Direct Use of Indoor Point Clouds for Path Planning and Navigation Exploration in Emergency Situations

Master's Thesis – May 2024

Author

Algan Mert Yasar

Supervisors

ir. E. Verbree

ir. R. Voûte

Responsible Professor

Prof.dr.ir. P.J.M. van Oosterom

University

Delft University of Technology

MSc thesis in GIMA

Direct Use of Indoor Point Clouds for Path Planning and Navigation Exploration in Emergency Situations

Algan Mert Yasar

May 2024

Geo-Database Management Centre
Delft University of Technology

Supervisors: ir. Edward Verbree
ir. Robert Voûte
Co-reader: Prof.dr.ir. P.J.M. van Oosterom

Disclaimer

The results of this research are presented in both this thesis report and the research article titled ”Direct Use of Indoor Point Clouds for Path Planning and Navigation Exploration” to be published at 19th International 3D GeoInfo Conference 2024 (provided in Appendix A.1). Therefore, some figures and content appear in both works. Additionally, I acknowledge that ChatGPT was used for proofreading and enhancing readability.

Abstract

The aim of this study is to explore the direct use of 3D indoor point clouds for indoor navigation to support emergency response process. The majority of the research on indoor navigation is focused on generating a navigation system from a pre-existing indoor model and product is a navigation graph where rooms are represented as nodes and connecting structures (doors, corridors, etc.) are represented with edges This representation can be extracted from a simple floor plan and is useful for day-to-day applications. However, graph representations of indoor environments do not depict obstacles present in those rooms or does not direct the subject through the room to the next one which is an issue in case of a large open space with complex layout such as a modern office floor. Another fallacy of the graph-based methods is the time and data requirements. Processing the models into network graphs and communicating the planned path with the user takes time to process which is a scarce resource in the communication process of emergency response operations.

This research builds upon an existing project where 3D models in the form of point clouds are generated in real-time with the combination of Red Green and Blue (RGB) images from cameras and the output of the depth sensor located on HoloLens 2. The current system can deploy multiple scanners into the environment and generate a co-localized, common 3D model. Therefore, there is a common 3D model where relative positions of the field agents (explorers) are known. By combining the 3D model and available position information, this research aims to provide navigation support to a Place of Interest (POI) within the 3D model. As the point cloud is generated in real-time during the response process, navigation implementation in this 3D model should be able to work with minimal pre-processing and output should be easy to interpret by the end-user. Therefore, communication of the path is achieved by visual cues.

Achieving 'navigation support' in this context requires path planning using the point clouds as well as communication of the said path with the user (emergency responder) which is path visualization. The dataset of the study is obtained with HoloLens 2 as well as survey grade mobile laser scanners. Not only for assessing the capabilities of HoloLens 2 but also to explore possibilities of more capable sensors, which could become affordable and available in the future.

In contrary to the common methodologies used in indoor navigation, study utilizes point clouds as raw as possible instead of segmentation and network extraction. Which is an existing methodology developed for autonomous robots. As robots once learned walking by mimicking humans, in this study, we aim to learn where to walk from them. This study proposes a navigation solution for humans with the implementation of well known sampling based Rapidly Exploring Random Trees (RRT) algorithm.

Keywords: Indoor navigation, Path planning, Game engine, RRT, Real-time, Point cloud.[1]

[1] GitHub Repository

Acknowledgements

I would like to express my deepest gratitude to my supervisors, Edward and Robert, for their unwavering support and guidance throughout this process.

Edward not only provided invaluable assistance with my work but also shared numerous fascinating studies with me, broadening my perspective and deepening my understanding beyond the scope of my own research.

Despite having one of the busiest schedules I've ever encountered, Robert always made time to help whenever needed. His genuine concern for my well-being often exceeded my own, for which I am immensely grateful.

More than just supervisors for my thesis, I regard Edward and Robert as mentors who have profoundly influenced my academic and personal growth. I am also grateful for the opportunity to publish my first research paper together with them.

I would like to extend my heartfelt thanks to Prof. Peter van Oosterom for his insightful feedback and invaluable advice throughout my research journey. His guidance has been instrumental in shaping the direction and quality of my work.

I am also grateful to my friends and colleagues at CGI. Their support made this journey much more enjoyable, and I am truly thankful for the opportunity to work alongside such wonderful individuals.

Finally, I would also like to express my deep appreciation to my family, my girlfriend, and my friends for their endless support.

...

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

1. Introduction

1.1. Motivation

Current advances in Geo-ICT enable us to create digital twins with high accuracy. However, a digital twin alone does not offer any added value unless it is applied to a specific domain and leverages its potential benefits. Such an application also poses domain-specific challenges. One of these domains is emergency response, which can benefit from digital twins especially in indoor environments. Unlike outdoor emergencies, where responders can use satellite imagery, Global Navigation Satellite System (GNSS), drones, and other systems for reconnaissance and operational support, indoor environments (e.g. high-rise buildings with complex architectures) require entering the unknown, as there is no reliable information available other than floor plans, blueprints or Building Information Model (BIM), which may be not available or outdated as they are produced before/during construction and do not include renovations done after construction Nikoohemat et al. [2020]. The availability of a digital building logbook could change the status quo. However, according to the study published by the European Commission, there are 21 initiatives in place, yet only eight of them require updating after renovations. Of these eight initiatives, only three are digitized and accessible online Carbonari et al. [2020]. Until an up-to-date digital building passport is no longer a dream, other solutions must be explored as temporary solutions, and at the same time contributing to the dream.

Emergency response scenarios demand urgency and precision, as the nature of such events often involves saving lives and minimizing damage Kapucu and Garayev [2011]. Navigating an unfamiliar location such as a high-rise building without prior knowledge can be a significant challenge for emergency responders who need to quickly locate resources like fire extinguishers, safe spots, and alternative exits. Even a few seconds saved through an improved navigation system could have a substantial impact on the outcome of an emergency. Current indoor navigation systems primarily rely on satellite signals (such as GNSS) and local Wireless Fidelity (Wi-Fi) or Bluetooth signals for positioning, which might offer great accuracy under normal circumstances. However, these systems often fail in emergency scenarios where power outages or structural damage can disrupt local networks and block satellite connections. Furthermore, these systems typically depend on pre-existing building data like floor plans or BIM, which may be outdated or poorly maintained Nikoohemat et al. [2020], leading to inaccurate and potentially dangerous guidance during emergencies. Moreover, another pitfall of indoor navigation solutions is that they represent the environment in 2D and simplify the spatial complexity into nodes and connecting lines, which is efficient for everyday use but inadequate for the intricate layouts Boguslawski et al. [2022].

In the context of emergency response: using digital twins / Three-Dimensional (3D) models for indoor environments can support decision making and increase situation awareness of the responders Smit et al. [2021] but this also poses its own challenges. The first challenge is time; creating accurate digital twins is possible, but it takes time, which is a scarce resource in emergency response. Therefore, the process from scanning, processing and utilizing the

3D model should happen at least near-real-time. As the scanning and processing is done in real-time, data coming from multiple field agents should be combined simultaneously as well. This challenge is addressed with the concept of Collaborative Simultaneous Localization And Mapping (SLAM) in the existing system van Schendel [2022]. It should be noted that agents in the acquired indoor scans are positioned instead of localized as position refers to exact placement in an absolute or arbitrary reference system while location refer to relative placement in a well-defined space Sithole and Zlatanova [2016]. Therefore one can argue that the methodology should be called Simultaneous Positioning And Mapping (SPAM). However, the term will be avoided to refrain from possible confusion.

The second challenge is complexity; the introduced system should not obstruct the responder by any means and be robust enough to not add risks to the already challenging environment Angelats and Navarro [2017].

The third challenge is uncertainty; emergencies are fast-paced and dynamic environments where assumptions and dependencies are highly prone to failure Kapucu and Garayev [2011]. Communication connections can fail, cameras can be useless in smoke, and audio-based communication might not be feasible in certain situations. Therefore, the system should be able to keep up with the dynamic nature of the environment.

System that this research is building upon has addressed these challenges when providing a solution that allows multiple field agents to scan the environment and generate a 3D model in near-real-time Morlighem et al. [2020]; Smit et al. [2021]; van Schendel [2022]. This research is aiming to advance this system by developing a navigation support system while addressing and handling these challenges as well. Current system utilize Microsoft HoloLens 2's internal SLAM system for positioning, which does not depend on external connectivity other than local network connections provided with the system for data transfer. System overview can be seen in Figure 1.1 below.

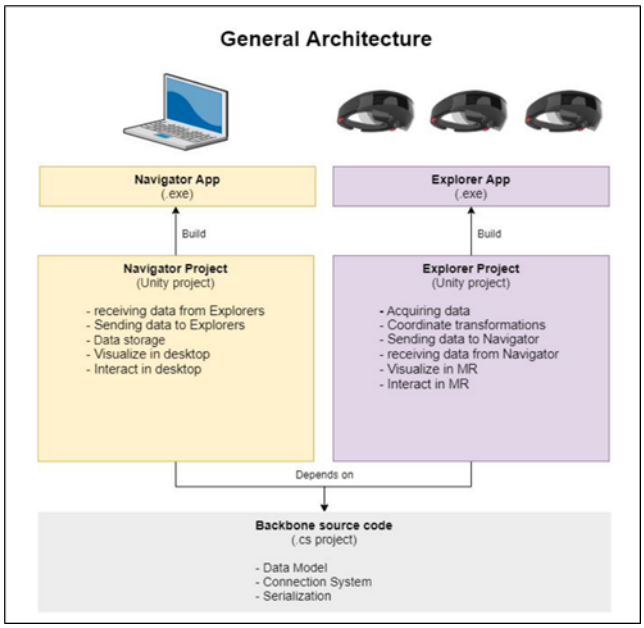


Figure 1.1.: System architecture Morlighem et al. [2020]

1.2. Gap in Knowledge

Traditional indoor navigation methodologies frequently depend on graph-based path planning, which necessitates pre-classifying the navigable spaces before routing can commence. This requirement renders such methodologies incompatible with projects that demand path planning concurrent with real-time indoor modeling. The approach taken in this research is to utilize sampling-based path planning, a technique commonly employed in the robotics field for autonomous vehicles, due to its suitability for dynamic search spaces without the need for pre-processed structures.

While sampling-based methods are well-established for navigating 2D search spaces, their application to 3D point clouds remains significantly under-explored. Moreover, these algorithms have not been widely adopted for navigation support systems intended for human use, particularly for emergency responders who operate in highly dynamic and unpredictable environments.

1.3. Solution Direction

This research seeks to address these limitations by implementing RRT path planning directly within point clouds. This innovative approach aims to achieve path planning speeds that are compatible with real-time scanning demands, while also ensuring that the navigation outputs are communicated in a format readily usable by emergency responders. By bridging this gap, the project intends to significantly advance the utility of indoor navigation systems in critical emergency response scenarios.

As sensor technology continues to advance rapidly, the potential for integrating higher-quality sensors into real-time navigation systems becomes increasingly feasible. Notable developments, such as the introduction of LiDAR sensors in consumer devices like Apple's mobile phones and tablets starting in 2020, highlight the trend towards more accessible and powerful sensing technologies Díaz-Vilariño et al. [2022]. The recent release of Apple's Vision Pro during the course of this research further underscores the progress in wearable sensor technology. These advancements suggest that features typically reserved for high-end mobile laser scanners could soon be available in smaller, more affordable, and wearable formats.

Employing the modified bi-directional Rapidly Exploring Random Trees (RRT) algorithm for path finding, where the points in the point cloud as obstacles, using any available empty space to establish a path from the start to the endpoint. The RRT algorithm (LaValle [1998]) is chosen for its efficiency in rapidly processing and adapting to new environmental data, crucial for navigating dynamically changing indoor environments during emergencies. Unlike traditional navigation systems that simplify environmental data into 2D maps (Balado et al. [2019]), this solution maintains the three-dimensional complexity of the environment. Therefore, allows for fast and effective navigation through and around physical obstacles within complex indoor settings, which is essential in emergency scenarios where environments are prone to sudden alterations such as obstructions from debris or changes in accessible paths.

The proposed navigation system can swiftly adapt to the dynamically changing environment of an emergency. By enhancing decision-making processes and providing efficient

paths to safety or to other agents in distress, the system optimizes the use of point cloud data with minimal processing. This navigation solution thus offers an alternative to conventional methods by leveraging advanced scanning technology and path finding algorithms, facilitating more effective and reliable navigation in complex indoor environments during emergency responses.

1.4. Scope

- Data acquisition with survey grade and low-cost sensors.

  Study uses data obtained from three different scanners: one is the Microsoft HoloLens 2, used to evaluate what could be achieved with the current system, and the other two are survey-grade mobile laser scanners (MLSs), namely GeoSLAM ZEB Horizon RT and Leica BLK2GO. Even though their workflows do not fit with the real-time mapping system, they are employed in this research to see what is achievable if sensors with higher point density and wider fields of view become available.

- Development of a path planning algorithm that utilizes unprocessed point clouds.

