL, WebGL, Three.js, etc.), in order to generate rendered images (usually in real-time), which can easily be displayed on the client side.

8.4.4 Visualization Methods

Since visualizing sensor data often requires reflecting the changes in real or semi-real time, the use of Web3D frameworks can be used to implement such visualization methods. For example, the use of WebGL-based frameworks allows for accessing the programmable graphics pipeline and utilize various per-vertex and per-pixel rendering techniques based on shader programs. This enables the implementation of real-time rendering methods for visualizing 3D geometry (e.g., point clouds, triangulated meshes), and to perform any texture-based rendering tasks (e.g., mapping the given sensor data readings to a given color value).

The visualization processing can either be implemented client or server-side, with the *Representation Processing* component streaming different stages of the complete visualization to the different client configurations (e.g., thin, medium and thick clients - Chpt. 3.3).

Three different visualization methods are used for graphical display of sensor data analytics: (1) thematic color-mapping, (2) textual data display, and (3) abstracted 3D geometric representation. The selected visualization methods are based on established *visualization idioms* used to convey geometric and textual abstractions of processed numerical data [102], and are described further in Chpt. 2.10.

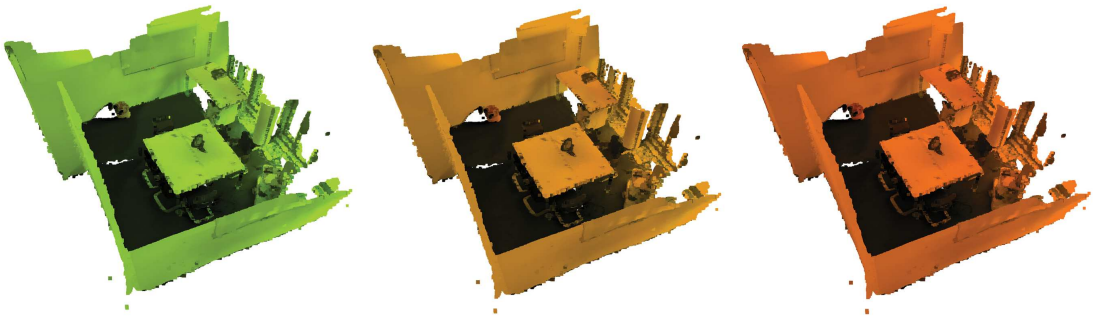
Additionally, 2D graphs can also be utilized to plot spatio-temporal data and to complement the overall visualization. In such cases, the use of 2D Web-based rendering frameworks e.g., (D3.js [7]), can be combined with the rendered 3D outputs in order to enhance the visualization by providing additional layers of information. Such layers of information, particularly 2D graphics, can sometimes be better understood by stakeholders who do not have experience in interpreting 3D visualizations [114]. Combining both 2D and 3D visual analytics in this sense allows for a complete visualization output of the analyzed spatio-temporal data, which has the farthest visual impact and engagement factor amongst stakeholders with different domain expertise levels [233, 28].

Thematic Color-Mapping The *Thematic Color-Mapping* method maps the sensor data variables (e.g., temperature, humidity, etc.), to a given color. If multiple sensor data variables need to visualized, each one can be mapped using e.g., different thematic color scales based on diverging, saturated or multi-hue color scales to visually represent the related changes (Fig. 8.2(a)). Such color scales were based on research by Engel *et al.* [73] looking at qualitative and quantitative evidence – collected from a user study evaluating impact of rendering techniques on information and spatial perception.

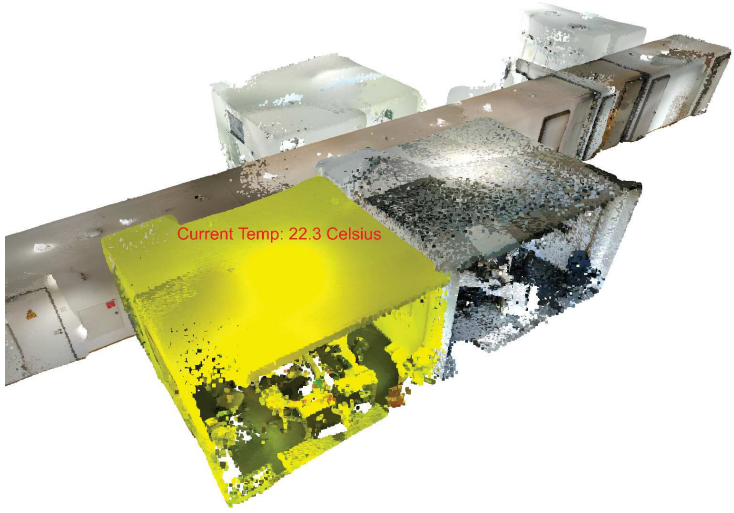
Textual Data Display The *Textual Data Display* method renders the parsed sensor data as a 2D billboard [5], displaying the current temperature. The billboard is further associated with each 3D cuboid used in the *Abstracted 3D Geometric Representation* method, using a line element whose starting point changes with the current height of the 3D cuboid (Fig. 8.2(b)). Billboard rendering is accomplished using the provided 2D text rendering functionality of the Web3D framework such as Three.js. This method is useful for giving better context alongside the other two visualization methods.

Abstracted 3D Geometric Representation The *Abstracted 3D Geometric Representation* method makes use of a vertically scaled 3D cuboid for 3D bar-graph style visualization (Fig. 8.2(c)). The scaling and the RGB color value of the cuboid is based on the current sensor value. This method is based on using an abstracted 3D solid to visualize spatio-temporal changes associated with the processed sensor data point that is currently being visualized, and is often used to emphasize changes over time [132].

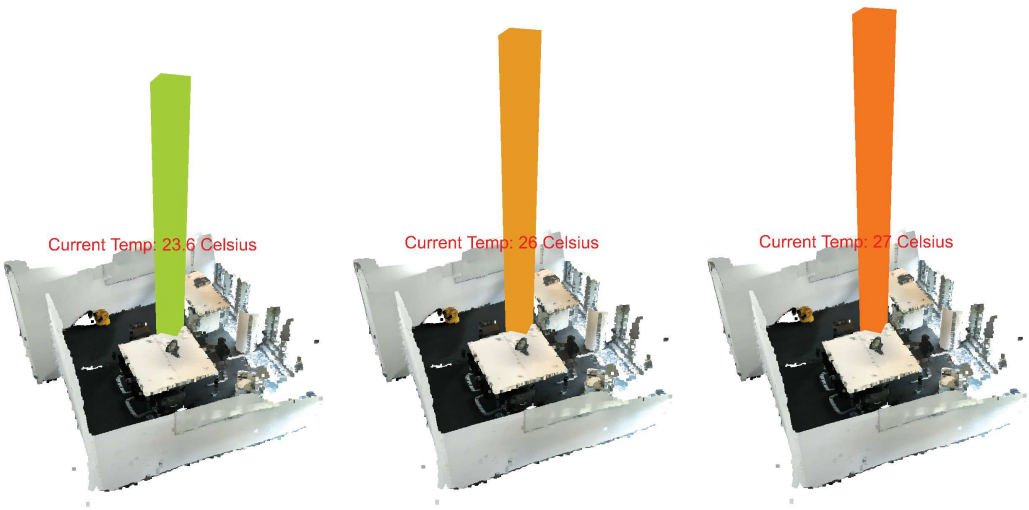
[7] D3.js: `https://d3js.org/`



(a) *Example of thematic color mapping, based on different sensor data values, mapped onto an indoor point cloud.*



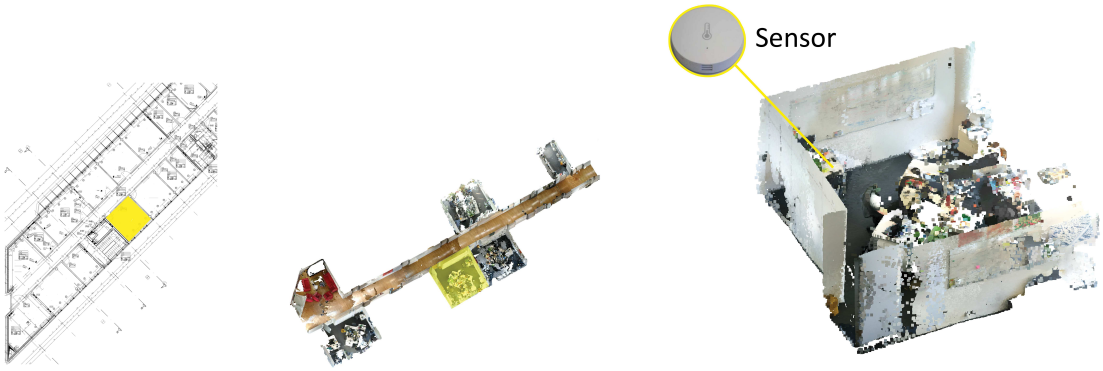
(b) *Example of textual sensor data display, projected as a 3D billboard texture.*



(c) *Example of the use of abstracted geometric representation visualized alongside a point cloud representation. In this example the 3D bar object changes in color and scale based on different sensor data readings.*

Figure 8.2: *Examples of different sensor data visualization methods using thematic color mapping, textual and abstracted geometry representations.*

8.5 Case Study



(a) *The office area floorplan.*

(b) *The captured and processed 3D point cloud used for the case study.*

(c) *Point cloud representation of the specific room according to the office floorplan, and an illustration of where one of the sensors (temperature sensor) is located within the room.*

Figure 8.3: *The indoor point cloud used in the presented case study.*

The main focus of the presented case study was to visualize room temperature, humidity and carbon dioxide recordings for occupant comfort assessment of a typical office environment – by linking this data to a given point cloud representation that visualizes the physical features of the corresponding room location. A point cloud of an indoor office area was captured and processed (Fig. 8.3(b)), based on the approaches discussed in Chpt. 4.2. While the entire office area was captured, the case study is focused on visualizing temporal changes for a single office where the sensors are located (Fig. 8.3(c)).

For the case study, captured sensor data was stored as JSON files on the server, and accessed to process sensor values obtained for a given time period. Three different sensor data variables were measured and visualized using the described visualization methods. These include: temperature, humidity and carbon dioxide readings. The readings for each of the sensor data variables were recorded over a period of one month. Since the sensors produced a large amount of sensor data, and for practicality in the case study, 100 time points in increments from the filtered sensor data were sampled and visualized (instead of all of the time points which were recorded once per hour for over a period of 30 days).

In order to visualize the sample sensor data, the time points needed to be normalized within given ranges that allowed them to be mapped to the scalar ranges used for the three different visualization methods. The maximum and minimum room temperatures, humidity and carbon dioxide levels were recorded in the sampled sensor data time-series. The minimum and maximum values were then used to interpolate between the selected color ranges every time new sensor data was parsed and sent to the visualization component.

Ranges from 0° to 30° Celsius were used for the temperature mapping, 0 to 100 percent for humidity mapping, and 0 to 1000 parts per million (ppm) for the carbon dioxide mapping. An additional 2D bar graph is also included alongside the main

visualization of each individual sensor data type to provide better visualization context in relation to previous and future sensor data values.

The presented empirical results of the case study show the visualization changes using two different time points (t_0, t_1), with different temperature, humidity and carbon dioxide readings (Fig. 8.4-8.7). Also shown are the simultaneous visualization results using a combined approach, featuring thematic color-mapping, textual and abstracted geometry visualization for each parsed time point of the temperature, humidity and carbon dioxide readings.

8.6 Concluding Remarks

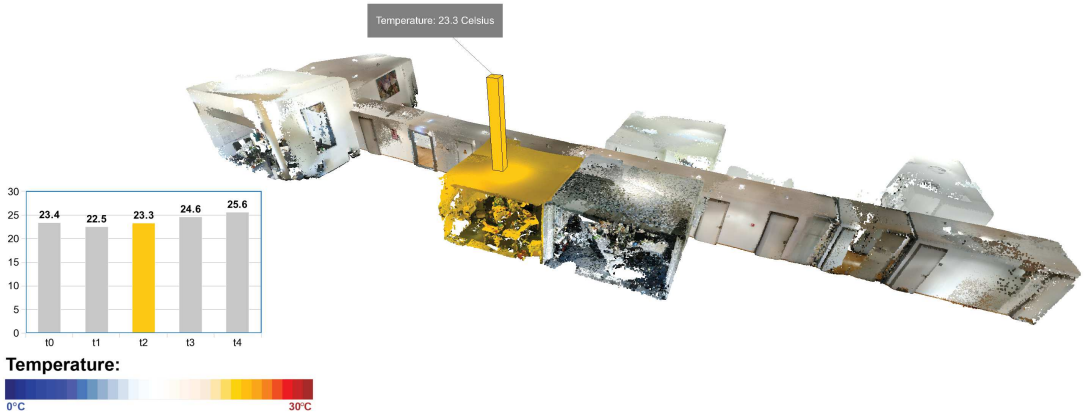
The experimental results obtained for the case study demonstrate a simple but effective approach for mapping acquired sensor data to an interactive 3D visualization, using visual analytics principals. The use of a spatially corresponding indoor point cloud adds an additional layer of information for better spatial association of the visualization results. In addition to this, the use of a prototypical implementation used in the case study demonstrates the following three key requirements outlined in the chapter introduction:

1. The prototypical application demonstrates that indoor point clouds, especially those acquired using photogrammetry and/or depth sensing with commodity mobile devices, provide enough visual information to enhance the spatially-aligned sensor data visualization. This means that they can be used as a practical alternative to BIM models for combined sensor data visualization.

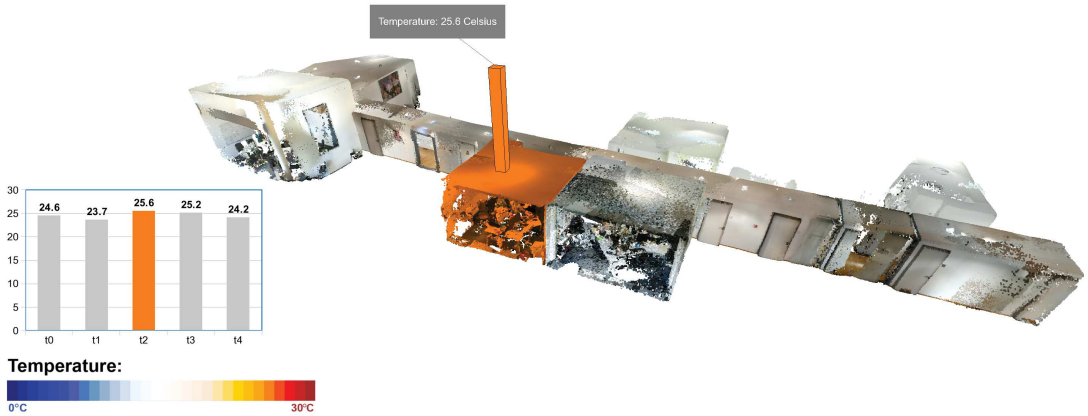
2. The results of the case study demonstrate that multiple sensor data sources can be utilized for capturing different natural and man-made phenomena from a given indoor environment, by processing and visualizing room temperature, humidity and carbon dioxide levels.

3. With the use of Web3D-based visualization frameworks, which are implemented as visualization components within a prototypical SOS, stakeholders can interactively and simultaneously visualize the indoor point cloud with the corresponding spatially correlated sensor readings for a given period of time – allowing the visualization system to intuitively present insights into the underlying spatio-temporal data through visualization to stakeholders.

Finally, the sensor data analysis and visualization approach has potential for integration as a flexible and practical solution for DTs and IWMSs platforms, which require the fusion of sensor data analytics and visualization for digital representations of indoor environments.

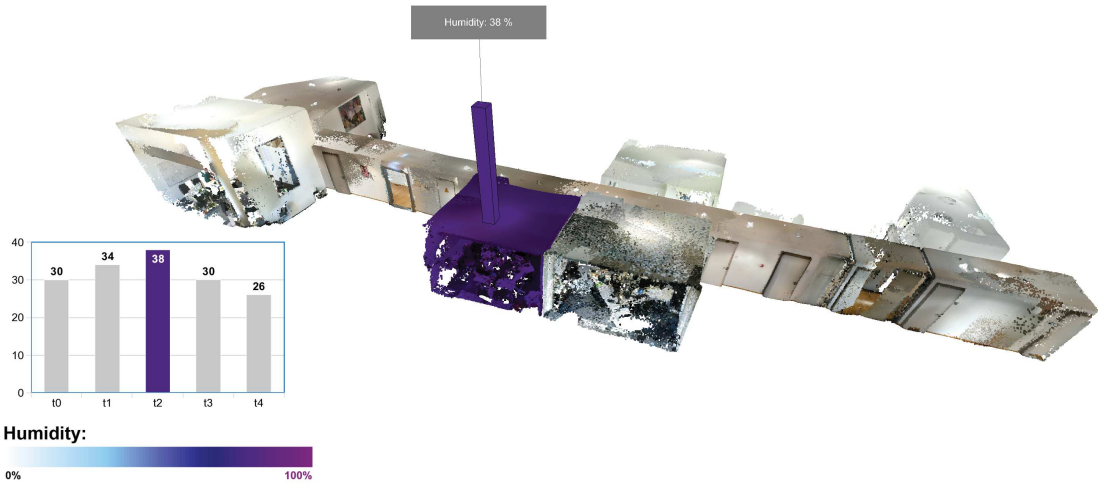


(a) t_0: 23.3° *Celsius.*



(b) t_1: 25.6° *Celsius.*

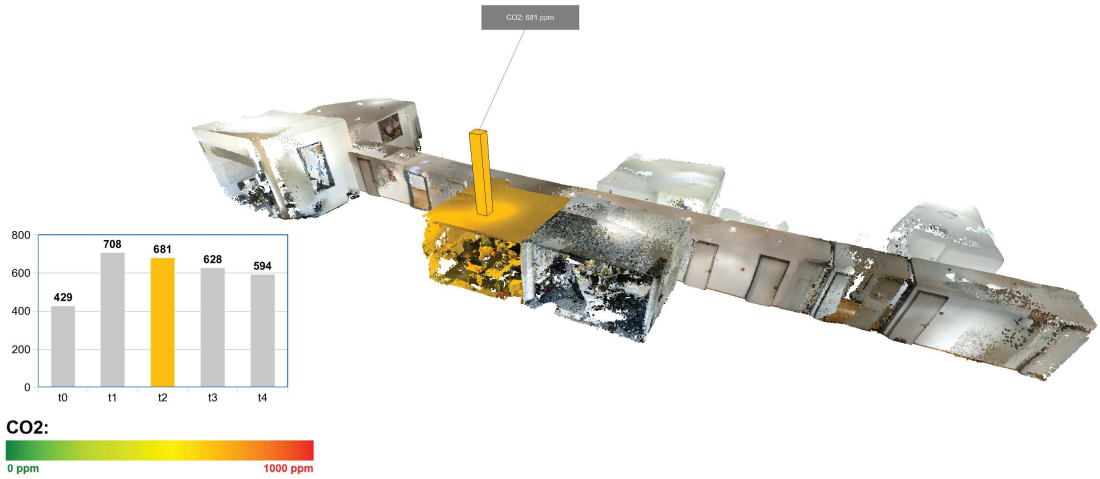
Figure 8.4: *Visualization of room temperature captured for two time points.*



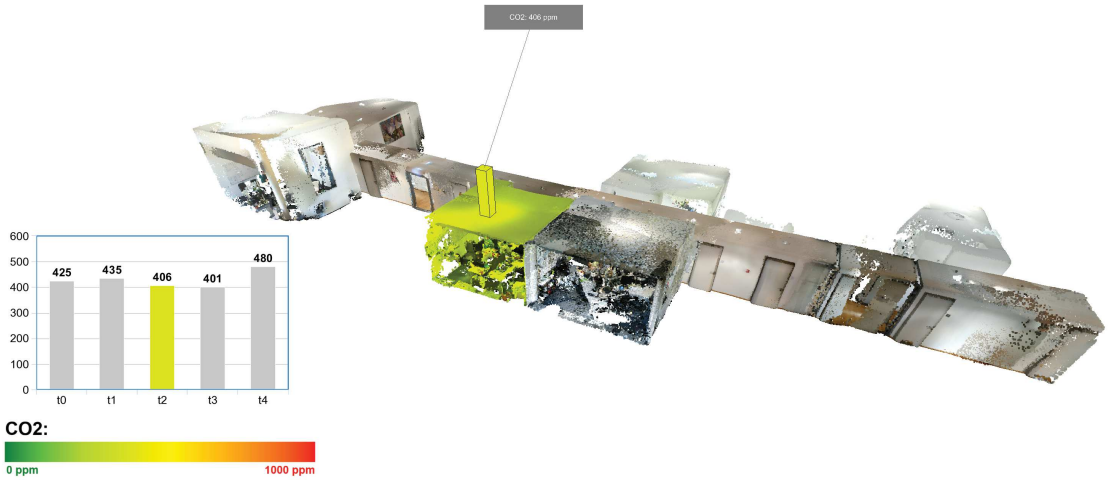
(a) t_0: *38%.*

(b) t_1: *22%*

Figure 8.5: *Visualization of room humidity captured for two time points.*

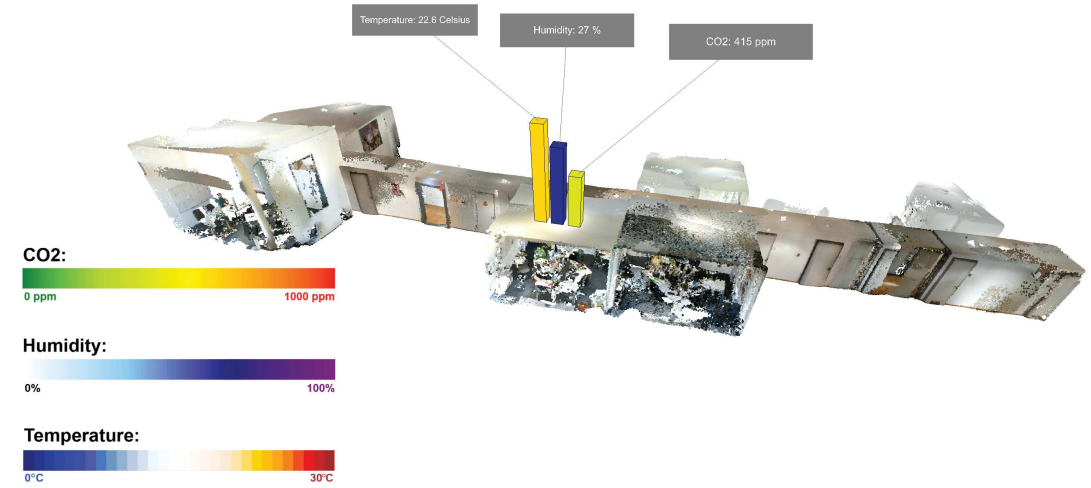


(a) t_0: *681 ppm.*

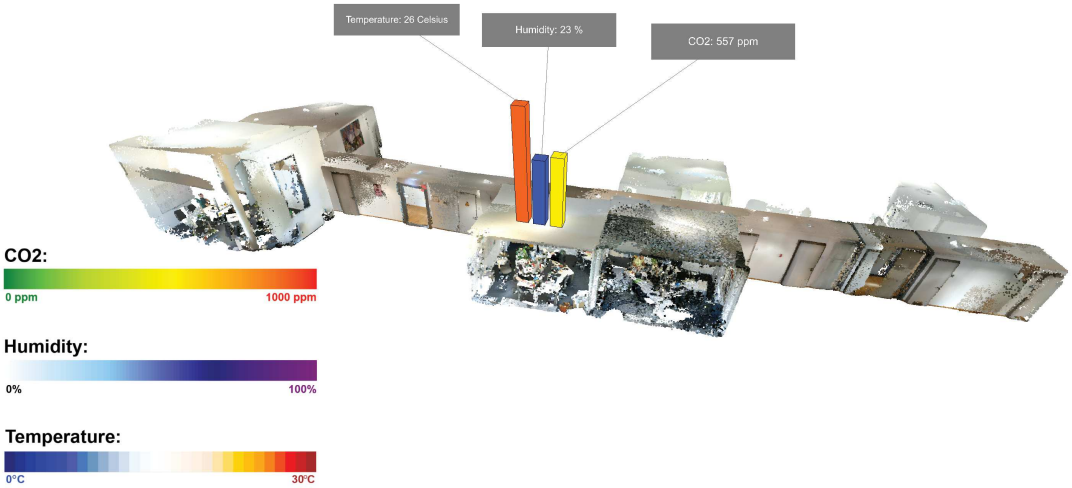


(b) t_1: *406 ppm.*

Figure 8.6: *Visualization of room carbon dioxide captured for two time points.*



(a) t_0: 22.6° *Celsius, 27% and 415 ppm*



(b) t_1: 26° *Celsius, 23% and 557 ppm*

Figure 8.7: *Visualization of combined room temperature, humidity and carbon dioxide captured for two time points.*

Chapter 9

Discussion

This section presents and discusses the answers that were formed to address the proposed research question. The associated research contributions are also discussed.

9.1 Research Outcomes

Fundamental Research Question The presented original research and contributions form the answer to the fundamental research question defined in Chpt. 1.2: *"How can digital data representing the built environment be semantically enriched and combined in order to enable digital representations of such environments within a service-oriented paradigm, and based on the concept of a Digital Twin (DT)?"*

The cumulative findings based on the the descriptions of approaches used as well as the evaluations of the results presented in each chapter case study, allow for the formation of an answer to this question.

The Answer to the Fundamental Research Question The answer to the proposed fundamental research question can thus be formulated in the following statement:

Data related to the digital representation of the built environment, primarily based on as-is point cloud representations, can be semantically enriched and combined with existing digital documentation, and further used to create an up-to-date cyberphyscial representation based on the concept of a DT. The feasibility of the presented approach is proven by implementing key software components and services within a SOS paradigm, which enables decoupling between hardware and software requirements for processing of digital data related to O&M applications within the realms of FM and Real Estate 4.0. Such software components and services could then be implemented in a future DT platform, and be used by stakeholders in order to gain potentially useful insights into the current and predicted state of a building.

This answer then leads onto the formulation of the answers to the four additional key research questions initially proposed in Chpt. 1.2, which form the foundations of the main research question. An attempt is made to answer each one, based on the conclusions of each associated chapter and case study findings.

Research Question 1 *How can indoor point cloud representations be processed and semantically enriched in order to make them useful as base-data for further FM-related analysis and applications?*

From the presented research findings of case studies in Chpt. 4-7, it can be concluded that point cloud representations of indoor environments play a fundamental role in the capture of the current *as-is* physical state of such environments. While raw point clouds do not contain any semantics by default, through various processing steps they can be enriched with semantics in order to make them suitable for further use in reconstruction, visualization and decision making.

These processing steps also include the generation of additional features for point clouds that otherwise contain only spatial and color information. Such features (e.g., normal vectors), can then be used by segmentation algorithms to spatially partition the point cloud into homogeneous regions. The point clusters that form these segmented regions can further be used as base data for generation of higher-level geometric representations e.g., reconstructed triangulated or voxelized geometry, or B-reps used by the IFC file format. The use of various geometry reconstruction methods allow point clouds to be used for generation of approximate boundary representations, including use for 2D and 3D floorplan approximations.