  Employing the modified bi-directional Rapidly Exploring Random Trees (RRT) algorithm for path finding, where the points in the point cloud as obstacles, using any available empty space to establish a path from the start to the endpoint. The RRT algorithm (LaValle [1998]) is chosen for its efficiency in rapidly processing and adapting to new environmental data, crucial for navigating dynamically changing indoor environments during emergencies. Unlike traditional navigation systems that simplify environmental data into 2D maps (Balado et al. [2019]), this solution maintains the three-dimensional complexity of the environment. Therefore, allows for fast and effective navigation through and around physical obstacles within complex indoor settings.

- Proof of concept / Demonstrator for visualization of the path.

  Study proposes a visualization methodology that shares a similar framework with the real-time 3D indoor modelling system. Therefore, it can be implemented into the system in the future.

1.4.1. Scope Limitations

Following constraints are placed due to time or resource limitations.

- First product will be developed with pre-existing high coverage point cloud.
- Indoor point clouds used in the study will cover a single story/floor. In case of a successful implementation, complexity of the environment can be increased (e.g., complex layout, multiple floors).
- Static POIs will be pre-determined.
- Navigation support (visualization) will be developed only as a Proof of Concept / Demonstrator

1.4.2. Out of Scope

- Mixed Reality (MR) visualization
- Complete navigation system merging with real-time scanning framework
- Application Programming Interface (API) development for Zeb Horizon and BLK2GO

1.5. Report Overview

Remainder of the report is structured into four main chapters: Research Objectives, Related Works, Methodology, and Results and Discussion.

The Research Objectives chapter outlines objectives of the research, briefly introducing what will be explored under each sub-question.

The Related Works chapter explains the concepts such as the path planning algorithms, navigation methodologies and performance related implementations. Additionally, chapter also elaborates on the reasoning behind selection of the methodologies used in this research.

The Methodology chapter elaborates in detail on how the work is set up and conducted. This includes technical comparison of the hardware, data acquisition, pre-processing of Mobile Laser Scanner (MLS) output, path planning and visualization.

Finally, the Results and Discussion chapter summarizes the outcomes of the methodology, discusses the answers to the questions mentioned in the Research Objectives chapter, and concludes with a discussion on challenges and future possibilities.

2. Research Objectives

The primary goal of this study is to explore the feasibility of using unprocessed point clouds for real-time navigation support in emergency response applications. This research develops a methodology for routing to a POI using raw point clouds, accompanied by a proof-of-concept User Interface (UI) for communicating these routes. Leveraging the capabilities of an existing real-time 3D indoor scanning system that outputs point clouds directly, this project aims to minimize pre-processing to enhance the speed and efficiency of path planning and navigation support. This module is designed to assist field agents in effectively navigating to POIs during emergencies.

This chapter outlines the main research question and the subsequent sub-questions that guide the investigation, laying the foundational steps towards achieving the research objectives.

2.1. Main Research Question

Central question of the research is:

"To what extent can we provide navigation support to places of interest using 3D indoor point clouds?"

2.2. Sub-Questions

Central research question is divided into four sub-questions to explore and answer "to what extent" it is possible to provide navigation support. Each and every sub-question can be seen as a stepping stone in this process.

- Sub-Question 1: How to navigate to a POI in a static 3D indoor point cloud?
- Sub-Question 2: How to navigate to a POI in a 3D indoor point cloud generated in real-time?
- Sub-Question 3: How to communicate POIs between units and provide navigation support to the explorers?
- Sub-Question 4: Is the solution reliable and feasible for emergency response applications?

2.2.1. How to navigate to a POI in a static 3D indoor point cloud?

Answer to this sub-question can be used to utilize an existing indoor point cloud obtained before the emergency response and routing obtained with this method can be communicated to the responders (explorers) which could speed up the decision making processes. Additionally, methodology of this sub-question simulates the output of sensors with higher capabilities which could be feasible to implement in the real-time scanning framework in the future.

2.2.2. How to navigate to a POI in a 3D indoor point cloud generated in real-time?

Second step in this research is implementation of the routing algorithm developed with the first sub-question to the real-time point clouds. Answer of this sub-question allows emergency responders to not rely on pre-existing data and utilize data that can be obtained during the response process. Solution developed in this step builds upon the existing architecture developed by CGI.

2.2.3. How to communicate POIs between units and provide navigation support to the explorers?

Previous questions explores how to obtain routing between POIs and a crucial step in this process is having a way to communicate the routes and POIs between and among teams. Third sub-question will be answered with a proof of concept / demonstrator design aimed to facilitate this communication.

2.2.4. Is the solution reliable and feasible for emergency response applications?

The last step in this process is to evaluate the proposed solutions according to emergency response standards. Methods should be able to keep up with a fast changing environment, be robust and feasible as well as should not require any additional work from users (responders) to get used in emergency response.

3. Related Works

Requirements of indoor navigation can be considered under few main topics: indoor positioning, and path finding, mapping (modelling), and communication of the found path. The indoor positioning aspect of this project has been addressed by preceding studies (Morlighem et al. [2020]; Smit et al. [2021]; van Schendel [2022]). Therefore, this section will focus on the latter. This section follows the natural progression of indoor navigation: first, modelling; second, path finding; and lastly, navigation support/communication. Each of these subsections will be considered from the perspective of emergency response.

3.1. 3D Indoor Mapping Techniques

As it is also mentioned in the previous section; outdated models, lack of availability and inaccuracies prove that there is a need for an up-to-date model (Nikoohemat et al. [2020]), whether it be a floor plan, BIM, or 3D model. An up-to-date 2D floor plan would still represent a complex office floor as a single room. Therefore, the requirement is an up-to-date, 3D indoor model. BIMs could be an option. However, a case study by the European Commission (Carbonari et al. [2020]) indicated that out of 21 building logbook initiatives, only 8 require updates after renovations, and only 3 are digitalized and accessible. Consequently, even if an up-to-date 3D model exists, it might not be available. To summarize, 2D models are inadequate for representing complexity, and existing 3D models could be inaccessible or outdated. Therefore, real-time 3D reconstruction appears to be the most viable option for emergency response scenarios.

3.2. Indoor Path Finding

In this section, studies that utilize 3D indoor point clouds for path planning are explored. These studies separated into two subsections namely 2D and 3D navigation graph according to their methodologies.

3.2.1. 2D Navigation Graph

Most common methodology for navigation implementations are navigation graphs. This method requires segmenting and classifying the 3D model to extract the navigable surface of the floor plane. An example is the work of Balado et al. [2019], where they directly use a 3D point cloud and use a height histogram to extract the floor surfaces. Then classify the navigable surface from the horizontal surfaces by removing the areas that has obstacles over it. Final product is achieved by down sampling and generating a navigation graph where

Dijkstra's algorithm is used to calculate the route. Study also mention processing times longer than 10 minutes to achieve these results.

Another study Flikweert et al. [2019] proposed a methodology that automates the process from point clouds to navigation graphs, their methodology is much more feasible than the manual extraction as every emergency response will require another process. Therefore, automatizing of the process can save quite a lot of time, which can perform even better with high-capacity hardware. However, author also mentions that the system is not suitable for navigation as each individual indoor space is represented by a single node. This is another pitfall of graph methodology. Extraction of the navigable surface speeds up the path finding process (after extraction) however, simplifies the environment.

Another 2D graph methodology is proposed by Fichtner et al. [2018]. Author uses octree data structure and semantic enrichment to extract navigation graph from point clouds, providing a hybrid approach Broersen et al. [2016] and Balado et al. [2019]. Where amount of pre-processing is lower than 3D graph methodology yet resulting navigation graph is suitable for multi-story path finding hence more capable than the traditional approach. Yet, performance in terms of processing time is unknown as the author does not mention any testing with path planning algorithms.

3.2.2. 3D Navigation Graph

Another navigation graph approach is used by Broersen et al. [2016]. Authors propose extracting the 3D empty space from the point clouds and structuring it into a graph to achieve path finding. This method also accommodates and depicts the obstacles better as they did not simplify the navigation graph to 2D. As the dimension of the graph is increased, they propose octree spatial indexing to speed up the search within the graph. And in comparison to Balado et al. [2019], authors use the improved version of the Dijkstra's algorithm: A* which is more efficient in terms of search speed. However, even it if more efficient than other graph based methods, there is still the requirement for processing the 3D model into a 2D representation. This is not feasible as simplifying the obstacles and navigable surface into a 2D representation reduces the level of detail Nikoohemat et al. [2020] which is important for emergency response process.

The work of Rodenberg [2016] addresses the issue of obstacle simplification by implementing a 3D navigation graph obtained via an octree data structure, as well as 3D path-finding using the A* algorithm for enhanced performance, achieving fast and reliable path-finding results. However, this approach is not feasible for this project as the navigation system relies on a secondary product after processing. This means that in a 3D model expanding in real-time, the navigation graph needs to be regenerated with every added scene. Additionally, Rodenberg [2016] utilizes the empty nodes of the octree as graph nodes for path-finding, whereas in this project, the octree structure is used to enhance the performance of collision and neighborhood checks in sampling-based path-finding. The reason is that extracting empty nodes from the octree structure for path-finding introduces a second layer of processing required at every iteration as the 3D model expands in real-time. In contrast, sampling-based path-finding only requires the octree as it is for collision checks.

An important takeaway from the works of Broersen et al. [2016], Rodenberg [2016], and Fichtner et al. [2018] is that spatial access methods (Van Oosterom [1999]) are crucial for performance improvements.

It can be concluded that graph-based methodologies are not feasible due to pre-processing requirements. Therefore, the focus of the literature study shifted from pedestrian navigation to vehicle navigation with the aim of finding a method that could work with an unstructured, unprocessed point cloud.

3.3. Path Finding Algorithms

3.3.1. * (star) Algorithms - Graph Based Algorithms

This section is dedicated to * Search Algorithms as it is defined by Nosrati et al. [2012]. These algorithms are widely used in path finding in navigation graph based implementations. This algorithm class include (but not limited to): A*, B*, D*, IDA* and SMA*. Even though the Dijkstra's Algorithm is not a star algorithm by the name, it is the predecessor of the A* hence it could be classified in these group. Common point of these algorithms focus on path finding by using a series of nodes. These algorithms do not generate nodes and require the generation of the 'search space' prior to the search process. Therefore they require pre-processing. Studies of Broersen et al. [2016] and Rodenberg [2016] uses A* graph based methodology for path finding as mentioned earlier.

3.3.2. Sampling based Algorithms

Sampling-based path finding algorithms are a class of algorithms used in robotics and artificial intelligence for planning paths in high-dimensional spaces. These algorithms are particularly useful for solving problems where the environment is complex and cluttered with obstacles, making it difficult to compute a path analytically. Two well known sampling based algorithms are RRT and Probabilistic Roadmaps (PRM) while PRM operates similar to a graph based path finding methodology due to it's pre-processing requirement. Therefore following section focuses on RRT and it's variants.

RRT

An alternative to graph based alternatives; this research adopts a methodology that emerged in the field of robotics: **Rapidly Exploring Random Trees** which is first proposed by LaValle [1998]. RRT is a simple and effective sampling based algorithm that starts at an initial location and randomly expands by sampling the search space with the aim of reaching to the goal location. However, Base algorithm has a few shortcomings. First one is that the algorithm checks for existence of a path with a defined probability. This mean that the iterations might continue indefinitely until the check is triggered even if a path is available. Second is that iterations continue even after the path is found until the predefined iteration count is reached Noreen et al. [2016].

RRT*

RRT* is an enhancement of the RRT algorithm proposed by Karaman and Frazzoli [2011], designed to achieve asymptotic optimality while retaining probabilistic completeness. It incrementally builds a tree of feasible trajectories and refines the solution to ensure that the cost of the path converges to the minimum possible value with enough samples. However, focus on path optimization results in lower performance.

RRT*-Smart

RRT-Smart* is an enhanced version of the RRT* algorithm that introduces intelligent sampling and path optimization techniques to accelerate the convergence rate towards an optimal path developed by Nasir et al. [2013]. It uses biasing points derived from the initial path to guide the sampling process and continuously optimizes the path by connecting visible nodes. The study demonstrates that RRT*-Smart significantly improves the efficiency of path finding, reducing both the number of iterations and computational time required to find an optimal or near-optimal solution.

BTO-RRT

Study of Zheng et al. [2022] introduces the Bidirectional Target-Oriented Rapidly-exploring Random Tree algorithm which is a path planning algorithm that combines bidirectional RRT and RRT-connect approaches to generate target-oriented paths. It directly uses point cloud data to compute trajectories, bypassing the need for 3D map discretization. The three-step optimization process further refines the path, ensuring it is smooth and dynamically feasible. In this research authors also utilize a spatial access method (Van Oosterom [1999]) for object detection with the use of KD-Tree.

AEB-RRT*

Adaptive Extension Bidirectional RRT* is proposed by Wang et al. [2022]. Main addition of this implementation is utilizing Manhattan distance for node calculations and variable step side implementation for handling both large openings and small passages. In addition, study also implements post-optimization for found path.

3.4. Visualization and Path Communication

The final part of this section involves the visualization and communication of the path. The term ”communication” is used because the output of our path finding methodology is a list of coordinates, which might be adequate for an autonomous drone but not for a field agent. Our end user, a field agent, has limited time and resources to comprehend complex data. Patel and Grewal [2022] conducted a user study comparing Augmented Reality (AR)-based indoor navigation with traditional 2D maps. Most users found the AR-based method to be mentally and physically **less tiring** and were able to reach their objectives faster. The study noted that a limitation was the need for users to hold a mobile phone in front of

them for navigation support. The authors proposed the use of an AR-headset as a further improvement, which could enhance the user experience by allowing hands-free navigation. Additionally, preceding studies related to this project Morlighem et al. [2020]; Smit et al. [2021] utilized the Unity game engine for visualizing 3D graphics, and the current workflow is optimized around this system. Therefore, decision is made to visualize the path in Unity in a way that can be overlaid onto the real world using the HoloLens 2's MR view. This approach ensures that the path is easily understandable and can be followed by the field agent in real-time, enhancing their navigation efficiency during emergency responses.

4. Methodology

The methodology will be discussed in four distinct subsections: System, Data Acquisition, Data Processing, and Visualization.

System: This section will outline the framework upon which this research builds, detailing both the hardware and software utilized. It will also elaborate on the calibration process for the GeoSLAM Zeb Horizon RT, highlighting the integration of these technologies within the project's infrastructure.

Data Acquisition: This part will delve into the processes involving the three different sensor systems and the environments selected for the study. It will provide a detailed explanation of the rationale behind the choice of sensors and settings, ensuring clarity on the methodological decisions made.

Data Processing: The workflow for achieving path planning will be described here, including the necessary pre-processing steps. While the study primarily aims to utilize point clouds with minimal processing, the mobile laser scanners (MLSs) employed require specific pre-processing due to their unique data outputs. The pre-processing of data from the HoloLens 2, which involves converting RGB+D scenes into point clouds, will be briefly covered. However, detailed discussion on this aspect is limited as it pertains to a commercial project outside the scope of this research.

Visualization: The final subsection will describe the techniques used to transform the path planning output into visual cues. This section aims to ensure that the results are communicated effectively to users, facilitating clear and actionable insights based on the navigation paths generated.

An overview of the methodology is illustrated in Figure 4.1.

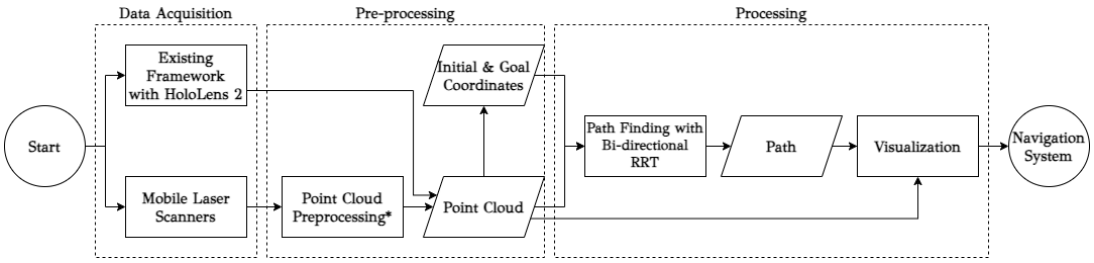


Figure 4.1.: Methodology Overview

4.1. System

The System section of the methodology is divided into three detailed subsections, each addressing a crucial component of the project's technical framework:

Hardware: Detailed information about the sensor systems and computer hardware utilized in the project is presented here. This includes an in-depth description of each sensor's attributes and functionalities. Reasoning behind the hardware choices and aspects that correlate to the results are mentioned in this section as well.

Software: This part lists and describes the software suites and packages employed in the development of the research. It details how these tools are used for handling both the literature review and the data processing tasks, thereby providing insights into the software environment integrated into the project.

Calibration: The final subsection focuses on the practical aspects of integrating the Zeb Vision module with the Zeb Horizon RT. It describes the installation process and the subsequent calibration steps required to ensure that the module functions correctly within the system, aligning with the project's data acquisition and processing needs.

This structured approach to detailing the system components ensures a comprehensive understanding of the technical foundations of the research, facilitating a clearer appreciation of how each element contributes to the overall methodology.

4.1.1. Hardware

This section of the methodology is divided into five subsections, each focusing on a different component of the technology used in this research:

Survey Grade Mobile Laser Scanners: First and third subsections are dedicated to survey grade MLSs employed in this study. These systems are used to demonstrate the capabilities and potential future applications of state-of-the-art sensor technology. The evolution of camera systems—from their early large configurations to the compact sizes available in today's mobile phones—illustrates the significant advancements in sensor technology. Given the recent integration of LiDAR sensors in consumer devices like tablets and mobile phones since 2020 Díaz-Vilariño et al. [2022], these MLSs represent what could feasibly be deployed within the real-time 3D scanning framework in the future.

GeoSLAM Zeb Vision: This subsection details the specific camera module added to the GeoSLAM Zeb Horizon RT, enhancing its data acquisition capabilities.

HoloLens 2: The third sensor system, the HoloLens 2, showcases the capabilities of the existing real-time 3D indoor scanning framework. This low-cost, wearable device streams sensor data in real-time and requires minimal user intervention, making it highly suitable for dynamic scanning environments. However, the limitations of its sensor capabilities, particularly in comparison to the MLSs, are further discussed in Chapter 5.

Development Computer Hardware: The last subsection addresses the computer hardware used in the development phase of the project. It is critical to specify the specifications of this system since one of the key metrics for evaluating the research is processing speed, which is directly influenced by the hardware capabilities.

GeoSLAM Zeb Horizon RT

GeoSLAM Zeb horizon RT is a MLS with the range up to 100 meters. Device utilizes 16 sensors coupled with a rotating scanner head providing 360°x 270°Field of Vision (FOV). In addition, device uses Class 1 / λ903nm laser which is a label for laser systems that cannot exceed optical radiation exposure limits for the human eyes Sliney [1995]. One of the main advantages of this system is that it is capable to generate high density (300.000 points/sec) as well as high accuracy (up to 6mm) point clouds while the user moves at a walking speed. Device allows user to view the point cloud while scanning if combined with a smart phone however, it is not possible to stream the data out without post-processing with GeoSLAM's own software suite. Therefore, Zeb Horizon RT is limited for the static point cloud section of this project. Device is provided by Delft University of Technology in collaboration with Geometius.

GeoSLAM Zeb Vision

Zeb Vision is the camera module that can be added to Zeb Horizon RT. While point clouds share the same specifications with the Horizon RT, Vision module allows Zeb Horizon RT to take two 4K resolution panoramic photographs every second during the scanning process. These photographs are then positioned with the SLAM module of the device and used for colorizing the point cloud. This feature allows user to clear points that can not be colored with the photographs hence eliminating part of the noise caused by reflective surfaces. Zeb Vision module is used to acquire the static and colored point clouds for this study. Device is provided by Delft University of Technology in collaboration with Geometius.

Leica BLK2GO

BLK2GO is another handheld MLS with the range up to 25 meters. Device utilizes 3 cameras for SLAM and colorization of the point cloud acquired by the Light Detection and Ranging (LIDAR) sensor. Uses Class 1 / λ830nm laser. Point measurement rate is 420,000 pts/sec with an accuracy of 3 to 10 mm depending on the scanning environment. Post-scan export of the data is in systems own format (.b2g) therefore, requires its own software suite for pre-processing. System allows the user to view the point cloud during scanning and manufacturer states that streaming the data with an API is possible. Device is provided by CGI in collaboration with Leica Geosystems Nederland.

Microsoft HoloLens 2

Microsoft HoloLens (2) are mixed reality glasses. It is a wearable device that allows user to see both the user interface on the clear lenses as well as their environment through them. Device has a battery with two to three hours operation time. Employs a 1 megapixel Time of Flight (ToF) depth sensor with 120°x 120°FOV, a camera with 8 megapixel stills and 1080 pixel 30 frame per second video capabilities, Inertial Measurement Unit (IMU) consisting of an accelometer, a gyroscope and a magnetometer, four gray-scale light cameras for localization, audio speaker and a microphone in terms of sensors. HoloLens allows user to export and process the sensor data in real-time. Therefore, HoloLens is used for acquisition of the real-time point cloud data for this study. Device is provided by CGI.

Development Computer

Used for conducting research for this study, development of the project, processing and visualization of the data. Device is provided by CGI. Hardware configuration of the device is as follows:

- Processor: Intel i7-10850H CPU @ 2.70GHz
- RAM: 32 GB DDR4
- GPU: NVIDIA Quadro T1000 4 GB DDR6
- OS: Windows 11 Pro

Table 4.1.: Hardware comparison table

Device	GeoSLAM Zeb Horizon RT	Zeb Horizon with Zeb Vision	Leica Geosystems BLK2GO	Microsoft HoloLens 2
Sensor Count	16	18	5	6*
FOV	360°x 270°	360°x 270°	360°x 270°	120°x 120°
Points / second	300,000	300,000	420,000	n/a
Camera Resolution	n/a	4K (3840 x 2160)	4K UHD* (4000x3000)	2K (1920x1080)
Processing	Post processing	Post processing	Post processing	Real-time
Battery Life	3 hours	2 to 3 hours	45 min x 3	2 to 3 hours
Weight	2850 kg	2850+ g*	775 g	566 g

4.1.2. Software

Dedicated software suites used in the project (excluding libraries) are listed in Table 4.2 below.

Table 4.2.: Software

Software	Version	Use
Cloud Compare	2.12.4	Point Cloud Handling
Leica Cyclone REGISTER 360	2023.1.0	BLK2GO output processing
Faro Connect Viewer	2024.1.3	Zeb Horizon output processing
Unity	3.8	Visualization
Visual Studio Code	1.89.1	Development
Zotero	6.0.30	Literature archive and referencing

4.1.3. Zeb Vision Setup and Sensor Calibration

During the process of this research, TU Delft GDMC lab acquired the Zeb Vision module for GeoSLAM Zeb Horizon. This module consists of 2 panoramic cameras responsible from acquiring two images (1 per camera) every second. Images are later used to colorize the point clouds in the post processing workflow. Zeb Vision module is installed on top of the Zeb Horizon scanner with the provided bracket using 4 screws. Module is then connected to the scanner head and to the data logger with provided cables. After installation, firmware update and calibration of the vision module is required to align the Horizon scan data with the Vision module. This process is conducted as follows;

Firmware Update

Initial step is to update the firmware in both scanner head and vision module to ensure they are compatible. For this process, GeoSLAM DataLogger Control Tool used 4.2. Vision module and scanner head (via data logger) are connected to the computer and their respective firmware is updated with the mentioned tool. Flowchart of the workflow is seen in 4.3.

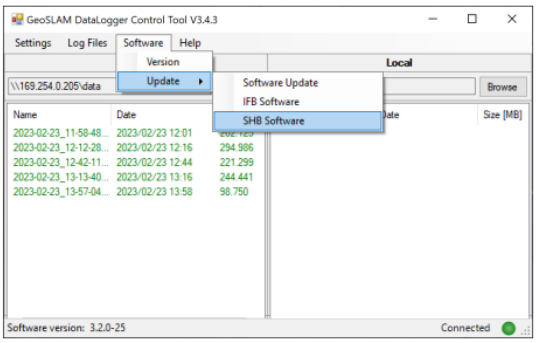


Figure 4.2.: GeoSLAM DataLogger Control Tool V3.4.3

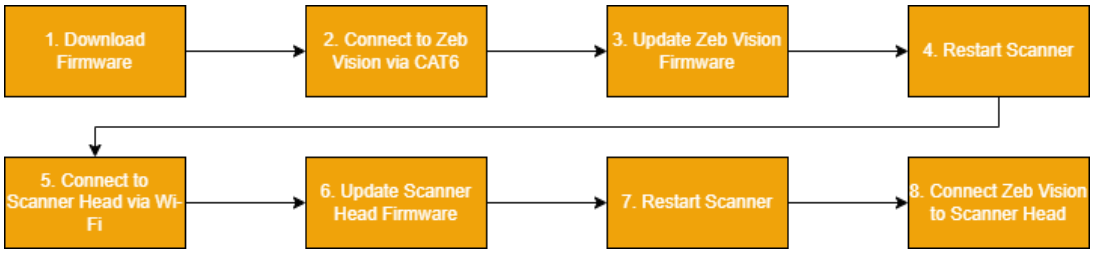


Figure 4.3.: Firmware Update Workflow

Module Calibration

Calibration is the most labor intensive and crucial part of this setup process. Flowchart provided by the manufacturer is seen in Figure 4.4.

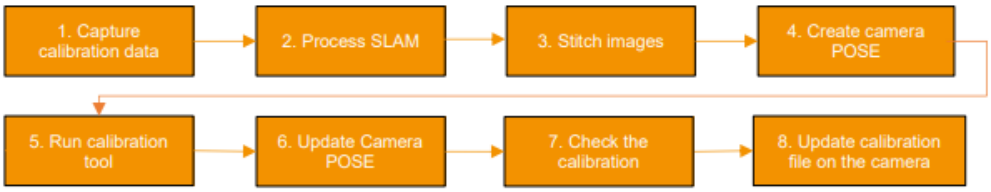


Figure 4.4.: Calibration Flowchart

Data Capture: First and foremost, calibration data is acquired. Manufacturer provided the following requirements for the calibration data accompanied with the following example in Figure 4.5.