Furthermore, with the use of supervised machine-learning methods, point clouds can be semantically segmented with the use of deep-learning. The use of CNNs with either multiview or object-based classification approaches allows for the generation and injection of semantic labels into point clusters. This allows for such point clusters to be visualized and processed with additional context provided by the added semantics.

Research Question 2 *How can as-is point cloud be compared against as-built and as-designed BIM geometry in order to highlight any spatial deviations?*

The use of as-is point cloud representations allows for comparison with existing as-designed or as-built BIM geometry. This comparison is focused on evaluation of spatial deviations between the two models, providing insight into the current physical dimensions of the as-is point cloud model versus the intended spatial constraints and dimensions found in the as-designed or as-built BIM model. Specifically, the implications of spatial deviation analysis for O&M procedures within the realm of FM is that it can be used to highlight spatial differences between the current state of the physical environment and its original design.

The comparison between the as-is point cloud and an as-designed or as-built BIM is enabled by aligning the two models in the same coordinate space, and approximating the distances between each of the points in the as-is model to the nearest geometric element from the as-designed or as-built BIM model. The geometry used for comparison that is extracted from the BIM model can either be based on projections of simplified boundary evaluations (e.g., planar surfaces or bounding boxes), or triangulated meshes with spatial partitioning schemes, or based on voxelized representations. These methods are discussed and evaluated in Chpt. 6.

Research Question 3 *How can indoor sensor data be processed and visualized alongside an indoor point cloud?*

The use of visual analytics principals can be applied to spatio-temporal data associated with the built environment. Particularly, the use of visual analytics for processing, transforming, mapping and visualizing indoor sensor data associated with occupancy comfort and building performance can provide useful insights for FM stakeholders. With the approach described in Chpt. 8, captured sensor data can be processed and the visualization results can be combined with a point cloud representation in order to provide enhanced spatial context.

The use of visualization idioms allows the processed sensor data to be mapped to various textual, geometric and thematic color representations, and to be presented alongside the point cloud model of the associated indoor environment for enhanced spatial context. In turn the analysis and visualization of sensor data has the potential to benefit FM stakeholders by allowing them to better address the needs of occupants as well as to monitor and forecast energy usage and associated sustainability indicators.

Research Question 4 *How can data related to the the digital representation of indoor environments be accessed, processed, analyzed and visualized within a service-oriented paradigm?*

The implementation of the key processing components and services, using the prototypical SOS variants, enables the decoupling of hardware and software requirements. This allows the software implementation to be potentially scaled to enterprise-level systems – using a SOA-based system design, and allows for the flexible implementation and maintenance of key processing components. Since a DT is based on processing data from multiple sources, an SOS enables a centralized system to capture and process data from different sources, and to stream results of selected processes to different client configurations (with possible integration with DBMSs).

Furthermore, due to the requirement of having to deal with large amounts of data (e.g., point clouds, sensor data, BIMs, CAD, etc.), the use of a SOS allows for out-of-core processing and streaming of results without the explicit requirement of having to use a high-end workstations for client configurations.

The proposed SOA is described in Chpt. 3, and prototypical SOS implementations are used for evaluating key software components in related chapter case studies, and are based on the DT paradigm. The SOS software components (e.g., reconstruction, classification, etc.), are implemented and tested for each of the main processing tasks concerned with analysis and visualization of data in the presented case studies.

9.2 Discussion of Research Contributions

An applied research methodology was used to form the conclusions based on case studies. In turn, the presented case studies were used to provide data for the generation of the results from testing of key software components, which were implemented for processing tasks that were determined as being core requirements for proving the feasibility of the presented approach. The original research contributions form the foundation for future development of a DTs, aimed at FM and Real Estate 4.0 use.

Point Cloud Representations The key data source that has been the focus of the implemented and evaluated methods for indoor representations for this research are point clouds. A particular focus was placed on the use of PCCT-enabled commodity mobile devices for capturing indoor point clouds, and the subsequent processing, reconstruction and semantic enrichment methods applied to them – in order to make them useful for FM applications. While the focus has been using lower-fidelity point clouds, the presented approaches and methods can be applied to more denser and larger point cloud datasets.

Key SOS Software Components The presented SOS software components are primarily focused on the processing, semantic enrichment, reconstruction and visualization of indoor point clouds for O&M applications within the scope of FM. The approaches used for semantic enrichment of the captured point clouds rely heavily on machine and deep-learning methods, with a particular focus on the use of CNNs for supervised learning applications. While the presented methods for semantic enrichment leave a lot of room for improvement in terms of classification accuracy, with increased access to training data and computational resources, the implemented CNN models can further be retrained and new versions of the CNN models can be used to increase the classification accuracy.

Web3D-based Visualization The use of Web3D-based visualization is deemed as suitable for processing and representation of visualization results on a variety of client configurations. Modern web-based 3D graphics APIs such as WebGL and associated frameworks allow for computation of complex and real-time visual outputs, and enable rendering of point clouds with associated data on modern web browsers. The presented visualization methods did not explicitly focus on out-of-core rendering of point clouds, but rather on how semantically-enriched point clouds can be further be combined with other data sources for visualization-driven decision making tasks (e.g., analysis and visualization of sensor data for decision making).

Point Cloud Reconstruction The reconstruction of semantically enriched point clouds is another feature of the original research contributions, and methods for generation of approximate higher-level geometric representations were presented and evaluated. The reconstruction of point clouds to highly accurate models, as offered by manual reconstruction using professional CAD-based software, is not yet feasible. However, the use of automated methods for approximation of geometric reconstructions allows for the at least partial alleviation of the associated time and cost requirements – based on domain expertise and labor investment typically involved with manual generation of

such data [265]. Additional methods for comparison of as-is point clouds and existing geometry were also presented, and such methods based on spatial deviation allow for further beneficial use in FM decision making.

Sensor Data Analysis The processing and visualization of spatio-temporal data, particularly the combined use of sensor data analytics and visualization of point clouds, demonstrates the feasibility of using Web3D-based technologies for visualization of such data. The integration of visual analytics within a prototypical SOS implementation allows for decoupling of hardware and software requirements, while enabling the processing and analysis of multiple data sources within one centralized system.

Case Study Evaluations All of the presented case studies made of use of real-world data, and in most cases the data sets, particularly indoor point clouds, were captured specifically for the required evaluations. What can be noted is that point clouds can be seen as versatile data for representation of the physical built environment, and that through further processing and semantic enrichment, they can provide useful insights. This is also the case regardless of the source of the point clouds, as even with point clouds that are captured using commodity mobile devices, in most common cases enough information could be extracted from them to be used for further semantic enrichment operations within the realm of FM and Real Estate 4.0.

Additionally, the case studies are presented with the aim of demonstrating the potential benefits of using a service-oriented approach for DT-driven representations of indoor environments. These case studies demonstrate the feasibility of the overall concept of DTs and their use within an SOS paradigm. This includes specific approaches for processing, analyzing and visualizing indoor environments with the use of point clouds, sensor and floorplan data.

The implemented software components, while prototypical, are able to handle the processing requirements of structured and spatio-temporal data. While the size and the complexity of the data used by DTs are expected to increase exponentially in the near future, the implemented technology based on the presented case studies, would only need to be changed or updated to reflect the bigger data requirements.

Chapter 10

Conclusions and Outlook

10.1 Conclusions Summary

The research findings demonstrate the feasibility of key software components and services for a DT representation of indoor environments, implemented within a service-oriented paradigm, and aimed at enhancing decision making related to O&M procedures within the scope of the post-construction life cycle stages of typical office buildings. These key software components and services include point cloud processing, semantic enrichment, data analytics, visualization as well as reconstruction to higher-level geometric representations that make use of semantics generated through applications of machine and deep-learning methods.

The findings also conclude that the digital representation of built environments, based primarily on semantically enriched point clouds, can be used to create an up-to-date digital versions for generating useful insights. Such insights, based on the computed visual and analytical results, can be used to assess past and current states of a built environment object (e.g., a building), with the possibility to forecast future states.

Furthermore, the presented conceptual SOA, and the variations of the prototypical SOS implementations, provide a suitable foundation for further integration and expansion within FM-related ICT infrastructure.

10.2 Summary of Research Contributions

The following research contributions demonstrate the feasibility of a DT implementation for FM and Real Estate 4.0 use – focusing on the current life cycle assessment of indoor environments of modern buildings:

1. Design of a conceptual SOA for a FM-oriented DT, where key processing components were implemented and tested as prototypical SOSs related to presented case studies.

2. Research, implementation and evaluation of methods for semantic enrichment of indoor point clouds based primarily on machine and deep-learning.

3. Research, implementation and evaluation of methods for spatial deviation analysis between as-is and as-designed and/or as-built BIM geometry.

4. Research, implementation and evaluation of methods for higher-level geometric reconstruction of semantically-enriched point clouds, with the ability to generate both approximate 2D and 3D floorplans, and IFC model representations.

5. Research, implementation and evaluation of methods for processing and visualization of sensor data associated with indoor point clouds.

6. Methods for Web3D-based visualization, interactive annotation and inspection of semantically-enriched point cloud representations of indoor environments within a service-oriented paradigm.

10.3 Future Work and Outlook

The obvious future work would be the development and testing of an actual complete DT software solution. However, the development, deployment and maintenance of such solutions, especially at enterprise levels, requires a vast amount of further research and development. It will be a few years into the future until DT software solutions become mainstream (and have the same wide adaptation as BIM), and are used commonly for management of smart buildings and cities.

However, prior to this, the following research questions and themes could possibly help form solutions to tackling such a large and complex problem, from the perspectives of software engineering and academic research. The proposed future research can be summarized in five possible key questions :

1. The first question for future research is: *What kind of a predictive maintenance model could be used, if it is based on a DT representation of a building?.*

   Since one of the key features of a DT that makes use of machine and deep-learning capabilities are to analyze current and historical data, the possibility to form predictions and forecast future operational states of a building should be included. This could make use of methods e.g., Bayesian probability and Multi-Criteria Decision Analysis (MCDA) [75].

2. The second question is: *How can an ontology-driven approach be used for better FM decision making within a DT paradigm?.*

   Even when point clouds are semantically enriched, they seldom contribute to the ontology of a building. Thus they are only used for single-use decision making cases without contributing to any sort of Knowledge Base (KB) for future evaluation. An ontology needs to be formed that relates to the building's digital representation as well as the associated O&M processes. There is a paucity for an ontology-driven analytics approach, where users can simply query such a system in order for it to generate, associate and present possible answers for FM decision making tasks.

In turn, the knowledge of such a "smart" system, based on a DT paradigm, would be expended when performing any subsequent tasks (e.g., recommended selection of optimal algorithms and associated parameters when processing new point cloud data). An initial conceptual approach to solving this question is proposed by Stojanovic *et al.* [235].

3. The third question is: *How can changes, comments and decisions made by stakeholders, using a digital and semantically-rich representation of a built environment (e.g., point clouds, BIMs, etc.), be immutably recorded in the form of digital documentation for future use?.*

   This digital documentation can then be used for further analysis, review of decisions or even as proof of contractual obligations with third parties. Recording of changes and annotations in an immutable manner requires the use of a specific data structure (e.g., Merkle tree [165]), which can ensure immutability between current and previously recorded data segments. The use of a private distributed ledger, based on *Blockchain Technology* (BT) principals, allows for a such a data structure to be used for immutable recording of *transaction* data [151].

4. The fourth question is: *How beneficial, informative and practical, is a DT for routine FM stakeholder engagement and decision making?.*

   The presented research has only focused on the *feasibility* of key DT software components and services, based on their implementation and evaluation of quantitative case study results. For future work it would be of interest to generate and evaluate qualitative results based on FM domain expert user feedback, specifically concerning the use of the investigated software components for O&M decision making. Such an approach could be based on testing of complete and specific DT features for different scenarios (Fig. 10.1), with selected stakeholder groups [28].

5. The fifth question is: *How can the presented approach be expanded for more general use, not just for indoor building environments, but for core built infrastructure (e.g., roads and bridges) as well as city-scale analysis and visualization?.*

   Research by Isailović *et al.* [116] and Döllner [60] describes the use of point cloud-based approaches for semantic enrichment, visualization and analysis of built infrastructure. Therefore an approach is needed to scale the use of DT systems to meet the data processing demands of such large scale and complex entities. It can be expected these approaches will be developed further, and eventually evolve into complex SOSs capable of autonomous and predictive analysis and decision making.