- Conducted in an open, outdoor environment.
- There should be at least three easily identifiable features on the ground, e.g., white, or yellow paint markings on asphalt, or corners of paving slabs.
- There should be at least three easily identifiable features at height, e.g., lampposts, road signs, and corners of buildings.
- Space for at least five metres of clearance on either side of the operator.
- Maximum distance to the features is approximately 20m.
- Total capture time is essentially defined by the number of identifiable features in the capture, but in general should be relatively short (5 minutes maximum).

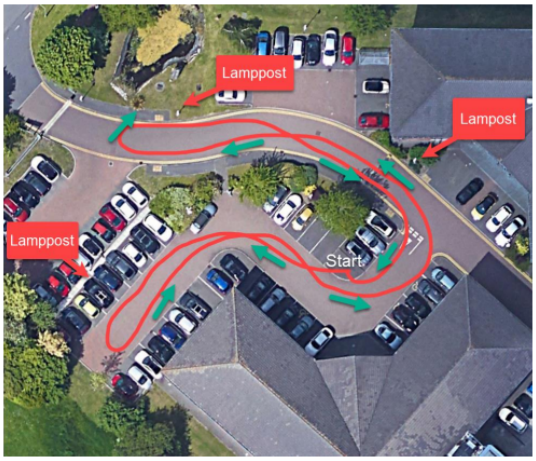


Figure 4.5.: Calibration Data Example

Data acquisition is conducted near TU Delft Faculty of Architecture and the attributes of the dataset is seen in Table 4.3 below accompanied with the resulting point cloud in Figure 4.6.

Pre-processing: Step 2 to 4 in Figure 4.4 are considered as the pre-processing prior to calibration.

Table 4.3.: Calibration Dataset

Points	16642669
Size	333 MB
Duration	2m 57s
Date	2/29/2024

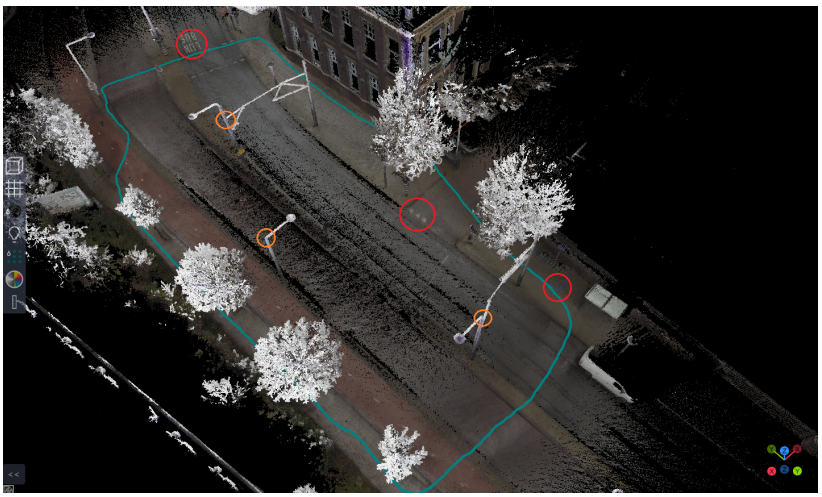


Figure 4.6.: Calibration Data: Ground (red), Height (orange) control points and Trajectory (green).

- Step 2 - Processing SLAM: initial step where .geoslam output of the scanner is processed into a point cloud file and the captured images by the Vision module is exported.
- Step 3 - Image Stitching: stitches the images captured by the cameras into a single panoramic image and produces a timing file which is later used to locate the images in the scan.
- Step 4 - Create Camera Pose: Timing file produced in the previous step is used to locate the images in the scan, generating a pose file that depicts the image capture locations to be used in both calibration and point cloud colorizing.

Calibration: Steps 5 to 8 will be discussed under this subtopic. After the required input files are generated. Camera calibration tool is engaged.

- Step 5 - Start Tool: Tool is started by using the input files: vision output, stitched images, point cloud output, timing file.
- Step 6 - Update Camera POSE: Selecting a control point in the point cloud (mentioned as identifiable points in Data Capture) seen in Figure , matching the selected control point in the images that it appears (Figure 4.7) and repeating the process for 6 control points.
- Step 7 - Calibration Check: Checking if the lens coverage is even (Figure 4.8a) and average distance error is between 4 to 8 cm margin (Figure 4.8b).

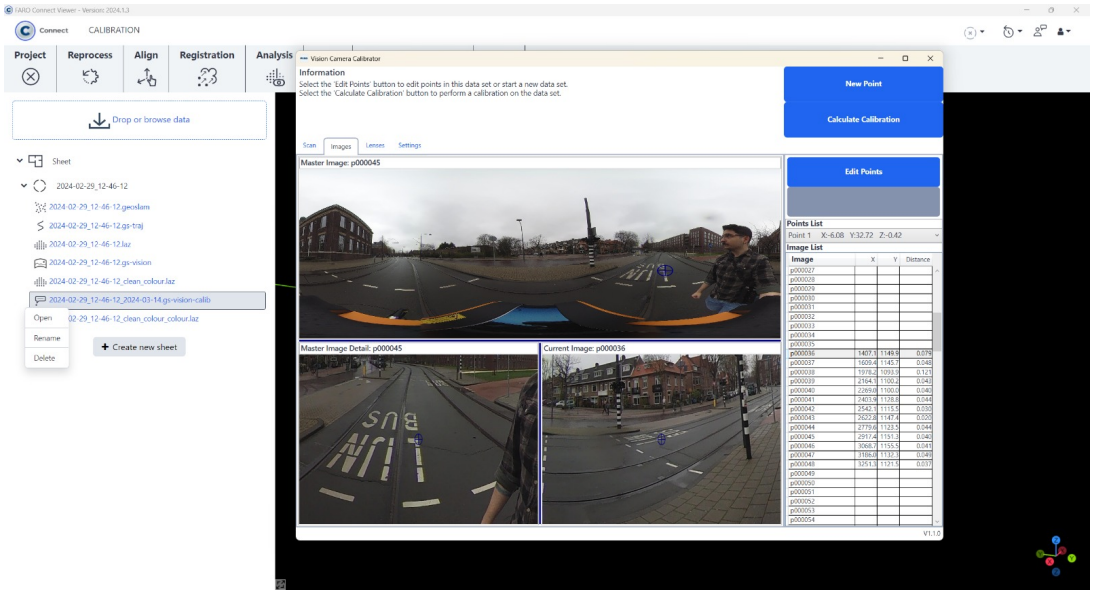
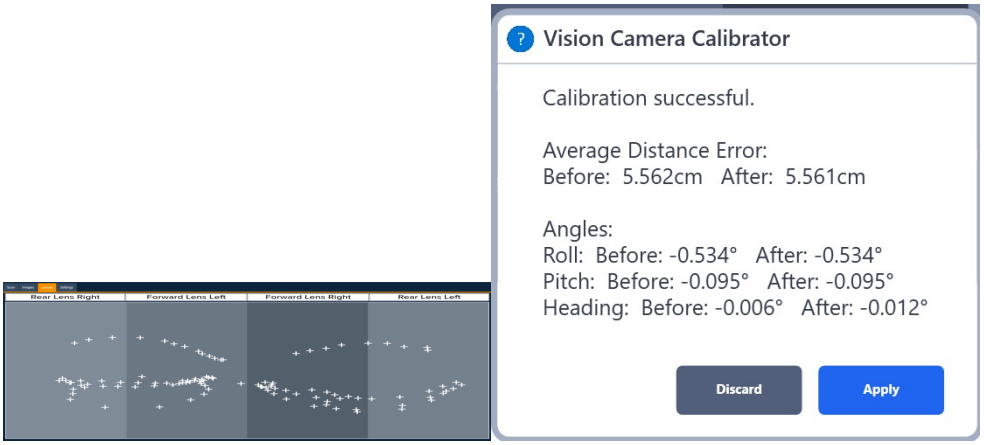


Figure 4.7.: Camera Calibration Tool

- Step 8 - Calibration Update: Generated calibration file is uploaded to the device using the Data Logger Control tool used in Firmware Update.



(a) Lens Coverage (b) Calibration Error

Figure 4.8.: Camera Calibration Results

4.2. Data Acquisition

Data acquisition is planned and executed with controlled variables to facilitate comparisons across different hardware setups and to test the developed methods under various scenarios.

- Data is acquired in the same environment using different hardware to facilitate hardware comparisons.
- Different environments are scanned with the same hardware to compare performance under diverse conditions. This approach also tests the proposed methodology across various scenarios and helps prevent overfitting to a single set of conditions.
- The environments selected for scanning are those that do not require additional permits for data collection and storage.
- Scans are conducted either in locations without pedestrian traffic or when the area is empty to avoid limitations related to the General Data Protection Regulation (GDPR).

Table 4.4 lists the scanned environments and the hardware used, while Table 4.5 shows the dataset attributes for quantitative comparison between sensor outputs. Full-size versions of the datasets are available in Appendix C.

Table 4.4.: Scanned Environments

Device	GeoSLAM Zeb Horizon RT	Zeb Horizon with Zeb Vision	Microsoft HoloLens 2	Leica BLK2GO
CGI 8th floor	x	x	x	
TU Delft BK Basement		x		x
Apartment Complex	x	x		x
TU Delft BK Orange Hall		x		x

4.2.1. Static Point Clouds

Point clouds that cannot be processed in real-time during acquisition are classified as static point clouds. These are acquired using devices such as the Leica Geosystems BLK2GO, GeoSLAM Zeb Horizon RT, or the Zeb Horizon RT equipped with the Zeb Vision module. Since it is not feasible to stream the point clouds in real-time with these scanners, the datasets are considered static and are available prior to routing. These devices are utilized to compare and contrast their outputs with those of the HoloLens 2 and among each other.

Furthermore, these higher-capability sensors illustrate the future possibilities of this navigation methodology as sensor technology continues to advance and become more accessible and affordable. By demonstrating the potential for higher-quality scans, these devices highlight the advantages of improved data fidelity and resolution. To ensure the reliability and accuracy of the data collected, variables related to data acquisition are meticulously controlled, prioritizing data quality and consistency across different scanning sessions.

Timing

Scans are conducted at times of the day when pedestrian traffic is minimal. This approach ensures that the scans are free from disturbances and significantly reduces noise in the data. Additionally, by avoiding pedestrians during the scans, the process becomes more compliant with GDPR regulations, making the use and storage of the data more feasible.

Scanning Path

The GeoSLAM scanning path is planned in a way to ensure features are scanned from multiple angles, and scans are started and ended at the exact same points to allow for loop closure (e.g.seen in Figure 4.9) as recommended by the GeoSLAM user manual. This practice, combined with the SLAM module, improves the positioning accuracy of the scan and enhances drift correction.

In contrast, loop closure is not practiced with the BLK2GO. Instead, the path is planned to avoid walking back and to follow a continuous path from start to finish as much as possible. The reasoning behind this approach is to observe the effects of loop closure, as it is not a feasible practice in an emergency response situation in the context of this application. By avoiding loop closure, we can evaluate the performance and accuracy of the scanning process under more realistic and practical conditions for emergency scenarios.

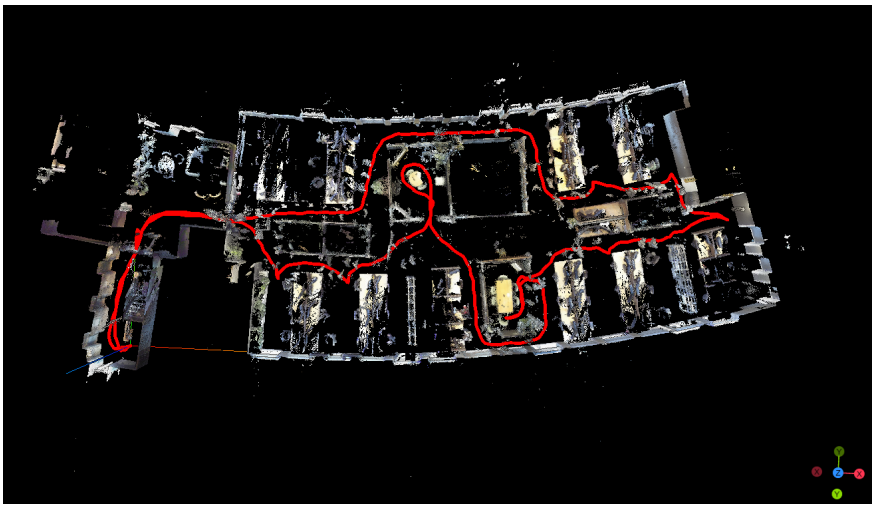


Figure 4.9.: Scanning path for CGI office scan

Scanning Speed

In contrast to the real-time data acquisition with HoloLens 2, the scanning speed with other devices is variable. Some scans are performed at a slow, steady pace, avoiding sudden direction changes to ensure the localization systems provided with the hardware perform optimally. Conversely, other scans are conducted at a walking pace to simulate the potential output if these sensors were integrated into a real-time scanning framework. This variation

in scanning speed allows for a comprehensive evaluation of the hardware's performance under different operational conditions.

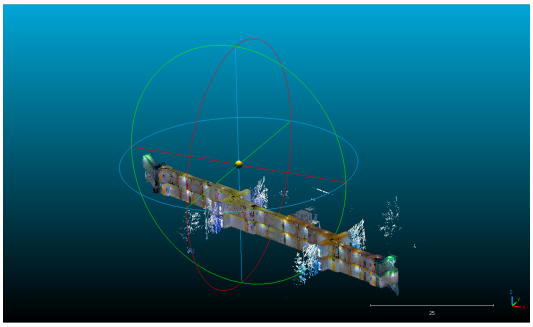
4.2.2. Real-time Point Cloud

As mentioned before, real-time acquisition is performed using Microsoft HoloLens 2 running the system developed by CGI Smit et al. [2021]; Morlighem et al. [2020]. In contrast to MLS, the indoor 3D point cloud is obtained from a combination of data from the RGB cameras and the depth sensor on the device. Sensor data is streamed in real-time to the edge server, where it is processed into the point cloud and co-localized with data streamed from other scanners in the field.

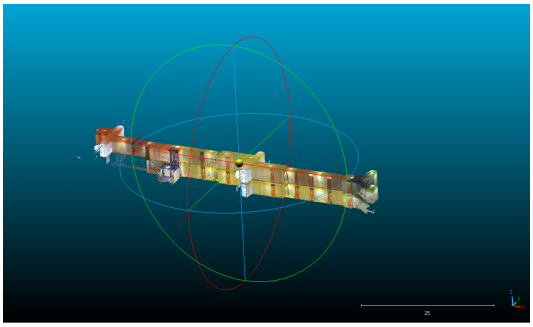
A test run was conducted to acquire an example dataset at the CGI office. During this test, walking speed was not regulated, and there were no practices such as loop closure or controlled movements, ensuring that the output closely resembles the actual use case scenario. This approach aims to simulate real-world conditions and assess the system's performance in a typical operational environment.

Table 4.5.: Dataset Attributes

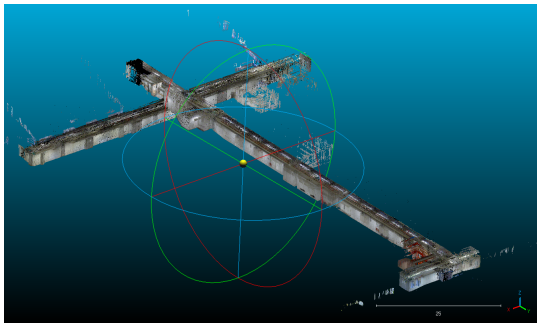
Sensor	Environment	Points (raw)	Points (colored)	Scan Duration	Visual
Zeb Horizon	Apartment	75,481,000	23,057,000	7m 18s	4.10b
Zeb Horizon	Office	47,201,000	14,074,000	4m 7s	4.10h
Zeb Horizon	Basement	65,405,000	26,491,000	5m 31s	4.10d
Zeb Horizon	Hall	23,545,000	5,899,000	4m 4s	4.10f
BLK2GO	Apartment	n/a	91,701,000	6m 42s	4.10a
BLK2GO	Basement	n/a	64,293,000	5m 4s	4.10c
BLK2GO	Hall	n/a	63,943,000	5m 12s	4.10e
HoloLens 2	Office	403,000	n/a	3m 7s	4.10g



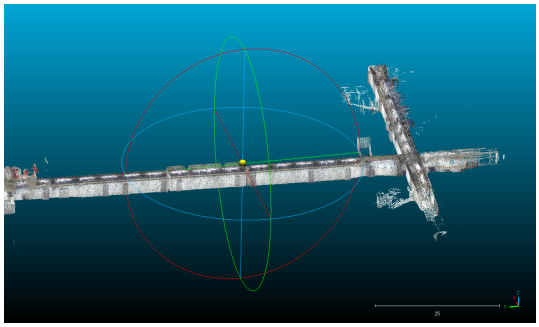
(a) BLK2GO Apartment Complex



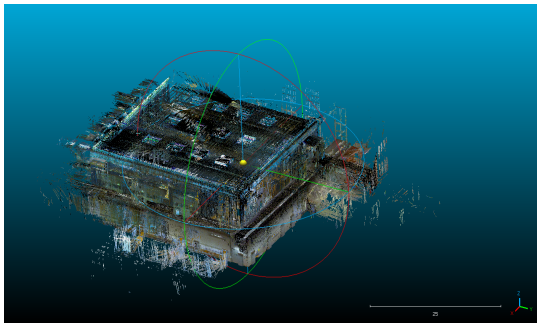
(b) Zeb Horizon Apartment Complex



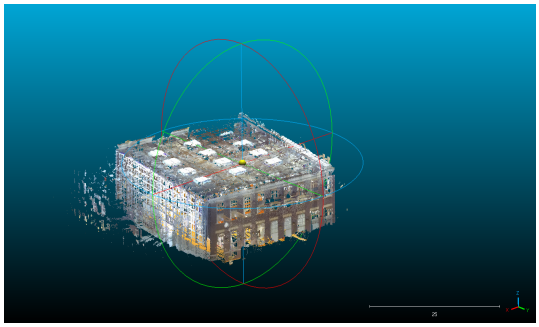
(c) BLK2GO Basement



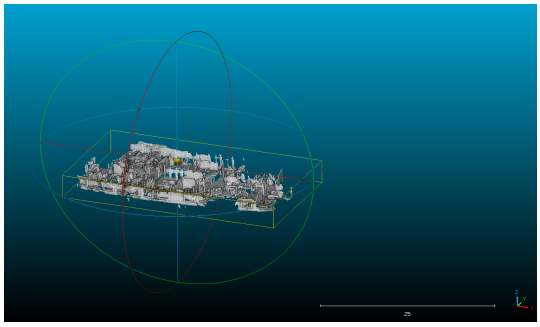
(d) Zeb Horizon Basement



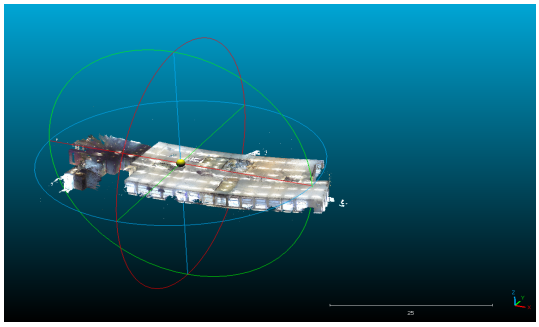
(e) BLK2GO Hall



(f) Zeb Horizon Hall



(g) HoloLens 2 Office



(h) Zeb Horizon Office

Figure 4.10.: Acquired Datasets

4.3. Pre-processing

4.3.1. GeoSLAM Zeb Horizon RT

Zeb horizon RT and Zeb vision come with their own software suite called Faro Connect (previously GeoSLAM Connect). This software suite is mandatory due to the output of the scanner is in .geoslam format which can only be processed by this software. After scanning, output is either exported with the use of a USB drive or downloaded via native wifi connection of the scanner to a computer or a mobile device. Furthermore, data is loaded into the software where SLAM algorithm is used to pre-process it. Data is then seperated into; point cloud, trajectory, image dataset (if vision module is used). Point cloud output of this initial processing can be customized by applying noise reduction and cleaning filters. Cleaning and filtering options are not utilized in this project however, colorizing process of point clouds also apply a degree of filtering. Results of the colorizing process can be seen in Table 4.5. Moreover, Zeb vision datasets share the same features with Horizon datasets with added image processing workflow. Images acquired during the scan are stored in the .geoslam file as well as the pointcloud data. In order to achieve colored point clouds, processes of image localization, image stitching, and point coloring processes need to be done. It should be noted that streaming of the point cloud in real-time during scanning is possible (Figure 4.11) however, available only with GeoSLAM's own interface and it is downsampled version of the actual output. This feature also supports the assumption that these high capability sensors might be available for real-time modelling in future. Scan outputs are exported in .las format prior to processing.

Figure 4.11.: GeoSLAM Zeb Horizon with Streaming Feature

4.3.2. Leica BLK2GO

Similar to Zeb Horizon, output of this scanner is native to its own software suite Cyclone 360. Similar to its counterpart, it is possible to view the downsampled point cloud in real-time during scan with device's own Wi-Fi connection paired with it's own mobile application. However, Leica also provides an API that can be used to view the scan in real-time in other portals. As this feature requires additional development and device was available for a limited period of time, it is not applied. Output of this scanner is a colored point cloud which does not require additional processing to combine RGB data with point cloud. And similarly again, pre-processing allows user to apply noise filtering and cleaning methods. Leica's software suite additionally includes segmentation models trained for various applications. Scan outputs are exported in the .las format prior to processing.

4.3.3. Microsoft HoloLens 2

HoloLens 2 is running on the system developed by CGI. Output of the RGB cameras, IMU and depth sensor is streamed to the edge server with a local connection. Sensor data is then used to generate the 3D model as well as to co-localize and stitch the models from other scanners. These outputs are visualized as a point cloud in the user interface of the system. However, raw output is a database file containing sensor data. In this research this database file is processed into a point cloud file (.ply) and used as is without additional filtering or processing.

Outputs of all scanners are exported into a common point cloud file type (.ply) and used as is without further filtering and processing. This step provides the ”point cloud” input of the methodology

4.3.4. Coordinate Delineation

Second step in pre-processing is deciding on the testing coordinated for the initial and goal locations for the algorithm to find a path between. For this purpose, data is loaded and visualized with Cloud Compare software. Afterwards, cross section tool is used to remove the ceiling and point picking tool is used to select a list of points which then used to delineate coordinates to be used in development and testing processes (Figure 4.12). This methodology guaranteed initial and goal coordinates selected from data is not colliding with obstacles.

In further development, python library open3D is used to implement a function that allows user to select coordinates from the 3D model. User is instructed to pick initial and goal points on the ceiling of the model. Furthermore, vertical coordinates (Z-values) of the selection is reduced by 0.6 meters to descend the point into the model, applied a collision check to control if it collides with an object and used as the input if passes (Figure 4.13). However, this methodology is avoided during testing process and pre-selected coordinates are used.

This step provides the ”initial and goal coordinates” input for the routing methodology

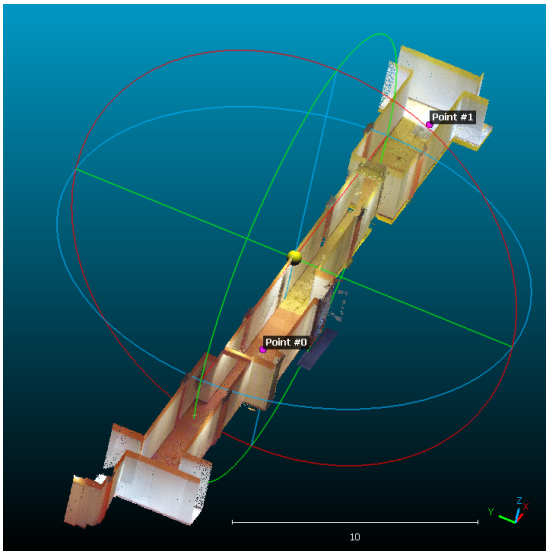


Figure 4.12.: Test Point Example

4.4. Path Planning

Inputs derived from pre-processing and coordinate delineation are the starting point of the processing methodology. In this section these inputs are used to prepare for the path planning algorithm. This section is separated from the pre-processing as it involves a common methodology that is applied to datasets from all three sensors. As the algorithm is written in python, any processing after mandatory software uses are done also in python to deliver a coherent workflow that can later be implemented into the real-time 3D modelling framework. Flowchart of the path planning framework is seen in Figure 4.14.

Development Environment:
Code Editor: Visual Studio Code
Language: Python
Version: 3.11.9
Libraries:

- Open3D: Handles 3D data as well as used for data structures such as KD-Trees and Octrees.
- NumPy: Computing library especially used for arrays and matrices.
- SciPy.spatial: Spatial submodule of SciPy library. Also used for data structures.
- Matplotlib: Plotting library used for visualization of resulting paths in 3D plots.
- time: Built-in library for time measurements used in performance calculations.
- tracemalloc: Built-in library for memory usage tracking. Used for performance assessment.
- tkinter: Build-in user interface toolkit. Used for data loading interface.