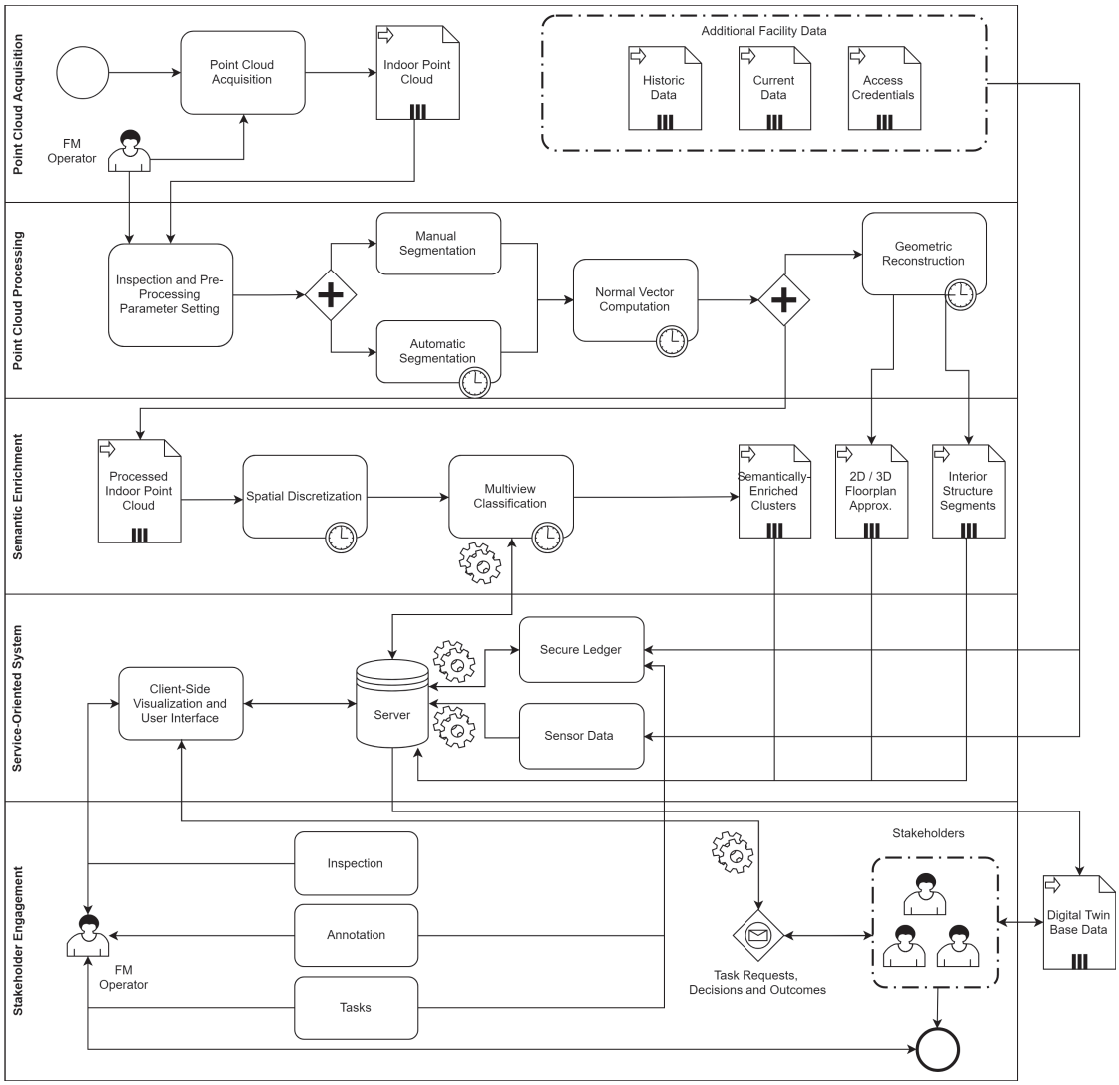


Figure 10.1: *An example conceptual approach for future work, focusing on semantic enrichment and use of indoor point clouds and associated data for FM decision making, and as base data for DT representations.*

Outlook The overall goal of a DT for FM and Real Estate 4.0 is to unify all of the processes and decision making tasks through ought the entire life cycle of a building. While the implementation of such DTs are still in infancy with the current state of technological advances, they will eventually become a standard for data-driven analysis, forecasting, and decision making related to buildings and larger built environment infrastructure. The main driving factor behind this is the unification of *Big Data* analytics [209], and existing data (e.g., BIM data).

The paradigm of Big Data is to enable the analysis of data with increasing *velocity*, *volume* and *variety*. It should also be taken into account that the growth of data for Real Estate 4.0 applications follows the principals of Big Data, and in most use cases it can be predicted that the size of the data used by a DT would increase exponentially. Thus adequate ICT infrastructure, especially storage and suitable DBMSs are required for access and processing of required data, and should be considered when planning the implementation of DTs based on the approaches presented in this research.

Apart from point cloud representations of the built environment, the use of higher-level representations (e.g., IFC representations, voxelized meshes, approximated floor-plans), can provide further useful insight into the state of the built environment, as they can be used for further analysis, decision making as well as asset management [182]. It can also be expected that the use of the IndoorGML [1] specification will need to be supported by future DT implementations [179], alongside standard IFC, CAD and other FM-related digital data. This will help address the need of FM stakeholders to have access to semantically richer and up-to-date representations of indoor environments. Furthermore, the unification of all related data, entities, relationships and an ontology into a *common model*, could be used by various component implementations and versions of DTs – allowing for portable DT systems and platforms that are applicable to any existing or new building.

The use of visualization and analytics for Big Data related to building management and operations would also allow FM stakeholders to gain insight and better understanding into the O&M requirements of a building – based on captured and analyzed data sources. This in turn creates greater *domain intelligence* within FM, provides better return of investment, and allows FM stakeholders to make faster and more accurate decisions.

Furthermore, new generation consumer electronic and robotic devices (e.g., UAV drones for as-is building inspection [84], etc.), will enable the use of once expensive technology for capturing of indoor point clouds to be affordable and practical for routine FM applications (e.g., use of an integrated LiDAR sensor in modern mobile phones [221], using new generation Vertical-Cavity Surface-Emitting Laser (VCSEL) amplifiers [144]). The affordability of new generation RFID-based sensors and RFID tags [90] will also allow them to be commonly used in environmental sensing and IoT applications (e.g., room ambient intelligence). Finally, the use of AI and ML paradigms will shift towards the use of models whose processes and results can be better understood by stakeholders – by utilizing the concepts and methods of Explainable AI (XAI) [2].

[1] IndoorGML Specification: `http://www.indoorgml.net/`

"Let the future tell the truth, and evaluate each one according to his work and accomplishments. The present is theirs; the future, for which I have really worked, is mine." – Nikola Tesla.

References

[1] Martın Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, *et al.* "TensorFlow: A System for Large-Scale Machine Learning." In: *OSDI*. Vol. 16. 2016, pp. 265–283.

[2] Amina Adadi and Mohammed Berrada. "Peeking inside the black-box: A survey on Explainable Artificial Intelligence (XAI)". In: *IEEE Access* 6 (2018), pp. 52138–52160.

[3] Sarthak Agarwal and KS Rajan. "Analyzing the performance of NoSQL vs. SQL databases for Spatial and Aggregate queries". In: *Free and Open Source Software for Geospatial (FOSS4G) Conference Proceedings*. Vol. 17. 1. 2017, p. 4.

[4] Charu C Aggarwal *et al. Neural networks and deep learning*. Springer, 2018.

[5] Tomas Akenine-Möller, Eric Haines, Naty Hoffman, Angelo Pesce, Michał Iwanicki, and Sébastien Hillaire. *Real-Time Rendering 4th Edition*. A K Peters/CRC Press, 2018, p. 1200. ISBN: 978-1-13862-700-0.

[6] Vito Albino, Umberto Berardi, and Rosa Maria Dangelico. "Smart cities: Definitions, dimensions, performance, and initiatives". In: *Journal of urban technology* 22.1 (2015), pp. 3–21.

[7] Harith Aljumaily, Debra F Laefer, and Dolores Cuadra. "Urban point cloud mining based on density clustering and MapReduce". In: *Journal of Computing in Civil Engineering* 31.5 (2017).

[8] Engin Burak Anil, Pingbo Tang, Burcu Akinci, and Daniel Huber. "Deviation analysis method for the assessment of the quality of the as-is Building Information Models generated from point cloud data". In: *Automation in Construction* 35 (2013), pp. 507–516.

[9] Hasan Asy'ari Arief, Mansur Arief, Guilin Zhang, Zuxin Liu, Manoj Bhat, Ulf Geir Indahl, Håvard Tveite, and Ding Zhao. "SAnE: Smart Annotation and Evaluation Tools for Point Cloud Data". In: *IEEE Access* 8 (2020), pp. 131848–131858.

[10] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. "3d semantic parsing of large-scale indoor spaces". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 1534–1543.

[11] Muhammad Arslan, Zainab Riaz, and Saba Munawar. “Building information modeling (BIM) enabled facilities management using hadoop architecture”. In: *2017 Portland International Conference on Management of Engineering and Technology (PICMET)*. IEEE. 2017, pp. 1–7.

[12] Steven Arthur, Haijiang Li, and Robert Lark. “The Emulation and Simulation of Internet of Things Devices for Building Information Modelling (BIM)”. In: *Workshop of the European Group for Intelligent Computing in Engineering*. Springer. 2018, pp. 325–338.

[13] K Babacan, L Chen, and G Sohn. “Semantic Segmentation of Indoor Point Clouds Using Convolutional Neural Network.” In: *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences* 4 (2017).

[14] Daniel Baeder, Eric Christensen, Anhvinh Doanvo, Sara Lindsay Grove, Andrew Han, Ben FM Intoy, Steven Hardy, Zachary Humayun, Melissa Kain, Kevin Liberman, *et al. Creating Digital Twins of Cities with FutureScape™*. Deloitte Consulting LLP, 2019.

[15] Dana H Ballard. “Generalizing the Hough transform to detect arbitrary shapes”. In: *Pattern recognition* 13.2 (1981), pp. 111–122.

[16] Oluleke Bamodu, Liang Xia, and Llewellyn Tang. “An indoor environment monitoring system using low-cost sensor network”. In: *Energy Procedia* 141 (2017), pp. 660–666.

[17] C Bradford Barber, David P Dobkin, David P Dobkin, and Hannu Huhdanpaa. “The quickhull algorithm for convex hulls”. In: *ACM Transactions on Mathematical Software (TOMS)* 22.4 (1996), pp. 469–483.

[18] Rafael Barea, Carlos Pérez, Luis M Bergasa, Elena López-Guillén, Eduardo Romera, Eduardo Molinos, Manuel Ocana, and Joaqun López. “Vehicle detection and localization using 3d lidar point cloud and image semantic segmentation”. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 3481–3486.

[19] Maarten Bassier, George Hadjidemetriou, Maarten Vergauwen, Nathalie Van Roy, and Els Verstrynge. “Implementation of Scan-to-BIM and FEM for the documentation and analysis of heritage timber roof structures”. In: *Euro-mediterranean conference*. Springer. 2016, pp. 79–90.

[20] Michael Batty, Kay W Axhausen, Fosca Giannotti, Alexei Pozdnoukhov, Armando Bazzani, Monica Wachowicz, Georgios Ouzounis, and Yuval Portugali. “Smart cities of the future”. In: *The European Physical Journal Special Topics* 214.1 (2012), pp. 481–518.

[21] B Beckers, T Pico, and S Jimenez. “A robust smoothed voxel representation for the generation of finite element models for computational urban physics”. In: (2016), pp. 249–259.

[22] Ben Bellekens, Vincent Spruyt, Rafael Berkvens, and Maarten Weyn. “A survey of rigid 3D pointcloud registration algorithms”. In: *AMBIENT 2014: the Fourth International Conference on Ambient Computing, Applications, Services and Technologies, August 24-28, 2014, Rome, Italy*. 2014, pp. 8–13.

[23] Jon Louis Bentley. “Multidimensional binary search trees used for associative searching”. In: *Communications of the ACM* 18.9 (1975), pp. 509–517.

[24] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. “The ball-pivoting algorithm for surface reconstruction”. In: *IEEE transactions on visualization and computer graphics* 5.4 (1999), pp. 349–359.

[25] Paul J. Besl and Ramesh C Jain. “Segmentation through variable-order surface fitting”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 10.2 (1988), pp. 167–192.

[26] Vighnesh Birodkar and Damodar Reddy Edla. “Enhanced K-Means Clustering Algorithm using A Heuristic Approach”. In: *Journal of Information and Computing Science* 9.4 (2014), pp. 277–284.

[27] ID Bishop, W-S Ye, and C Karadaglis. “Experiential approaches to perception response in virtual worlds”. In: *Landscape and Urban Planning* 54.1-4 (2001), pp. 117–125.

[28] David J Blackwood, Daniel J Gilmour, John P Isaacs, Thomas Kurka, and Ruth E Falconer. “Sustainable urban development in practice: the SAVE concept”. In: *Environment and Planning B: Planning and Design* 41.5 (2014), pp. 885–906.

[29] Tanya Bloch and Rafael Sacks. “Comparing machine learning and rule-based inferencing for semantic enrichment of BIM models”. In: *Automation in Construction* 91 (2018), pp. 256–272.

[30] Răzvan Gabriel Boboc, Florin Gırbacia, Cristian Cezar Postelnicu, and Teodora Gırbacia. “Evaluation of Using Mobile Devices for 3D Reconstruction of Cultural Heritage Artifacts”. In: *International Conference on VR Technologies in Cultural Heritage*. Springer. 2018, pp. 46–59.

[31] Mathias Bonduel, Maarten Bassier, Maarten Vergauwen, Pieter Pauwels, and Ralf Klein. “Scan-to-BIM Output validation: Towards a standardized geometric quality assessment of Building Information Models based on point clouds”. In: *5th International Workshop LowCost 3D-Sensors, Algorithms, Applications*. Vol. 42. Copernicus GmbH. 2017, pp. 45–52.

[32] Michael Borth, Jacques Verriet, and Gerrit Muller. “Digital twin strategies for SoS: 4 challenges and 4 architecture setups for digital twins of SoS”. In: *14th Annual Conference System of Systems Engineering, SoSE 2019*. Institute of Electrical and Electronics Engineers Inc. 2019, pp. 164–169.

[33] F Bosche and Carl T Haas. “Automated retrieval of 3D CAD model objects in construction range images”. In: *Automation in Construction* 17.4 (2008), pp. 499–512.

[34] Frédéric Bosché and Emeline Guenet. "Automating surface flatness control using terrestrial laser scanning and building information models". In: *Automation in construction* 44 (2014), pp. 212–226.

[35] Richard K Brail and Richard E Klosterman. *Planning support systems: Integrating geographic information systems, models, and visualization tools*. ESRI, Inc., 2001.