Inputs:

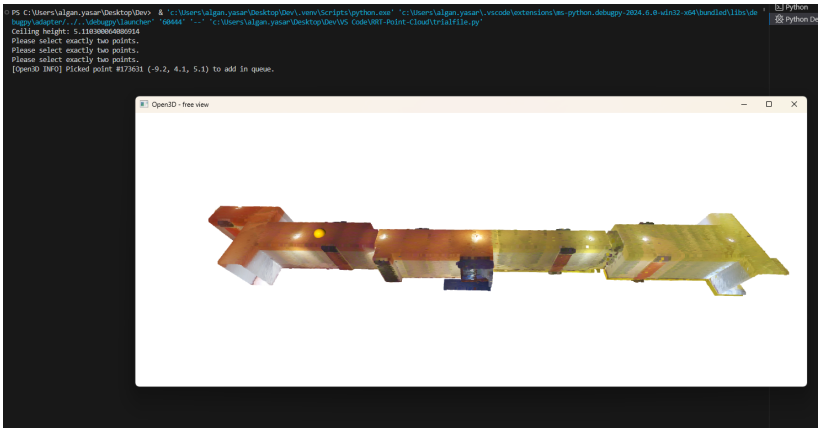


Figure 4.13.: Coordinate Delineation with Open3D

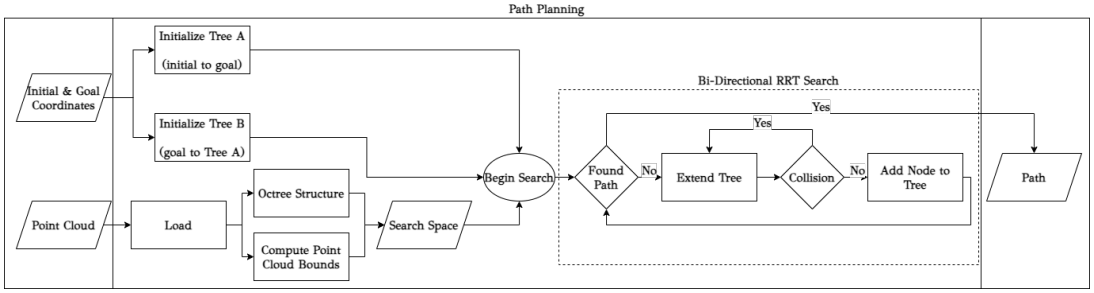


Figure 4.14.: Path Planning Flowchart

- Starting Coordinates: $n_{initial}$
- Goal Coordinates: n_{goal}
- Point Cloud

Classes:

- Node: Represents edge or junction points of the tree branches in the RRT trees. Attributes: Self (x,y,z), Parent(x,y,z).
- Tree: Combination of the Nodes and branches. Attributes: Root (x,y,z), Nodes (list).
- Octree: Data structure used for three dimensional space partition of the point cloud.
- KD-Tree: Data structure used for k-dimensional space partition of the RRT search Trees.

Process:

- Load Data
- Initialize Trees (T_A, T_B)
- Compute Bounds
- Spatial Partitioning (Octree)

- Bi-Directional RRT Search
  - Collision Check
  - Target Bias
  - Step Size
  - Tree Re-structure (Kd-Tree)
  - Path Optimization & Normalization
- Path Export

Outputs:

- Path
- Metrics

4.4.1. Process

4.4.2. Load Data

initial step of the data processing workflow. File path of the desired point cloud data is obtained with the help of tkinter module. This module allows the script to call for a file browser window and select a file to get it's path. Point cloud data is stored in PLY file type. This file type is available in either ASCII or binary file format. This file type stores points in a list format where each point is a list entry that is represented by coordinates. If available these list entries can also involve infirmation such as color, transparency etc. Open3D's load function is used to read this data type.

4.4.3. Initializing Trees

Bi-directional aspect of the RRT algorithm requires two trees instead of one compared to the original RRT algorithm LaValle [1998]. In original methodology, a single tree is initialized at $n_{initial}$ and it randomly explores the search space until there is a connection between $n_{initial}$ and n_{goal}. In bi-directional search there is a second tree initialized at n_{goal}. Therefore, two trees: $T_A(n_{initial})$ and $T_B(n_{goal})$ are initialized.

4.4.4. Compute Bounds

A simple bounding box that is calculated from the extent of the point cloud data is used for limiting the space partition and search space. This is achieved by transforming the loaded point cloud into an array with NumPy library and retrieving the minimum and maximum values across three dimensions.

4.4.5. Spatial Partitioning

One of the most crucial part of this methodology is data structuring. Unstructured nature of the point clouds make them demanding when it comes to processing which is often becomes the bottleneck in the system. Especially when dealing with higher quality datasets obtained with MLSs, it becomes more obvious. In order to address this issue, Octree data structure is used. Octree data structure partitions the three-dimensional space by subdividing it to eight leaf nodes. This process is repeated for every leaf node until the desired level of partitioning (Depth D) is achieved (Figure 4.15) resulting in a tree with 8^D leaf nodes. Depth of the tree in this research is decided as 6 in result of empirical testing as higher D provides faster indexing, yet generating the octree takes exponentially longer. Implementation of this structure is achieved with Open3D python library.

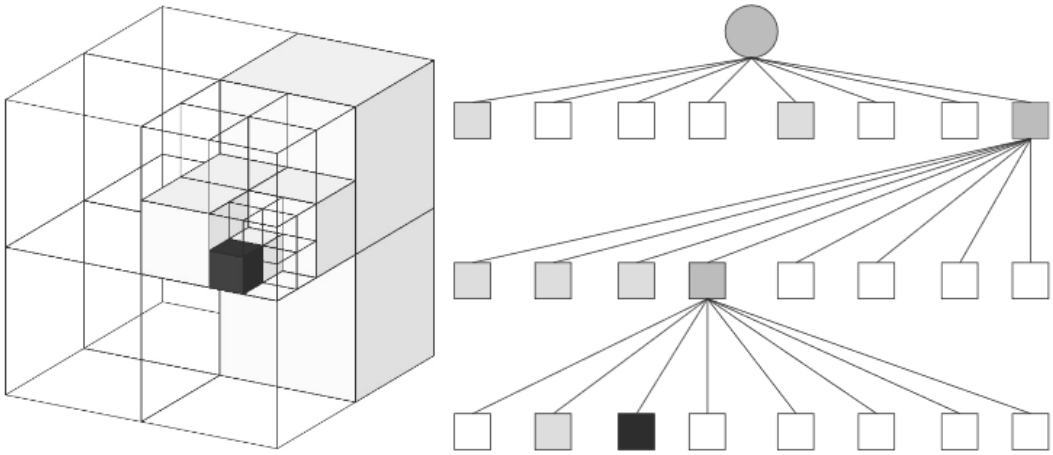


Figure 4.15.: Octree data structure visualization with $D = 3$ from Aernouts et al. [2018]

4.4.6. Bi-directional RRT Search

After the search space is initialized by the preceding processes, iterative search process of the algorithm starts. Generalized structure of the search function is seen in Algorithm 4.1. Algorithm iterates until a path is found or the iteration limit is met. Limit is defined manually during the development to limit total run time. Sub-functions of the search algorithm are explained below:

Collision Check

One of the most crucial functions of the algorithm is collision checking. In most RRT implementations, obstacles are depicted as 'regions,' so the collision check involves determining if the sample coordinates fall within these obstacle limits. However, in this implementation, obstacles are the *points* of the point cloud, representing an indoor environment. Therefore, the aim is to stay 'in' the obstacle. To achieve this, an axis-aligned bounding box that roughly represents the size of an average adult human is used to check if there are 'points' around

the path. Another challenge is defining the threshold for a collision. Since the data was not cleaned or filtered, noise in the scans could result in false collisions. Thus, we calculated the density (ρ) of the obstacles to decide on this threshold. This density is sensor-specific due to their varying point density (Table 4.1) and should be adjusted according to the data source.

The collision check for a node n_{child} is performed using an axis-aligned bounding box (AABB) centered around n_{child}. Let $B(n_{\text{child}})$ denote this bounding box and ρ be the density threshold. The collision check can be described as follows:

$$\text{CollisionFree}(n_{\text{child}}) = \begin{cases} \text{True} & \text{if } \frac{N(B(n_{\text{child}}))}{V(B(n_{\text{child}}))} \leq \rho \\ \text{False} & \text{otherwise} \end{cases}$$

where:

$$N(B(n_{\text{child}})) = \text{number of points inside the bounding box } B(n_{\text{child}})$$
$$V(B(n_{\text{child}})) = \text{volume of the bounding box } B(n_{\text{child}})$$

Target Bias

In standard RRT, sampling is uniformly random across the search space, which can result in inefficiencies in large spaces or narrow openings. A performance-enhancing solution is target-biased sampling. The algorithm still randomly explores but, with a probability of p, the tree is expanded from the $n_{nearest}$ to the n_{target}.

The target node n_{target} is chosen based on the target bias probability p. The process can be described as follows:

$$n_{\text{target}} = \begin{cases} n_{\text{goal}} & \text{if } r < p \\ \text{Random Sample} & \text{otherwise} \end{cases}$$

where r is a random number uniformly distributed between 0 and 1. Effect of the target biased can be observed in the Figure 4.16 below. Majority of the dark blue lines of T_A are headed towards the green lines of T_B due to biased sampling.

Step Size

Defined as the distance between n_{parent} and n_{child}, step size is set as the width of the bounding box used for collision checking. As the algorithm checks for collisions around n_{child}, checks for the second half of the step while, the previous check for n_{parent} ensures the first half of the step is obstacle-free, resulting in a clear path. Bounding boxes and step size is seen in Figure 4.17.

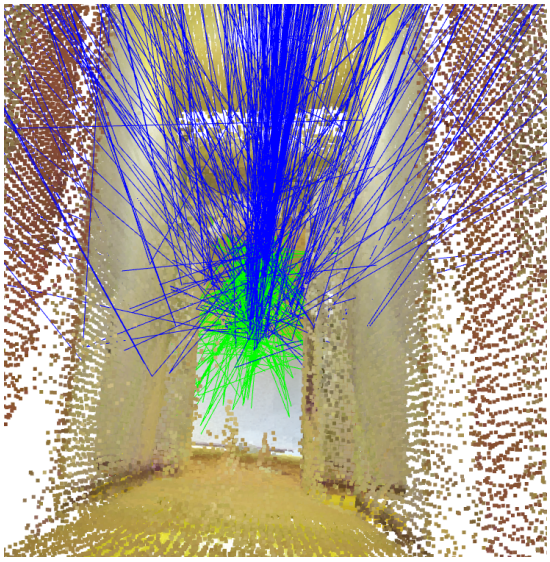


Figure 4.16.: Biased Sampling in Apartment Dataset

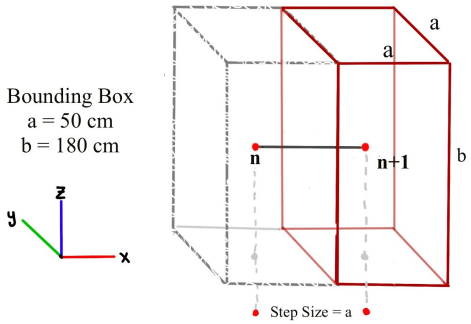


Figure 4.17.: Step size and bounding box

Tree Re-Structure

Similar to spatial partitioning applied to the point cloud, KD-tree is used to index the search trees T_A, T_B. As the trees expand, nearest neighbor search for the sampled points take longer due to high node count in trees. Therefore, a KD-tree is generated for each tree which is updated every 100 iterations with the new nodes, keeping the iteration speeds at feasible levels.

Path Optimization & Normalization

When a connection is established between initial and goal points, rewiring is executed which then checks if there is a path with lower cost. Furthermore, as the path represents a connection between crucial points of collision check bounding boxes, the coordinates are normal-

ized to the floor level by reducing the Z component by 0.7 meters (Figure 4.18). This process is done in order to enhance the visualization in the following steps.

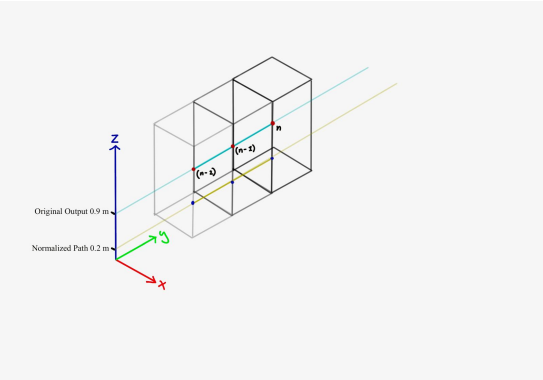


Figure 4.18.: Path Normalization

4.4.7. Path Export

If the search algorithm returns a path, it is a lift of coordinates depicting the nodes between $n_{initial}$ and n_{goal}. This list is adjusted to replace the values in Z axis with the values in Y axis as the visualization is done in the Unity environment where the default vertical axis is Y. And finally, list is exported in a text file (.txt) as the final step in the path planning process.