[36] buildingSMART International. *IFC4 Addenum 2 Specification*. Available at `https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/`, last accessed: 13/09/2019. 2016.

[37] Ricardo Cabello *et al. Three. js*. Available at `https://github.com/mrdoob/three.js/`, last accessed: 20/05/2020. 2020.

[38] Dace A Campbell. "Building information modeling: the Web3D application for AEC". In: *Proceedings of the twelfth international conference on 3D web technology*. ACM. 2007, pp. 173–176.

[39] Pascual Castelló, Mateu Sbert, Miguel Chover, and Miquel Feixas. "Techniques for computing viewpoint entropy of a 3d scene". In: *International Conference on Computational Science*. Springer. 2006, pp. 263–270.

[40] Jim H Chandler, Simon Buckley, MB Carpenter, CM Keane, Geoscience Handbook, *et al.* "Structure from motion (SFM) photogrammetry vs terrestrial laser scanning". In: *Geoscience handbook* (2016).

[41] Kai-Ming Chang, Ren-Jye Dzeng, and Yi-Ju Wu. "An Automated IoT Visualization BIM Platform for Decision Support in Facilities Management". In: *Applied Sciences* 8.7 (2018), p. 1086.

[42] Erzhuo Che, Jaehoon Jung, and Michael J Olsen. "Object recognition, segmentation, and classification of mobile laser scanning point clouds: A state of the art review". In: *Sensors* 19.4 (2019), p. 810.

[43] Kang Chen, Yu-Kun Lai, and Shi-Min Hu. "3D indoor scene modeling from RGB-D data: a survey". In: *Computational Visual Media* 1.4 (2015), pp. 267–278.

[44] Ke Chen, Weisheng Lu, Yi Peng, Steve Rowlinson, and George Q Huang. "Bridging BIM and building: From a literature review to an integrated conceptual framework". In: *International journal of project management* 33.6 (2015), pp. 1405–1416.

[45] Songlin Chen, Liangliang Nan, Renbo Xia, Jibin Zhao, and Peter Wonka. "PLADE: A Plane-Based Descriptor for Point Cloud Registration With Small Overlap". In: *IEEE Transactions on Geoscience and Remote Sensing* 58.4 (2019), pp. 2530–2540.

[46] Yang Chen and Gérard Medioni. "Object modelling by registration of multiple range images". In: *Image and vision computing* 10.3 (1992), pp. 145–155.

[47] Filiberto Chiabrando, Roberto Chiabrando, Dario Piatti, and Fulvio Rinaudo. "Sensors for 3D imaging: Metric evaluation and calibration of a CCD/CMOS time-of-flight camera". In: *Sensors* 9.12 (2009), pp. 10080–10096.

[48] Hung-Yueh Chiang, Yen-Liang Lin, Yueh-Cheng Liu, and Winston H Hsu. "A unified point-based framework for 3D segmentation". In: *2019 International Conference on 3D Vision (3DV)*. IEEE. 2019, pp. 155–163.

[49] Szu-cheng Chien, Tzu-chun Chuang, Huei-Sheng Yu, Yi Han, Boon Hee Soong, and King Jet Tseng. "Implementation of cloud BIM-based platform towards high-performance building services". In: *Procedia environmental sciences* 38 (2017), pp. 436–444.

[50] Woncheol Choi and Thomas R Kurfess. "Dimensional measurement data analysis, part 1: a zone fitting algorithm". In: *Journal of manufacturing science and engineering* 121.2 (1999), pp. 238–245.

[51] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. "Meshlab: an open-source mesh processing tool." In: *Eurographics Italian chapter conference*. Vol. 2008. 2008, pp. 129–136.

[52] M Clauss, J Götze, E Müller, and CA Schumann. "Real-time Data Access through Semantic Technologies and Augmented Reality in Facility Management Processes". In: *Proc. Int. Conf. on Innovative Technologies, IN-TECH*. 2014, pp. 45–48.

[53] Anthony Cormier, Sylvain Robert, Pierrick Roger, Louis Stephan, and Etienne Wurtz. "Towards a BIM-based service oriented platform: application to building energy performance simulation". In: *Proceedings of the 12th conference of international building performance simulation association, Sydney, Australia*. Vol. 1416. 2011.

[54] Abdoulaye Abou Diakité and Sisi Zlatanova. "Valid space description in BIM for 3D indoor navigation". In: *International Journal of 3-D Information Modeling (IJ3DIM)* 5.3 (2016), pp. 1–17.

[55] 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 (2015), pp. 32–45.

[56] Sören Discher, Rico Richter, and Jürgen Döllner. "A scalable webGL-based approach for visualizing massive 3D point clouds using semantics-dependent rendering techniques". In: *Proceedings of the 23rd International ACM Conference on 3D Web Technology*. ACM. 2018, p. 19.

[57] Sören Discher, Rico Richter, and Jürgen Döllner. "Concepts and Techniques for Web-based Visualization and Processing of Massive 3D Point Clouds with Semantics". In: *Graphical Models* (2019), p. 101036.

[58] Sören Discher, Rico Richter, Matthias Trapp, and Jürgen Döllner. "Service-Oriented Processing and Analysis of Massive Point Clouds in Geoinformation Management". In: *Service-Oriented Mapping*. Springer, 2019, pp. 43–61.

[59] Jozef Doboš, Carmen Fan, Pavol Knapo, and Charence Wong. “Applications of web3D technology in architecture, engineering and construction”. In: *Proceedings of the 23rd International ACM Conference on 3D Web Technology*. ACM. 2018, p. 30.

[60] Jürgen Döllner. “Geospatial Artificial Intelligence: Potentials of Machine Learning for 3D Point Clouds and Geospatial Digital Twins”. In: *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science* (2020), pp. 1–10.

[61] Jürgen Döllner and Benjamin Hagedorn. “Integrating urban GIS, CAD, and BIM data by servicebased virtual 3D city models”. In: (2007).

[62] Jürgen Döllner, Benjamin Hagedorn, and Jan Klimke. “Server-based rendering of large 3D scenes for mobile devices using G-buffer cube maps”. In: *Proceedings of the 17th International Conference on 3D Web Technology*. ACM. 2012, pp. 97–100.

[63] Jürgen Döllner, Markus Jobst, and Peter Schmitz. *Service-Oriented Mapping: Changing Paradigm in Map Production and Geoinformation Management*. Springer, 2018.

[64] David H Douglas and Thomas K Peucker. “Algorithms for the reduction of the number of points required to represent a digitized line or its caricature”. In: *Cartographica: The International Journal for Geographic Information and Geovisualization* 10.2 (1973), pp. 112–122.

[65] Robert Eadie, Mike Browne, Henry Odeyinka, Clare McKeown, and Sean McNiff. “BIM implementation throughout the UK construction project lifecycle: An analysis”. In: *Automation in construction* 36 (2013), pp. 145–151.

[66] E William East. “Construction operations building information exchange (Cobie): Requirements definition and pilot implementation standard”. In: (2007).

[67] Chuck Eastman, Paul Teicholz, Rafael Sacks, and Kathleen Liston. *BIM handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors*. John Wiley & Sons, 2011.

[68] David Eberly. *Distance between point and triangle in 3D*. Available at `http://www.magic-software.com/Documentation/pt3tri3.pdf`, last accessed: 01/11/2019. 1999.

[69] David Eberly. “Least squares fitting of data”. In: *Chapel Hill, NC: Magic Software* (2000).

[70] Herbert Edelsbrunner, David Kirkpatrick, and Raimund Seidel. “On the shape of a set of points in the plane”. In: *IEEE Transactions on information theory* 29.4 (1983), pp. 551–559.

[71] Peter Eisert. “Reconstruction of volumetric 3d models”. In: *3D Videocommunication: Algorithms, Concepts and Real-Time Systems in Human Centred Communication* (2005), p. 133.

[72] Mark Endrei, Jenny Ang, Ali Arsanjani, Sook Chua, Philippe Comte, Pål Krogdahl, Min Luo, and Tony Newling. *Patterns: service-oriented architecture and web services*. IBM Corporation, International Technical Support Organization, 2004.

[73] Juri Engel, Amir Semmo, Matthias Trapp, and Jürgen Döllner. "Evaluating the Perceptual Impact of Rendering Techniques on Thematic Color Mappings in 3D Virtual Environments." In: *Vision, Modeling and Visualization*. 2013, pp. 25–32.

[74] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, *et al.* "A density-based algorithm for discovering clusters in large spatial databases with noise." In: *Kdd*. Vol. 96. 34. 1996, pp. 226–231.

[75] Norman Fenton and Martin Neil. "Making decisions: using Bayesian nets and MCDA". In: *Knowledge-Based Systems* 14.7 (2001), pp. 307–325.

[76] Florian W Fichtner. *Semantic enrichment of a point cloud based on an octree for multi-storey pathfinding*. 2016.

[77] Roy T Fielding and Richard N Taylor. *Architectural styles and the design of network-based software architectures*. Vol. 7. University of California, Irvine Irvine, 2000.

[78] Raphael A. Finkel and Jon Louis Bentley. "Quad trees a data structure for retrieval on composite keys". In: *Acta informatica* 4.1 (1974), pp. 1–9.

[79] Martin A Fischler and Robert C Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[80] Ronald Aylmer Fisher, Frank Yates, *et al.* "Statistical tables for biological, agricultural and medical research." In: *Statistical tables for biological, agricultural and medical research.* Ed. 3. (1949).

[81] James D Foley, Foley Dan Van, Andries Van Dam, Steven K Feiner, John F Hughes, J Hughes, and Edward Angel. *Computer graphics: principles and practice*. Vol. 12110. Addison-Wesley Professional, 1996.

[82] W Randolph Franklin. *Pnpoly-point inclusion in polygon test*. Available at `http://www.ecse.rpi.edu/Homepages/wrf/Research/Short_Notes/pnpoly.html`, last accessed: 03/06/2020. 2006.

[83] Steffen Franz, Robert Irmler, and Uwe Rüppel. "Real-time collaborative reconstruction of digital building models with mobile devices". In: *Advanced Engineering Informatics* 38 (2018), pp. 569–580.

[84] Henk Freimuth and Markus König. "A framework for automated acquisition and processing of As-built data with autonomous unmanned aerial vehicles". In: *Sensors* 19.20 (2019), p. 4513.

[85] Mark Froehlich, Salman Azhar, and Matthew Vanture. "An investigation of google tango® tablet for low cost 3D scanning". In: *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*. Vol. 34. 2017, pp. 1–9.

[86] Te Gao, Semiha Ergan, Burcu Akinci, and James Garrett. “Evaluation of different features for matching point clouds to building information models”. In: *Journal of Computing in Civil Engineering* 30.1 (2014), p. 04014107.

[87] Tom Lloyd Garwood, Ben Richard Hughes, Dominic O’Connor, John K Calautit, Michael R Oates, and Thomas Hodgson. “A framework for producing gbXML building geometry from Point Clouds for accurate and efficient Building Energy Modelling”. In: *Applied energy* 224 (2018), pp. 527–537.

[88] Daniel Girardeau-Montaut. “Cloud compare—3d point cloud and mesh processing software”. In: *Open Source Project* (2015).

[89] Daniel Girardeau-Montaut, Michel Roux, Raphaël Marc, and Guillaume Thibault. “Change detection on points cloud data acquired with a ground laser scanner”. In: *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 36.part 3 (2005), W19.

[90] Edoardo Giusto, Filippo Gandino, Michele Luigi Greco, Michelangelo Grosso, Bartolomeo Montrucchio, and Salvatore Rinaudo. “An investigation on pervasive technologies for IoT-based thermal monitoring”. In: *Sensors* 19.3 (2019), p. 663.

[91] Steffen Goebbels and Regina Pohle-Fröhlich. “Beautification of city models based on mixed integer linear programming”. In: *Operations Research Proceedings 2018*. Springer, 2019, pp. 119–125.

[92] Frank Goetz, Bernd Eßmann, and Thorsten Hampel. “Collaboration by illustration: real-time visualization in Web3D”. In: *Proceedings of the eleventh international conference on 3D web technology*. ACM. 2006, pp. 47–56.

[93] Rhys Goldstein, Simon Breslav, and Azam Khan. “Towards voxel-based algorithms for building performance simulation”. In: *Proceedings of the IBPSA-Canada eSim Conference*. 2014, p. 14.

[94] Mani Golparvar-Fard, Jeffrey Bohn, Jochen Teizer, Silvio Savarese, and Feniosky Peña-Mora. “Evaluation of image-based modeling and laser scanning accuracy for emerging automated performance monitoring techniques”. In: *Automation in construction* 20.8 (2011), pp. 1143–1155.

[95] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. Vol. 1. MIT press Massachusetts, USA: 2017.

[96] Ned Greene. “Environment mapping and other applications of world projections”. In: *IEEE Computer Graphics and Applications* 6.11 (1986), pp. 21–29.

[97] Michael Grieves. “Digital twin: Manufacturing excellence through virtual factory replication”. In: *White paper* (2014), pp. 1–7.

[98] David Griffiths and Jan Boehm. “A review on deep learning techniques for 3D sensed data classification”. In: *Remote Sensing* 11.12 (2019), p. 1499.

[99] Eleonora Grilli, Fabio Menna, and Fabio Remondino. “A review of point clouds segmentation and classification algorithms”. In: *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 42 (2017), p. 339.

[100] BIM Industry Working Group *et al.* “A report for the government construction client group building information modelling (BIM) working party strategy paper”. In: *Communications. London, UK* (2011).

[101] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. “Internet of Things (IoT): A vision, architectural elements, and future directions”. In: *Future generation computer systems* 29.7 (2013), pp. 1645–1660.

[102] Robert B Haber and David A McNabb. “Visualization idioms: A conceptual model for scientific visualization systems”. In: *Visualization in scientific computing* 74 (1990), p. 93.

[103] Benjamin Hagedorn and Jürgen Döllner. “High-level web service for 3D building information visualization and analysis”. In: *Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems*. ACM. 2007, p. 8.

[104] Hamid Hajian and Burcin Becerik-Gerber. “Scan to BIM: factors affecting operational and computational errors and productivity loss”. In: *27th International Symposium on Automation and Robotics in Construction*. 2010, pp. 265–272.

[105] Thomas C Hales. “The Jordan curve theorem, formally and informally”. In: *The American Mathematical Monthly* 114.10 (2007), pp. 882–894.

[106] Felix G Hamza-Lup and Marcel Maghiar. “Web3D graphics enabled through sensor networks for cost-effective assessment and management of energy efficiency in buildings”. In: *Graphical Models* 88 (2016), pp. 66–74.

[107] Adrian Herwig and Philip Paar. “Game engines: tools for landscape visualization and planning”. In: *Trends in GIS and virtualization in environmental planning and design* 161 (2002), p. 172.

[108] Dieter Hildebrandt, Jan Klimke, Benjamin Hagedorn, and Jürgen Döllner. “Service-oriented interactive 3D visualization of massive 3D city models on thin clients”. In: *Proceedings of the 2nd International Conference on Computing for Geospatial Research & Applications*. ACM. 2011, p. 6.

[109] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. *Surface reconstruction from unorganized points*. Vol. 26. 2. ACM, 1992.

[110] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. “OctoMap: An efficient probabilistic 3D mapping framework based on octrees”. In: *Autonomous robots* 34.3 (2013), pp. 189–206.

[111] Jing Huang and Suya You. “Point cloud labeling using 3d convolutional neural network”. In: *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE. 2016, pp. 2670–2675.

[112] Satoshi Ikehata, Hang Yang, and Yasutaka Furukawa. “Structured indoor modeling”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1323–1331.

[113] Anastasia Ioannidou, Elisavet Chatzilari, Spiros Nikolopoulos, and Ioannis Kompatsiaris. “Deep learning advances in computer vision with 3d data: A survey”. In: *ACM Computing Surveys (CSUR)* 50.2 (2017), p. 20.

[114] John P Isaacs, Vladeta Stojanovic, Chris McCreadie, Ruth E Falconer, Daniel J Gilmour, and David J Blackwood. “The SAVE Concept: Sustainability Assessment & Enhancement Through Novel Visualisation”. In: (2012).

[115] Dušan Isailović, Rade Hajdin, and JC Matos. “Bridge quality control using Bayesian net”. In: *IABSE Symposium, Nantes 2018: Tomorrow’s Megastructures*. International Association for Bridge and Structural Engineering (IABSE). 2019.

[116] Dušan Isailović, Vladeta Stojanovic, Matthias Trapp, Rico Richter, Rade Hajdin, and Jürgen Döllner. “Bridge damage: Detection, IFC-based semantic enrichment and visualization”. In: *Automation in Construction* 112 (2020), p. 103088.

[117] Raymond Issa, I Flood, and W O’Brien. “Benefits of 3D and 4D models for facility managers and AEC service providers”. In: *4D CAD and Visualization in Construction*. CRC Press, 2003, pp. 9–40.

[118] Gómez J. *IRML Source Code and Database repository*. Available online `https://github.com/jvgomez/irml`, last accessed: 02/01/208. 2018.

[119] Achin Jain, Derek Nong, Truong X Nghiem, and Rahul Mangharam. “Digital Twins for Efficient Modeling and Control of Buildings: An Integrated Solution with SCADA Systems”. In: *2018 Building Performance Analysis Conference and SimBuild*. 2018, pp. 1–8.

[120] Wonkwi Jang, Changsoo Je, Yongduek Seo, and Sang Wook Lee. “Structured-light stereo: Comparative analysis and integration of structured-light and active stereo for measuring dynamic shape”. In: *Optics and Lasers in Engineering* 51.11 (2013), pp. 1255–1264.

[121] NIE Jian, Wei-sheng XU, Da-zhang CHENG, and You-ling YU. “Digital Twin-based Smart Building Management and Control Framework”. In: *DEStech Transactions on Computer Science and Engineering* icaic (2019).

[122] Ruinian Jiang, David V Jáuregui, and Kenneth R White. “Close-range photogrammetry applications in bridge measurement: Literature review”. In: *Measurement* 41.8 (2008), pp. 823–834.

[123] Jim Whittaker. *Facility Management Information Technologies*. Available at `https://www.wbdg.org/facilities-operations-maintenance/facility-management-information-technologies`, last accessed: 24/09/2020. 2016.

[124] Nicolai M Josuttis. *SOA in practice: the art of distributed system design*. O'Reilly Media, Inc., 2007.

[125] Sakdirat Kaewunruen, Panrawee Rungskunroch, and Joshua Welsh. "A digital-twin evaluation of net zero energy building for existing buildings". In: *Sustainability* 11.1 (2019), p. 159.

[126] Sakdirat Kaewunruen and Ningfang Xu. "Digital twin for sustainability evaluation of railway station buildings". In: *Frontiers in Built Environment* 4 (2018), p. 77.

[127] T Sri Kalyan, Puyan A Zadeh, Sheryl Staub-French, and Thomas M Froese. "Construction quality assessment using 3D as-built models generated with Project Tango". In: *Procedia Engineering* 145 (2016), pp. 1416–1423.

[128] Ryan S Kaminsky, Noah Snavely, Steven M Seitz, and Richard Szeliski. "Alignment of 3D point clouds to overhead images". In: *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE. 2009, pp. 63–70.

[129] Daniel Kazado, Miroslava Kavgic, and Rasit Eskicioglu. "Integrating Building Information Modeling (BIM) and sensor technology for Facility Management". In: *Journal of Information Technology in Construction (ITcon)* 24.23 (2019), pp. 440–458.

[130] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. "Poisson surface reconstruction". In: *Proceedings of the fourth Eurographics symposium on Geometry processing*. Vol. 7. 2006.

[131] Daniel A Keim, Florian Mansmann, Jörn Schneidewind, Jim Thomas, and Hartmut Ziegler. "Visual analytics: Scope and challenges". In: *Visual data mining*. Springer, 2008, pp. 76–90.

[132] Daniel Keim, Jörn Kohlhammer, Geoffrey Ellis, and Florian Mansmann. *Mastering the information age: solving problems with visual analytics*. 2010.

[133] Graham Kelly, Michael Serginson, Steve Lockley, Nashwan Dawood, and Mohamad Kassem. "BIM for facility management: a review and a case study investigating the value and challenges". In: *Proceedings of the 13th International Conference on Construction Applications of Virtual Reality*. 2013, pp. 30–31.

[134] Karen Kensek. "BIM guidelines inform facilities management databases: a case study over time". In: *Buildings* 5.3 (2015), pp. 899–916.

[135] Karen Kensek. "Integration of Environmental Sensors with BIM: case studies using Arduino, Dynamo, and the Revit API". In: 66.536 (2014).

[136] Muhammad Umar Khalid, Muhammad Khalid Bashir, and Darryl Newport. "Development of a building information modelling (BIM)-based real-time data integration system using a building management system (BMS)". In: *Building Information Modelling, Building Performance, Design and Smart Construction*. Springer, 2017, pp. 93–104.

[137] Azam Khan and Kasper Hornbæk. “Big data from the built environment”. In: *Proceedings of the 2nd international workshop on Research in the large*. ACM. 2011, pp. 29–32.

[138] David Inkyu Kim and Gaurav S Sukhatme. “Semantic labeling of 3d point clouds with object affordance for robot manipulation”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 5578–5584.

[139] Rob Kitchin. “The real-time city? Big data and smart urbanism”. In: *GeoJournal* 79.1 (2014), pp. 1–14.

[140] Jan Klimke. “Web-based provisioning and application of large-scale virtual 3D city models”. PhD thesis. Hasso Plattner Institute, Faculty of Digital Engineering, University of Potsdam, Germany, 2019.

[141] Roman Klokov and Victor Lempitsky. “Escape from cells: Deep kd-networks for the recognition of 3d point cloud models”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 863–872.

[142] Donald E Knuth. *The art of computer programming: Volume 3: Sorting and Searching*. Addison-Wesley Professional, 1998.

[143] Elmar de Koning. *Polyline Simplification*. Available at `https://www.codeproject.com/Articles/114797/Polyline-Simplification`, last accessed: 09/08/2019. 2011.

[144] Fumio Koyama. “High Power VCSEL Amplifier for 3D Sensing”. In: *CLEO: Science and Innovations*. Optical Society of America. 2020, STu4M–3.

[145] Thomas Krijnen and Jakob Beetz. “An IFC schema extension and binary serialization format to efficiently integrate point cloud data into building models”. In: *Advanced Engineering Informatics* 33 (2017), pp. 473–490.

[146] Heiner Lasi, Peter Fettke, Hans-Georg Kemper, Thomas Feld, and Michael Hoffmann. “Industry 4.0”. In: *Business & information systems engineering* 6.4 (2014), pp. 239–242.

[147] Sarel Lavy and Salil Jawadekar. *A case study of using BIM and COBie for facility management*. 2014.

[148] Donghwan Lee, Gichun Cha, and Seunghee Park. “A study on data visualization of embedded sensors for building energy monitoring using BIM”. In: *International journal of precision engineering and manufacturing* 17.6 (2016), pp. 807–814.

[149] Seul-Ki Lee, Hyo-Kyung An, and Jung-Ho Yu. “An extension of the technology acceptance model for BIM-based FM”. In: *Construction Research Congress 2012: Construction Challenges in a Flat World*. 2012, pp. 602–611.

[150] Wan-Li Lee, Meng-Han Tsai, Cheng-Hsuan Yang, Jhih-Ren Juang, and Jui-Yu Su. “V3DM+: BIM interactive collaboration system for facility management”. In: *Visualization in Engineering* 4.1 (2016), p. 5.

[151] Jennifer Li, David Greenwood, and Mohamad Kassem. “Blockchain in the built environment: analysing current applications and developing an emergent framework”. In: Diamond Congress Ltd. 2018.

[152] Seng Poh Lim and Habibollah Haron. “Surface reconstruction techniques: a review”. In: *Artificial Intelligence Review* 42.1 (2014), pp. 59–78.

[153] Xiaojun Liu, Ning Xie, and Jinyuan Jia. “WebVis_BIM: real time web3D visualization of big BIM data”. In: *Proceedings of the 14th ACM SIGGRAPH International Conference on Virtual Reality Continuum and its Applications in Industry*. ACM. 2015, pp. 43–50.

[154] Xinyi Liu, Yongjun Zhang, Xiao Ling, Yi Wan, Linyu Liu, and Qian Li. “TopoLAP: Topology Recovery for Building Reconstruction by Deducing the Relationships between Linear and Planar Primitives”. In: *Remote Sensing* 11.11 (2019), p. 1372.

[155] Stuart Lloyd. “Least squares quantization in PCM”. In: *IEEE transactions on information theory* 28.2 (1982), pp. 129–137.

[156] Steve Lockley, Claudio Benghi, and Martin Černý. “Xbim.Essentials: A Library for Interoperable Building Information Applications”. In: *The Journal of Open Source Software* 2.20 (2017), p. 473.

[157] Facundo José López, Pedro M Lerones, José Llamas, Jaime Gómez-Garcıa-Bermejo, and Eduardo Zalama. “A review of heritage building information modeling (H-BIM)”. In: *Multimodal Technologies and Interaction* 2.2 (2018), p. 21.

[158] Ruodan Lu and Ioannis Brilakis. “Digital twinning of existing reinforced concrete bridges from labelled point clusters”. In: *Automation in Construction* 105 (2019), p. 102837.

[159] C Matthew MacKenzie, Ken Laskey, Francis McCabe, Peter F Brown, Rebekah Metz, and Booz Allen Hamilton. “Reference model for service oriented architecture 1.0”. In: *OASIS standard* 12.S 18 (2006).

[160] Alexander Majercik, Cyril Crassin, Peter Shirley, and Morgan McGuire. “A ray-box intersection algorithm and efficient dynamic voxel rendering”. In: *Journal of Computer Graphics Techniques Vol* 7.3 (2018).

[161] ES Malinverni, R Pierdicca, M Paolanti, M Martini, C Morbidoni, F Matrone, and A Lingua. “Deep Learning for Semantic Segmentation of 3D Point Cloud”. In: *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences* (2019).

[162] Gerardo Santillán Martínez, Seppo Sierla, Tommi Karhela, and Valeriy Vyatkin. “Automatic Generation of a Simulation-Based Digital Twin of an Industrial Process Plant”. In: *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*. IEEE. 2018, pp. 3084–3089.

[163] Donald JR Meagher. “Octree encoding: A new technique for the representation, manipulation and display of arbitrary 3-d objects by computer”. In: *Technical Report IPL-TR-80-111* (1980).

[164] Hao Men, Biruk Gebre, and Kishore Pochiraju. “Color point cloud registration with 4D ICP algorithm”. In: *2011 IEEE International Conference on Robotics and Automation*. IEEE. 2011, pp. 1511–1516.

[165] Ralph C Merkle. “A digital signature based on a conventional encryption function”. In: *Conference on the theory and application of cryptographic techniques*. Springer. 1987, pp. 369–378.

[166] Patrick Min. *binvox*. Available at `http://www.patrickmin.com/binvox`, last accessed: 19/04/2020. 2019.

[167] Niloy J Mitra and An Nguyen. “Estimating surface normals in noisy point cloud data”. In: *Proceedings of the nineteenth annual symposium on Computational geometry*. ACM. 2003, pp. 322–328.