Algorithm 4.1: Bi-directional RRT with Target Bias

Input: n_initial, n_goal, p
Output: Path from n_initial to n_goal

$T_A \leftarrow$ Tree($n\_initial$);
$T_B \leftarrow$ Tree(n_goal);
$max\_iterations \leftarrow 20000$;
$iterations \leftarrow 0$;
while $iterations < max\_iterations$ **do**
  $iterations \leftarrow iterations + 1$;
  $n_{rand} \leftarrow$ SampleNode(n_goal, p);
  $n_{nearest} \leftarrow$ NearestNode(T_A, n_{rand});
  $n_{new} \leftarrow$ Steer($n_{nearest}, n_{rand}$);
  if *!isCollision*($n_{nearest}$, n_{new}) **then**
    $T_A \leftarrow$ AddNode(T_A, n_{new});
    $T_A \leftarrow$ AddEdge($T_A, n_{nearest}, n_{new}$);
    if *ConnectTrees*(T_A, T_B, n_{new}) **then**
      return ExtractPath(T_A, T_B)
  Swap(T_A, T_B);
  return Failure

Function `SampleNode(`n_goal, p`)`**:**
- **if** *rand()* ¡ p **then**
  - **return** n_goal
- **else**
  - **return** SampleRandomNode()

Function `isCollision(`n_{new}`)`**:**
- $B \leftarrow$ BoundingBox(n_{new}) $N \leftarrow$ NumberOfPointsInside(B) $V \leftarrow$ Volume(B)
- **if** $N/V \leq \rho$ **then**
  - **return** False
- **else**
  - **return** True

Function `ConnectTrees(`T_A, T_B, n_{new}`)`**:**
- **if** $n_{connect} \leftarrow$ *NearestNode*(T_B, n_{new}) **then**
  - $n_{connect\_new} \leftarrow$ Steer$(n_{connect}, n_{new})$ **if** *!isCollision*$(n_{connect}, n_{connect\_new})$ **then**
    - $T_B \leftarrow$ AddNode$(T_B, n_{connect_new})$ $T_B \leftarrow$ AddEdge$(T_B, n_{connect}, n_{connect_new})$ **return** True
- **return** False

Function `ExtractPath(`T_A, T_B`)`**:**
- $path_A \leftarrow$ ExtractPathFromTree(T_A) $path_B \leftarrow$ ExtractPathFromTree(T_B)
- $path \leftarrow$ CombinePaths$(path_A, path_B)$
- **foreach** $node \in path$ **do**
  - $temp \leftarrow node.y$ $node.y \leftarrow node.z$ $node.z \leftarrow temp$
- **return** path

4.5. Path Visualization

4.5.1. Visualization Tools and Software

For the visualization of navigation paths and 3D point clouds, our project leverages the Unity game engine. Specifically, the Pcx point cloud rendering asset within Unity is used to display the point cloud data, while Unity's native line renderer tool, supplemented by custom C# scripts, is employed to visualize paths determined by the path planning algorithm. The choice of Unity as the visualization platform was strategic, aligning with the parent project's existing user interface, which was also developed in Unity. This ensures seamless integration and consistency across project components, facilitating a unified user experience and straightforward implementation. Pseudo-code of the script can be seen in Algorithm 4.2 below.

4.5.2. Integration with the 3D Point Clouds

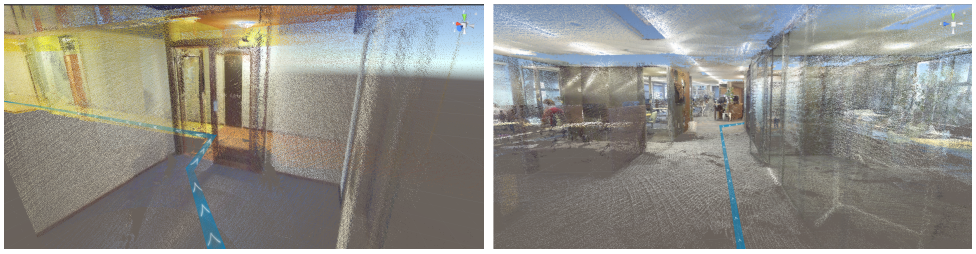
Path visualization is directly integrated with the 3D point clouds, which serve as the foundational dataset for path finding. Since the paths are derived from the same coordinate system as the point clouds, they overlay perfectly without requiring additional alignment processes. This direct usage of raw point cloud data in path finding simplifies the overlay process, eliminating potential complexities that might arise from converting or aligning disparate data formats or coordinate systems.

4.5.3. User Interface and Path Visualization

In the Unity environment, the navigation path is visualized to hover slightly above the ground, enhancing visibility and ensuring it does not blend into the background of the point cloud data. This method was chosen both for the proof of concept within the Unity platform and is intended for real-world implementation in mixed reality settings. The path is rendered in a bright color, providing a stark contrast against various backgrounds to ensure it stands out effectively. Additionally, the path features a texture with small arrows that point towards the destination, offering intuitive directional cues to guide the user along the correct route. Currently, the system does not support user interaction with the visualization, focusing solely on delivering clear and effective navigational guidance. Hand drawn demo visualizations can be seen in Figures 4.19a and 4.19b. Final proof of concept visualization rendered in Unity can be also seen in 5.3.

4.5.4. Accuracy and Clarity of Visualization

To maintain the clarity and accuracy of the visualization, special attention has been given to the aesthetic and functional aspects of the path design. The chosen color and arrow textures are specifically designed to provide easy recognition and directional orientation, essential for navigating complex environments efficiently.



(a) Route demo visual 1 (b) Route demo visual 2

Figure 4.19.: Demo Visuals

4.5.5. Game Scene

The game scene is created to demonstrate the path visualization and its integration with the point cloud data in the Unity environment (Figure 4.20). This method is chosen to provide a proof-of-concept visualization.

The scene is constructed as a 3D environment with a first-person point-of-view control system. The character used in the system has the same dimensions as the bounding box used for collision checks in the RRT search, ensuring accurate representation. The scaling of both the character and the point cloud data replicates real-world sizes.

Key elements in the scene include the point cloud environment, which serves as the backdrop, the path generated by the RRT algorithm as the main focus, and the first-person controller, which allows users to explore the environment. The first-person controller is spawned at the initial node of the path, and the point cloud is positioned at its original coordinates along with the path to ensure correct alignment.

Standard lighting is used to provide clear visibility of the environment and the path. The camera is connected to the first-person controller, allowing users to navigate and explore the scene as if they were the field agent. This perspective provides an immersive experience, making it easier to understand the navigation path in relation to the point cloud environment.

The Line Renderer component is configured with a bright orange color and a chevron-shaped texture to visualize the path. The properties of the Line Renderer are set to have a start and end color of bright orange with white chevrons, a uniform width of 0.1 units, and a tiled texture mode to repeat the chevron pattern.

When the game is started, the user can use game controls to explore the environment and follow the path. This interactive element helps users experience how the navigation support would function in a real-world scenario.

The path visualization script is integrated into the scene by attaching it to an empty GameObject. The script reads the coordinates from the `path.txt` file and uses the Line Renderer to draw the path.

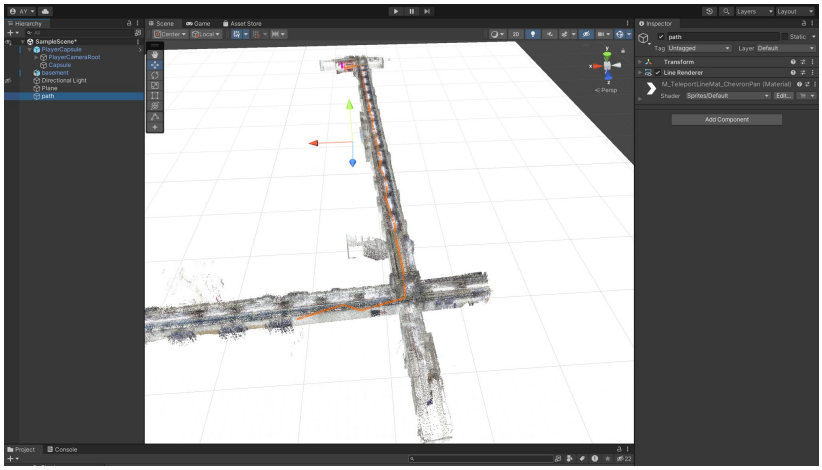


Figure 4.20.: Game Scene

Algorithm 4.2: Path Visualization using Line Renderer

Input: path.txt
Output: Visualized path in Unity

Start:
Initialize Line Renderer component;
Initialize an empty list of path points;
`LoadPath` (path_file);
`RenderPath` ();

Function `LoadPath`(*path.txt*)**:**
    Read path_file line by line;
    foreach *line in path_file* **do**
        Split line into coordinates (x, y, z);
        `AddPoint`((*x*, *y*, *z*));

Function `AddPoint`((*x*, *y*, *z*))**:**
    Create a Vector3 point from (x, y, z);
    Add the point to the list of path points;

Function `RenderPath()`**:**
    Set the number of positions in Line Renderer to the number of path points;
    Set the positions of Line Renderer using the list of path points;
    `SetProperties()`;

Function `SetProperties()`**:**
    Set the material of Line Renderer to a chevron-shaped texture;
    Set the start and end color of Line Renderer to bright orange and white;
    Set the start and end width of Line Renderer to 0.1;
    Set Line Renderer texture mode to Tile;
    Set Line Renderer alignment to TransformZ;

5. Results & Discussion

5.1. Results

This section is structured similarly to the research scope section, with the main difference being that the main research question will be answered in the final section. The sub-questions serve to outline the path leading to the final objective, detailing the opportunities and limitations encountered along the way. Each sub-question addresses specific aspects of the research, providing a comprehensive understanding of the factors influencing the main research question and guiding the overall investigation towards its ultimate goal.

5.1.1. Sub-Question 1

How to navigate to a POI in a static 3D indoor point cloud?

This question can be answered by summarizing the methodology of this research. Static point clouds, defined as datasets acquired with MLSs, provide the high coverage necessary for the proposed approach. These datasets are used as the search space to implement a bi-directional, target-biased, Rapidly Exploring Random Trees (RRT) algorithm, which finds a path between two coordinates within the dataset.

The resulting path, a simple list of coordinates, is then exported into a text file (.txt). This file is subsequently imported into an environment created in the Unity game engine, where the coordinates are translated into a visual path suitable for navigation support. This visual representation enhances users' ability to understand and follow the path within the 3D model, improving their navigation experience.

For this sub-question, performance measures aimed at real-time path planning (e.g., target bias) can be avoided to achieve higher success rates, even if it increases computation time. By focusing on accuracy over speed, the methodology ensures more reliable path finding results within the high-coverage static point cloud datasets.

5.1.2. Sub-Question 2

How to navigate to a POI in a 3D indoor point cloud generated in real-time?

It is not possible to achieve this objective due to sensor limitations mentioned in the discussion section 5.2.1. The current methodology of acquiring real-time point cloud data is done with a HoloLens 2. The resulting dataset from the sensor has lower coverage due to FOV and range limitations . Gaps in the dataset result in the following drawbacks: missing outer structures such as ceilings, floors, and walls are depicted as empty space by the RRT algorithm, leading the algorithm to explore beyond the actual limits of the environment.

Another drawback is un-scanned obstacles in the environment, causing paths that collide with obstacles in reality while appearing feasible in the scan.

To mitigate these issues, a bounding box (Figure 5.1) is introduced to limit the search space. However, the bounding box is a rectangle, while the scan could be in any shape, making it inefficient to limit the search space effectively enough to account for missing structures. Another solution tested was lowering the collision threshold to allow the algorithm to consider semi-scanned obstacles as obstacles, yet this caused noise in the environment to be considered as obstacles, resulting in a failure to find a path.

As mentioned in Sub-question 1, if higher capacity sensor systems become feasible to implement in this modeling framework, the developed methodology can deliver the desired results.

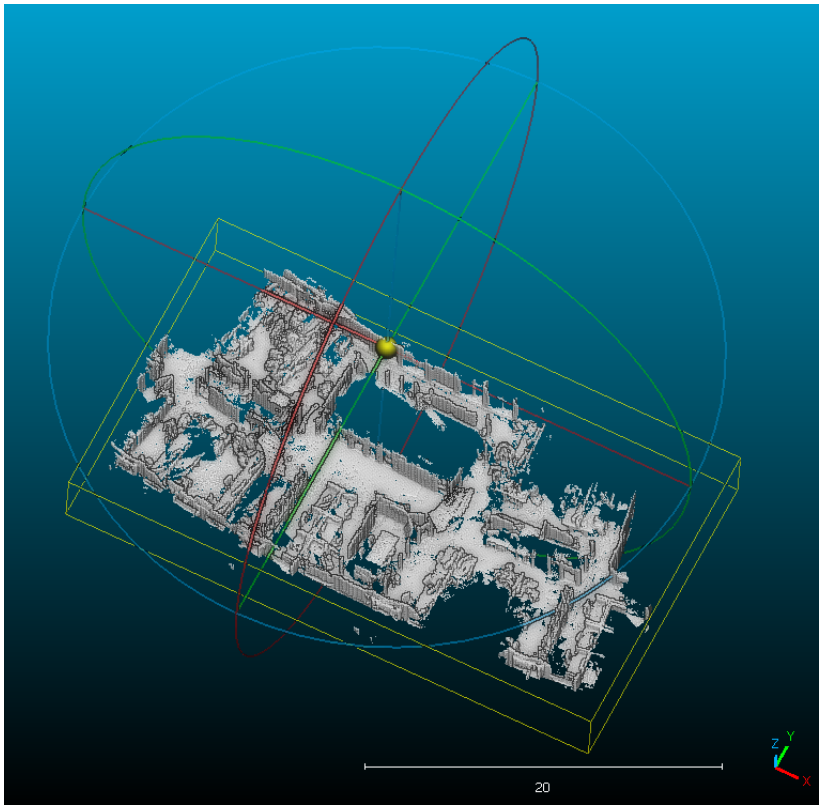


Figure 5.1.: HoloLens 2 Office Environment with Bounding Box

5.1.3. Sub-Question 3

How to communicate POIs between units and provide navigation support to the explorers?