[168] Carsten Moenning and Neil A Dodgson. “Intrinsic point cloud simplification”. In: *Proc. 14th GrahiCon* 14 (2004), p. 23.

[169] Srivathsan Murali, Pablo Speciale, Martin R Oswald, and Marc Pollefeys. “Indoor scan2bim: Building information models of house interiors”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 6126–6133.

[170] James Mure-Dubois and Heinz Hügli. “Fusion of time of flight camera point clouds”. In: 2008.

[171] Esmeide A Leal Narváez and Nallig Eduardo Leal Narváez. “Point cloud denoising using robust principal component analysis.” In: *GRAPP*. 2006, pp. 51–58.

[172] Worawan Natephra, Ali Motamedi, Nobuyoshi Yabuki, and Tomohiro Fukuda. “Integrating 4D thermal information with BIM for building envelope thermal performance analysis and thermal comfort evaluation in naturally ventilated environments”. In: *Building and Environment* 124 (2017), pp. 194–208.

[173] Anh Nguyen and Bac Le. “3D point cloud segmentation: A survey”. In: *2013 6th IEEE conference on robotics, automation and mechatronics (RAM)*. IEEE. 2013, pp. 225–230.

[174] Shayan Nikoohemat, Abdoulaye A Diakité, Sisi Zlatanova, and George Vosselman. “Indoor 3D reconstruction from point clouds for optimal routing in complex buildings to support disaster management”. In: *Automation in construction* 113 (2020), p. 103109.

[175] Erica Nocerino, Fabio Polesi, Fabio Remondino, and Luc Van Gool. “Point clouds from smartphones”. In: *Gim International* 32.3 (2018), pp. 18–21.

[176] Fakir S. Nooruddin and Greg Turk. “Simplification and Repair of Polygonal Models Using Volumetric Techniques”. In: *IEEE Transactions on Visualization and Computer Graphics* 9.2 (2003), pp. 191–205.

[177] Hichri Nouha, Chiara Stefani, Livio de Luca, Philippe Véron, and Gael Hamon. “From Point Cloud to BIM: A Survey of Existing Approaches”. In: *XXIV International CIPA Symposium*. Vol. 40. 2013, pp. 343–348.

[178] 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.

[179] Sebastián Ortega, José Miguel Santana, Jochen Wendel, Agustın Trujillo, and Syed Monjur Murshed. “Generating 3D City Models from Open LiDAR Point Clouds: Advancing Towards Smart City Applications”. In: *Open Source Geospatial Science for Urban Studies*. Springer, pp. 97–116.

[180] Meltem Ozturk. “Real Estate 4.0 and the Digital Marketing Age”. In: *Selected Writings on Financial and Economical Behaviours in the New Economy* (2019), p. 100.

[181] Albert Palomer, Pere Ridao, David Ribas, Angelos Mallios, and Guillem Vallicrosa. “Octree-based subsampling criteria for bathymetric SLAM”. In: *Proceedings of the XXXV Jornadas de Automática, Valencia, Spain* (2014), pp. 3–5.

[182] Erika A Pärn, David J Edwards, and Michael CP Sing. “The building information modelling trajectory in facilities management: A review”. In: *Automation in construction* 75 (2017), pp. 45–55.

[183] Erika Anneli Pärn and David Edwards. “Vision and advocacy of optoelectronic technology developments in the AECO sector”. In: *Built Environment Project and Asset Management* 7.3 (2017), pp. 330–348.

[184] Daniela Pasini, Silvia Mastrolembo Ventura, Stefano Rinaldi, Paolo Bellagente, Alessandra Flammini, and Angelo Luigi Camillo Ciribini. “Exploiting Internet of Things and building information modeling framework for management of cognitive buildings”. In: *2016 IEEE International Smart Cities Conference (ISC2)*. IEEE. 2016, pp. 1–6.

[185] Panagiotis Patlakas, Georgios Koronaios, Rokia Raslan, Gareth Neighbour, and Hasim Altan. “Case studies of environmental visualization”. In: *Energies* 10.10 (2017), p. 1459.

[186] Tuomas Pelkonen, Scott Franklin, Justin Teller, Paul Cavallaro, Qi Huang, Justin Meza, and Kaushik Veeraraghavan. “Gorilla: A fast, scalable, in-memory time series database”. In: *Proceedings of the VLDB Endowment* 8.12 (2015), pp. 1816–1827.

[187] Jean-Jacques Ponciano, Alain Trémeau, and Frank Boochs. “Automatic Detection of Objects in 3D Point Clouds Based on Exclusively Semantic Guided Processes”. In: *ISPRS International Journal of Geo-Information* 8.10 (2019), p. 442.

[188] Jorge Posada, Mikel Zorrilla, Ana Dominguez, Bruno Simoes, Peter Eisert, Didier Stricker, Jason Rambach, Jürgen Döllner, and Miguel Guevara. “Graphics and media technologies for operators in industry 4.0”. In: *IEEE computer graphics and applications* 38.5 (2018), pp. 119–132.

[189] Matti Pouke, Juho-Pekka Virtanen, Mahmoud Badri, and Timo Ojala. "Comparison of two workflows for Web-based 3D smart home visualizations". In: *2018 IEEE International Conference on Future IoT Technologies (Future IoT)*. IEEE. 2018, pp. 1–8.

[190] Florent Poux, Romain Neuville, Pierre Hallot, and Roland Billen. "Smart point cloud: Definition and remaining challenges". In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 4.W1 (2016), pp. 119–127.

[191] Florent Poux, Romain Neuville, Gilles-Antoine Nys, and Roland Billen. "3D Point Cloud Semantic Modelling: Integrated Framework for Indoor Spaces and Furniture". In: *Remote Sensing* 10.9 (2018), p. 1412.

[192] Chandler Prall. *ThreeCSG.js*. Available at `https://github.com/chandlerprall/ThreeCSG`, last accessed: 05/09/2019. 2012.

[193] Franco P Preparata and Michael I Shamos. "Computational geometry: an introduction". In: (1985).

[194] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. "Pointnet: Deep learning on point sets for 3d classification and segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 652–660.

[195] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. "Pointnet++: Deep hierarchical feature learning on point sets in a metric space". In: *Advances in neural information processing systems*. 2017, pp. 5099–5108.

[196] Xiuquan Qiao, Pei Ren, Schahram Dustdar, Ling Liu, Huadong Ma, and Junliang Chen. "Web AR: A Promising Future for Mobile Augmented Reality—State of the Art, Challenges, and Insights". In: *Proceedings of the IEEE* 107.4 (2019), pp. 651–666.

[197] Vivi Qiuchen Lu, Ajith Kumar Parlikad, Philip Woodall, Gishan Don Ranasinghe, and James Heaton. "Developing a Dynamic Digital Twin at a Building Level: using Cambridge Campus as Case Study". In: *International Conference on Smart Infrastructure and Construction 2019 (ICSIC) Driving data-informed decision-making*. ICE Publishing. 2019, pp. 67–75.

[198] Tan Qu and Wei Sun. "Usage of 3D Point Cloud Data in BIM (Building Information Modelling): Current Applications and Challenges". In: *Journal of Civil Engineering and Architecture* 9.11 (2015), pp. 1269–1278.

[199] Blanca Quintana, Samuel A Prieto, Antonio Adán, and Frédéric Bosché. "Door detection in 3D coloured point clouds of indoor environments". In: *Automation in Construction* 85 (2018), pp. 146–166.

[200] Edvinas Rasys, Michael Hodds, Nashwan Dawood, and Mohamad Kassem. "A Web3D enabled information integration framework for facility management". In: *Australasian Journal of Construction Economics and Building-Conference Series*. Vol. 2. 1. 2014, pp. 1–12.

[201] Paul Reed. *Virtual Reality - An Introduction*. Available at `http://paul-reed.co.uk/programming.html`, last accessed: 19/04/2020. 2017.

[202] Fabio Remondino, Maria Grazia Spera, Erica Nocerino, Fabio Menna, Francesco Nex, and Sara Gonizzi-Barsanti. "Dense image matching: comparisons and analyses". In: *2013 Digital Heritage International Congress (DigitalHeritage)*. Vol. 1. IEEE. 2013, pp. 47–54.

[203] Michael Repplinger, Alexander Löffler, Benjamin Schug, and Philipp Slusallek. "Sora: a service-oriented rendering architecture". In: *Proceedings of Software Engineering and Architectures for Realtime Interactive Systems* (2010).

[204] Rico Richter. "Concepts and techniques for processing and rendering of massive 3D point clouds". In: (2018).

[205] Rico Richter and Jürgen Döllner. "Out-of-core real-time visualization of massive 3D point clouds". In: *Proceedings of the 7th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*. ACM. 2010, pp. 121–128.

[206] Kathy Roper and Richard Payant. *The facility management handbook*. Amacom, 2014.

[207] E Rosen, E Jansson, and M Brundin. "Implementation of a fast and efficient concave hull algorithm". In: *Uppsala Univ* (2014).

[208] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, *et al.* "Imagenet large scale visual recognition challenge". In: *International Journal of Computer Vision* 115.3 (2015), pp. 211–252.

[209] Philip Russom *et al.* "Big data analytics". In: *TDWI best practices report, fourth quarter* 19.4 (2011), pp. 1–34.

[210] Radu Bogdan Rusu. "Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments". PhD thesis. Computer Science department, Technische Universitaet Muenchen, Germany, 2009.

[211] Radu Bogdan Rusu and Steve Cousins. "3d is here: Point cloud library (pcl)". In: *2011 IEEE international conference on robotics and automation*. IEEE. 2011, pp. 1–4.

[212] Rafael Sacks, Charles Eastman, Ghang Lee, and Paul Teicholz. *BIM handbook: a guide to building information modeling for owners, designers, engineers, contractors, and facility managers*. ISBN: 9781119287544. John Wiley & Sons, 2018.

[213] Farhad Sadeghineko, Bimal Kumar, and Warren Chan. "A semantic web-based approach for generating parametric models using RDF". In: *Workshop of the European Group for Intelligent Computing in Engineering*. Springer. 2018, pp. 361–377.

[214] Thunyathep Santhanavanich, Sven Schneider, Preston Rodrigues, and Volker Coors. "Integration and Visualization of Heterogeneous Sensor Data and Geospatial Information". In: *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences* 4 (2018).

[215] Oliver Schall, Alexander Belyaev, and H-P Seidel. "Robust filtering of noisy scattered point data". In: *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005.* IEEE. 2005, pp. 71–144.

[216] M Scheffer, M Konig, T Engelmann, LC Tagliabue, ALC Ciribini, S Rinaldi, and M Pasetti. "Evaluation of Open Data Models for the Exchange of Sensor Data in Cognitive Building". In: *2018 Workshop on Metrology for Industry 4.0 and IoT*. IEEE. 2018, pp. 151–156.

[217] Andreas Schiefer, René Berndt, Torsten Ullrich, Volker Settgast, and Dieter W Fellner. "Service-oriented scene graph manipulation". In: *Proceedings of the 15th International Conference on Web 3D Technology*. 2010, pp. 55–62.

[218] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. "Efficient RANSAC for point-cloud shape detection". In: *Computer graphics forum*. Vol. 26. 2. Wiley Online Library. 2007, pp. 214–226.

[219] Markus Schütz. "Potree: Rendering large point clouds in web browsers". In: *Technische Universität Wien, Wiedeń* (2016).

[220] Timothy Scully, Jozef Doboš, Timo Sturm, and Yvonne Jung. "3drepo. io: building the next generation Web3D repository with AngularJS and X3DOM". In: *Proceedings of the 20th international conference on 3D web technology*. ACM. 2015, pp. 235–243.

[221] Evgenii Semenishchev, Viacheslav Voronin, Aleksandr Zelensky, Irina Tolstova, and Ilya Khamidullin. "Identification of local features on a group of images obtained in different electromagnetic ranges". In: *Signal Processing, Sensor/Information Fusion, and Target Recognition XXIX*. Vol. 11423. International Society for Optics and Photonics. 2020, p. 114230L.

[222] Madhumitha Senthilvel, Ranjith K Soman, and Koshy Varghese. "Comparison of Handheld devices for 3D Reconstruction in Construction". In: *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*. Vol. 34. 2017.

[223] Claude E Shannon. "A mathematical theory of communication". In: *Bell system technical journal* 27.3 (1948), pp. 379–423.

[224] Wenzhong Shi and ChuiKwan Cheung. “Performance evaluation of line simplification algorithms for vector generalization”. In: *The Cartographic Journal* 43.1 (2006), pp. 27–44.

[225] Mohd Fairuz Shiratuddin and Walid Thabet. “Virtual office walkthrough using a 3D game engine”. In: *International Journal of Design Computing* 4 (2002).

[226] Ben Shneiderman. “The eyes have it: A task by data type taxonomy for information visualizations”. In: *Proceedings 1996 IEEE symposium on visual languages*. IEEE. 1996, pp. 336–343.

[227] R Sihombing and V Coors. “Linking 3D Building Models, Maps and Energy-related Data in a Web-based Visualization System”. In: *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences* 4 (2018).

[228] Bradley Skinner, Teresa Vidal-Calleja, J Valls Miro, Freek De Bruijn, and Raphael Falque. “3D point cloud upsampling for accurate reconstruction of dense 2.5 D thickness maps”. In: *Australasian Conference on Robotics and Automation, ACRA*. 2014.

[229] Fredrik Smedberg. “Performance analysis of javascript”. PhD thesis. Linköping University, Department of Computer and Information Science, 2010.

[230] Peter Smith. “BIM & the 5D project cost manager”. In: *Procedia-Social and Behavioral Sciences* 119 (2014), pp. 475–484.

[231] John P Snyder. *Flattening the earth: two thousand years of map projections*. University of Chicago Press, 1997.

[232] V. Stojanovic, M. Trapp, R. Richter, and J. Döllner. “A Service-Oriented Indoor Point Cloud Processing Pipeline”. In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLII-2/W17 (2019), pp. 339–346.

[233] V Stojanovic, R Falconer, J Isaacs, D Gilmour, and D Blackwood. “Interactive visualisation to support sustainability assessment in land use scenario planning”. In: *Proceedings of the 29th Annual ARCOM Conference* 1373 (2013), p. 1382.

[234] V Stojanovic, R Richter, J Döllner, and M Trapp. “Comparative Visualization of BIM Geometry and Corresponding Point Clouds”. In: *International Journal of Sustainable Development and Planning* 13.1 (2018), pp. 12–23.

[235] Vladeta Stojanovic, Benjamin Hagedorn, Matthias Trapp, and Jürgen Döllner. “Ontology-Driven Analytics for Indoor Point Clouds”. In: *RE: Anthropocene, Design in the Age of Humans - Proceedings of the 25th CAADRIA Conference - Volume 2*. 2020, pp. 537–546.

[236] Vladeta Stojanovic, Matthias Trapp, Jürgen Döllner, and Rico Richter. “Classification of Indoor Point Clouds Using Multiviews”. In: *The 24th International Conference on 3D Web Technology*. ACM. 2019, pp. 1–9.

[237] Vladeta Stojanovic, Matthias Trapp, Benjamin Hagedorn, Jan Klimke, Rico Richter, and Jürgen Döllner. “Sensor Data Visualization for Indoor Point Clouds.” In: *Advances in Cartography and GIScience of the ICA* 2 (2019), pp. 1–8.

[238] Vladeta Stojanovic, Matthias Trapp, Rico Richter, and Jürgen Döllner. “A service-oriented approach for classifying 3D points clouds by example of office furniture classification”. In: *Proceedings of the 23rd International ACM Conference on 3D Web Technology*. 2018, pp. 1–9.

[239] Vladeta Stojanovic, Matthias Trapp, Rico Richter, and Jürgen Döllner. “Comparison of Deep-Learning Classification Approaches for Indoor Point Clouds”. In: *Publikationen der DGPF, Band 29, 2020* (2020), pp. 1–11.

[240] Vladeta Stojanovic, Matthias Trapp, Rico Richter, and Jürgen Döllner. “Generation of Approximate 2D and 3D Floor Plans from 3D Point Clouds”. In: *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 1: GRAPP,* INSTICC. SciTePress, 2019, pp. 177–184.

[241] Vladeta Stojanovic, Matthias Trapp, Rico Richter, and Jürgen Döllner. “Service-oriented semantic enrichment of indoor point clouds using octree-based multiview classification”. In: *Graphical Models* 105 (2019), p. 101039.

[242] Vladeta Stojanovic, Matthias Trapp, Rico Richter, Benjamin Hagedorn, and Jürgen Döllner. “Towards the Generation of Digital Twins for Facility Management Based on 3D Point Clouds”. In: *Gorse, C and Neilson, C J (Eds) Proceeding of the 34th Annual ARCOM Conference*. ARCOM. 2018, pp. 270–279.

[243] Paul S Strauss and Rikk Carey. “An object-oriented 3D graphics toolkit”. In: *ACM SIGGRAPH Computer Graphics* 26.2 (1992), pp. 341–349.

[244] Ian Stroud. *Boundary representation modelling techniques*. Springer Science & Business Media, 2006.

[245] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. “Multi-view convolutional neural networks for 3d shape recognition”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 945–953.

[246] Jong-Chyi Su, Matheus Gadelha, Rui Wang, and Subhransu Maji. “A deeper look at 3D shape classifiers”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, p. 16.

[247] Catherine A Sugar and Gareth M James. “Finding the number of clusters in a dataset: An information-theoretic approach”. In: *Journal of the American Statistical Association* 98.463 (2003), pp. 750–763.

[248] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. “Rethinking the inception architecture for computer vision”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2818–2826.

[249] Saeed Talebi, Lauri Koskela, and Patricia Tzortzopoulos. “Tolerance Compliance Measurement Using Terrestrial Laser Scanner”. In: *26th Annual Conference of the International Group for Lean Construction: Evolving Lean Construction-Towards Mature Production Across Cultures and Frontiers*. 2018, pp. 166–176.

[250] Pingbo Tang, Daniel Huber, Burcu Akinci, Robert Lipman, and Alan Lytle. “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.

[251] Shu Tang, Dennis R Shelden, Charles M Eastman, Pardis Pishdad-Bozorgi, and Xinghua Gao. “A review of building information modeling (BIM) and the internet of things (IoT) devices integration: Present status and future trends”. In: *Automation in Construction* 101 (2019), pp. 127–139.

[252] Paul Teicholz *et al. BIM for facility managers*. John Wiley & Sons, 2013.

[253] Jochen Teizer, Mario Wolf, Olga Golovina, Manuel Perschewski, Markus Propach, Matthias Neges, and Markus König. “Internet of Things (IoT) for integrating environmental and localization data in Building Information Modeling (BIM)”. In: *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*. Vol. 34. Vilnius Gediminas Technical University. 2017.

[254] BA Tezel, ZUH Aziz, *et al.* “From conventional to IT based visual management: a conceptual discussion for lean construction”. In: *Journal of information technology in construction* 22 (2017), pp. 220–246.

[255] Charles Thomson and Jan Boehm. “Automatic geometry generation from point clouds for BIM”. In: *Remote Sensing* 7.9 (2015), pp. 11753–11775.

[256] Russell Toris, Julius Kammerl, David V Lu, Jihoon Lee, Odest Chadwicke Jenkins, Sarah Osentoski, Mitchell Wills, and Sonia Chernova. “Robot web tools: Efficient messaging for cloud robotics”. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 4530–4537.

[257] H Tran, K Khoshelham, A Kealy, and L Dıaz-Vilariño. “Shape Grammar Approach to 3D Modeling of Indoor Environments Using Point Clouds”. In: *Journal of Computing in Civil Engineering* 33.1 (2018), p. 04018055.

[258] Matthias Trapp and Jürgen Döllner. “Real-Time Volumetric Tests Using Layered Depth Images”. In: *Eurographics (Short Papers)*. 2008, pp. 49–52.

[259] Francisco Troncoso-Pastoriza, Javier López-Gómez, and Lara Febrero-Garrido. “Generalized Vision-Based Detection, Identification and Pose Estimation of Lamps for BIM Integration”. In: *Sensors* 18.7 (2018), p. 2364.

[260] Yelda Turkan, Frédéric Bosché, Carl T. Haas, and Ralph Haas. “Tracking of secondary and temporary objects in structural concrete work”. In: *Construction Innovation* 14.2 (2014), pp. 145–167.

[261] Renaud Vanlande, Christophe Nicolle, and Christophe Cruz. “IFC and building lifecycle management”. In: *Automation in construction* 18.1 (2008), pp. 70–78.

[262] Juho-Pekka Virtanen, Matti Kurkela, Hannu Hyyppä, Sami Niemi, Sami Kalliokoski, Seppo Vanhatalo, Juha Hyyppä, Henrik Haggrén, Suvi Nenonen, Juha-Matti Junnonen, *et al.* "Visualization of building models and sensor data using open 3D platforms". In: *Proceedings of the CIB World Building Congress*. Vol. 4. Tampere University of Technology. 2016, pp. 178–189.

[263] Juho-Pekka Virtanen, Matti Kurkela, Tuomas Turppa, Matti T Vaaja, Arttu Julin, Antero Kukko, Juha Hyyppä, Marika Ahlavuo, Jessica Edén von Numers, Henrik Haggrén, *et al.* "Depth camera indoor mapping for 3D virtual radio play". In: *The Photogrammetric Record* 33.162 (2018), pp. 171–195.

[264] Maheswari Visvalingam and James D Whyatt. "Line generalisation by repeated elimination of points". In: *The cartographic journal* 30.1 (1993), pp. 46–51.

[265] 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.

[266] George Vosselman, Ben GH Gorte, George Sithole, and Tahir Rabbani. "Recognising structure in laser scanner point clouds". In: *International archives of photogrammetry, remote sensing and spatial information sciences* 46.8 (2004), pp. 33–38.

[267] Aaron E Walsh and Mikaël Bourges-Sévenier. *core WEB3D*. Prentice Hall Professional, 2001.

[268] Jun Wang, Weizhuo Sun, Wenchi Shou, Xiangyu Wang, Changzhi Wu, Heap-Yih Chong, Yan Liu, and Cenfei Sun. "Integrating BIM and LiDAR for real-time construction quality control". In: *Journal of Intelligent & Robotic Systems* 79.3-4 (2015), pp. 417–432.

[269] L Wang and G Sohn. "Automatic co-registration of terrestrial laser scanning data and 2D floor plan". In: *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 38.Part 1 (2010), pp. 158–164.

[270] Qian Wang and Min-Koo Kim. "Applications of 3D point cloud data in the construction industry: A fifteen-year review from 2004 to 2018". In: *Advanced Engineering Informatics* 39 (2019), pp. 306–319.

[271] Weiyue Wang, Ronald Yu, Qiangui Huang, and Ulrich Neumann. "Sgpn: Similarity group proposal network for 3d point cloud instance segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2569–2578.

[272] Martin Weinmann, Boris Jutzi, and Clément Mallet. "Feature relevance assessment for the semantic interpretation of 3D point cloud data". In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 5.W2 (2013), pp. 313–318.

[273] Karl Wiegers and Joy Beatty. *Software requirements*. Pearson Education, 2013.

[274] Arif Wismadi. “Predicting Future Urban Housing Frontage Typology: Building Façade as the Interface of Transport, Logistics, and AEC 4.0”. In: *EduARCHsia & Senvar 2019 International Conference (EduARCHsia 2019)*. Atlantis Press. 2020, pp. 171–175.

[275] Johannes Wolf, Rico Richter, and Jürgen Döllner. “Techniques for Automated Classification and Segregation of Mobile Mapping 3D Point Clouds”. In: *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 1 GRAPP: GRAPP,* INSTICC. SciTePress, 2019, pp. 201–208.

[276] Jeong Han Woo. “BIM (building information modeling) and pedagogical challenges”. In: *Proceedings of the 43rd ASC national annual conference*. 2006, pp. 12–14.

[277] Jainhua Wu and Leif Kobbelt. “Optimized sub-sampling of point sets for surface splatting”. In: *Computer Graphics Forum*. Vol. 23. 3. Wiley Online Library. 2004, pp. 643–652.

[278] Xuehan Xiong, Antonio Adan, Burcu Akinci, and Daniel Huber. “Automatic creation of semantically rich 3D building models from laser scanner data”. In: *Automation in construction* 31 (2013), pp. 325–337.

[279] Fengting Yan, Jinyuan Jia, Yonghao Hu, Qinghua Guo, and Hehua Zhu. “Smart fire evacuation service based on Internet of Things computing for Web3D”. In: *Journal of Internet Technology* 20.2 (2019), pp. 521–532.

[280] Wei Yan, Charles Culp, and Robert Graf. “Integrating BIM and gaming for real-time interactive architectural visualization”. In: *Automation in Construction* 20.4 (2011), pp. 446–458.

[281] Fenggen Yu, Kun Liu, Yan Zhang, Chenyang Zhu, and Kai Xu. “PartNet: A Recursive Part Decomposition Network for Fine-Grained and Hierarchical Shape Segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9491–9500.

[282] Andrea Zanella, Nicola Bui, Angelo Castellani, Lorenzo Vangelista, and Michele Zorzi. “Internet of things for smart cities”. In: *IEEE Internet of Things journal* 1.1 (2014), pp. 22–32.

[283] JP Zhang, Q Liu, FQ Yu, ZZ Hu, and WZ Zhao. “A framework of cloud-computing-based BIM service for building lifecycle”. In: *Computing in Civil and Building Engineering (2014)*. 2014, pp. 1514–1521.

[284] Xiao-Yang Zhang, Zhen-Zhong Hu, Heng-Wei Wang, and Mohamad Kassem. “An Industry Foundation Classes (IFC) web-based approach and Platform for Bi-Drectional conversion of structural analysis models”. In: *Computing in Civil and Building Engineering (2014)*. 2014, pp. 390–397.

[285] Xiaopeng Zhang, Hongjun Li, and Zhanglin Cheng. “Curvature estimation of 3D point cloud surfaces through the fitting of normal section curvatures”. In: *Proceedings of ASIAGRAPH* 2008 (2008), pp. 23–26.

[286] Hanli Zhao, Xiaogang Jin, Jianbing Shen, and Shufang Lu. “Fast and reliable mouse picking using graphics hardware”. In: *International Journal of Computer Games Technology* 2009 (2009).

[287] Hu Zhao, Zoltán Nagy, Daren Thomas, and Arno Schlueter. “Service-oriented architecture for data exchange between a building information model and a building energy model”. In: *Proceedings of International Conference CISBAT 2015 Future Buildings and Districts Sustainability from Nano to Urban Scale*. CONF. LESO-PB, EPFL. 2015, pp. 761–766.

[288] Xiaoping Zhou, Jia Wang, Ming Guo, and Zhe Gao. “Cross-platform online visualization system for open BIM based on WebGL”. In: *Multimedia Tools and Applications* 78.20 (2019), pp. 28575–28590.

Statutory Declaration

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text. Any thoughts from others or literal quotations are clearly marked. The PhD Thesis was not used in the same or in a similar version to achieve an academic grading, nor is being published elsewhere.

Potsdam, September 27, 2020
(Place, Date)

Vladeta Stojanovic
(Signature)