Communication of both POIs and the navigation support with the explorers is done with visual cues. An alternative could be sound-based support, which could be easily suppressed by environmental noise and requires the implementation of a complex framework such as IndoorGML to correctly structure the voice cues. This alternative requires further steps, such

as segmentation and classification of the model, to apply the correct cues. For example, to give the direction "leave the room from the second door on the right," the environment must be segmented into classes like 'room' and 'doors,' as well as constantly tracking the pose and position of the field agent.

In contrast, visual cues directly overlay with the 3D model as they share the same coordinate system. The resulting path is then suitable to be displayed through the MR view of the HoloLens 2, where the field agent will see the path as shown in Figure 5.3. This solution additionally addresses one important aspect of situational awareness mentioned by Smit et al. [2021]. In order to achieve any level of situational awareness, having enough mental resources to comprehend the situation is as important as the availability of the data Endsley et al. [2003]. Therefore, using easy-to-understand visual cues not only enhances the performance of the system but also aids the situational awareness objectives of the overarching project.

5.1.4. Sub-Question 4

Is the solution reliable and feasible for emergency response applications?

The following answer is given in the context of a current 3D modeling system. In this system, the environment is modeled in real-time, ensuring that the available information is always up-to-date. This real-time modeling is crucial because outdated information can lead to inaccuracies and potential safety hazards. Moreover, if a part of the building appears in the scan, it implies that it is safe to traverse, as a field agent (explorer) must have passed through that area to capture the scan. Therefore, when we consider that the path generated by this methodology exclusively utilizes the available scan data, we can conclude that the path should be both safe and reliable. These factors are significantly more important than simply having the shortest path, especially in emergency situations where safety and reliability are paramount.

Regarding feasibility, the primary concern is the processing time required to obtain the path. Given the dynamic and fast-paced nature of the environment, speed becomes the most critical aspect of this methodology. The performance metrics of the system demonstrate its capability to handle the largest test dataset (Basement) in under 12 seconds. This time frame includes all necessary processes: loading the data, building the octree structure, finding the path, and exporting the results. Notably, these tasks were performed on a consumer-grade computer, which underscores the system's efficiency. It is anticipated that further optimizations will enhance the system's performance, particularly when deployed on edge servers, thereby reinforcing the method's feasibility.

- Environment: Basement
- Sensor: GeoSLAM Zeb Horizon RT
- Point Cloud Size: 1,407,430 points
- Path Length: 83 meters
- Average Time per 1000 Iterations: 0.61 seconds
- Total Run Time (octree generation and approx. 15,000 iterations): 11.89 seconds

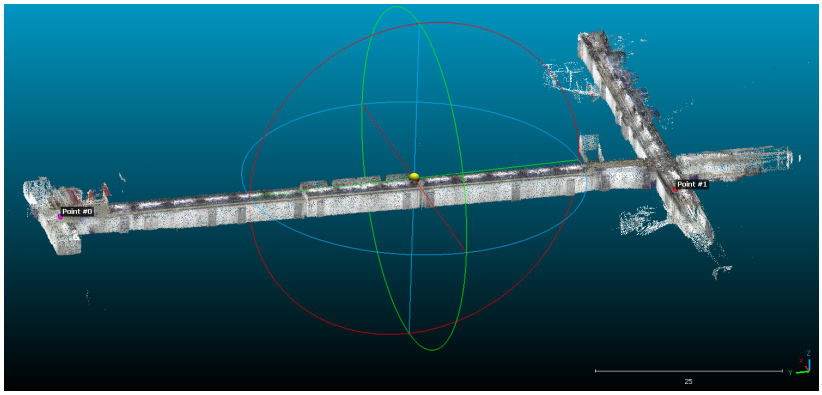


Figure 5.2.: Basement Dataset - Initial: Point 0 / Goal: Point 1

However, it is important to address a significant limitation: the quality of the datasets. The conclusions drawn here are only applicable to high-quality datasets. Due to the current limitations of sensor technology, the 3D modeling system's output may not meet the required standards for this methodology in all scenarios. Consequently, there is a need for advancements in sensor technology to ensure that the 3D modeling system can consistently produce high-quality outputs that are usable for this methodology.

In summary, the real-time 3D modeling system provides a safe and reliable method for path generation, crucial in emergency situations. Its feasibility is supported by efficient processing times, even on consumer-grade hardware, and the potential for further optimization on edge servers. However, the methodology's applicability is contingent on the quality of the datasets, highlighting the need for continued improvements in sensor technology.

5.1.5. Main Research Question

To what extent can we provide navigation support to places of interest using 3D indoor point clouds?

To sum up, it is possible to provide navigation support using 3D indoor point clouds as is. The main requirement for this is the comprehensive coverage of the point clouds. Since the methodology avoids extensive processing of the point clouds, any shortcomings of the dataset are not mitigated. By implementing a robotics path finding algorithm, specifically a bi-directional, target-biased Rapidly Exploring Random Trees (RRT) algorithm, it is possible to avoid the time-consuming and computationally heavy processes typically required for generating navigation graphs, such as segmenting and classifying the navigable space. This approach allows the methodology to utilize an unstructured data type like point clouds efficiently, enabling it to keep pace with a real-time 3D modeling framework. In order to achieve true real-time path planning and navigation, a more capable sensor system that is also suitable for real-time modelling should be integrated into the current system. MLSs used in this study prove that higher coverage point clouds made it possible to achieve path planning however, it is not possible to export the point cloud from MLSs in real-time.

Moreover, as traditional structures are avoided, the output of the method is also unstructured. Therefore, the output must be communicated and presented in a manner that is

comprehensible to the users, who are not robots. Visualization in a 3D environment, such as through the Unity game engine, is used to effectively communicate the results. The coordinates generated by the path finding algorithm are translated into a visual path, making it suitable for navigation support. This visual representation aids users in understanding and following the path within the 3D model, enhancing their navigation experience (Figure 5.3).

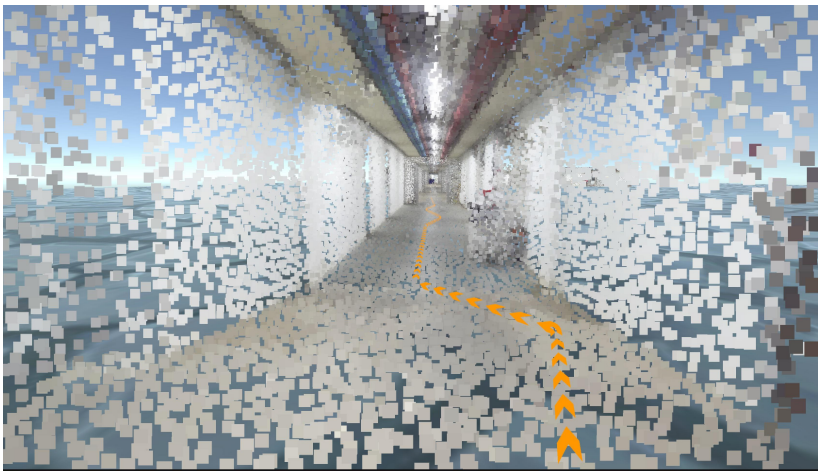


Figure 5.3.: Path visuals overlaying the basement dataset

5.2. Discussion

5.2.1. Challenges

Sensor Limitations

On one hand, it is feasible to generate a point cloud in real-time using current technology. However, this real-time point cloud often contains gaps due to the limitations of the FOV, which may leave certain areas unscanned. Additionally, obstacles that are not fully captured by the scan might be incorrectly identified as viable openings, leading to potential inaccuracies in the model. These issues can compromise the reliability of the real-time point cloud data.

On the other hand, there are sensors available that can rapidly scan the environment with significantly higher coverage and precision. These sensors produce datasets that are ideal for this methodology, as they provide a more comprehensive and accurate representation of the environment. However, a notable limitation of these high-coverage sensors is the necessity for pre-processing the data before it can be used. This pre-processing step can introduce delays, making it less suitable for applications that require immediate real-time data.

A comparison of the same environment scanned with different devices is illustrated below (Figure 5.4). This figure highlights the discrepancies in coverage and detail between the two scanning methods. For instance, due to the field of vision limitations inherent in the

HoloLens, the ceiling is not captured in its scan. This omission demonstrates one of the practical limitations of using such a device for comprehensive 3D modeling.

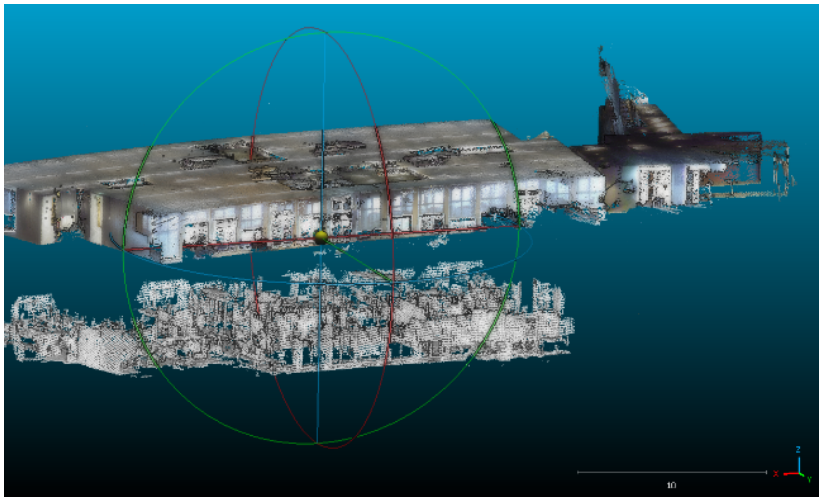


Figure 5.4.: Office Environment with MLS (top) and HoloLens 2 (bottom)

Step Size

As mentioned in the methodology, the step size is set to 0.5 meters to accommodate the bounding box for collision checks. This step size is effective for ensuring accurate collision detection. However, a small step size negatively affects performance in large areas, such as the basement dataset, because it increases the number of calculations required.

Variable step size implementations have proven useful in such scenarios (Zhang et al. [2019]). They adjust the step size based on the environment, allowing for larger steps in open areas and smaller steps in confined spaces. This adaptation can enhance performance by reducing the computational load.

However, using a variable step size causes the collision check method to fail (Figure 5.5). The current collision detection algorithm is designed for a fixed step size, and changing this parameter disrupts its ability to consistently detect collisions. Therefore, while variable step size methods offer performance improvements, they require a redesign of the collision detection process to maintain reliability.

Target Bias

Another method to overcome the challenges of large areas is by introducing a target bias in the path planning algorithm. This method directs the algorithm toward the goal, improving performance in simpler layouts by reducing the time needed to find a path. However, this target bias also introduces a failure probability in more complex layouts.

As shown in Figure 5.6, the algorithm with target bias quickly advanced towards the objective. Despite this rapid progress, it ultimately failed because the biased trees iterated toward opposite sides of a wall and could not connect. The red and yellow dots in the figure

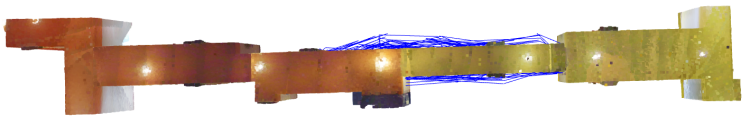


Figure 5.5.: Variable Step Size Failure

represent nodes that are closest to each other, only 0.2 meters apart, yet separated by an obstacle.

This issue becomes especially pronounced in scenarios with multiple floors or intricate structures. In such environments, the target bias can mislead the algorithm, causing it to focus on unreachable areas and resulting in incomplete or inefficient paths. While the target bias enhances performance in straightforward scenarios, its drawbacks must be carefully managed in more complex settings to ensure reliable pathfinding.

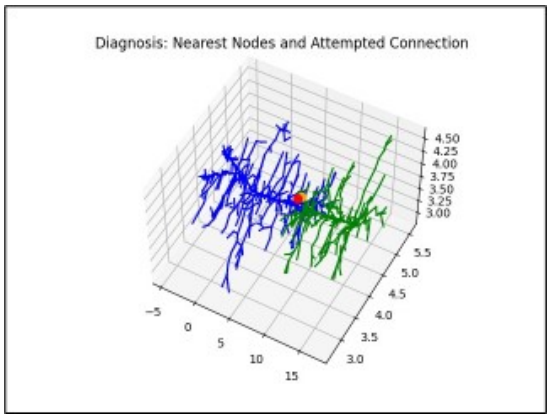


Figure 5.6.: Failure due to biased sampling

Drift

The existence of drift does not affect the methodology used in this research, though it is important to mention it. As discussed in the data acquisition section, some scans are performed with loop closure practice, while others are not. In emergency response applications, scan quality and loop closure are not expected to be considered due to the urgency of the situation.

From a methodological standpoint, since the path planning is executed over the point cloud data and the resulting path shares the same coordinate system, the visualized path will be correct. This accuracy is ensured because the field agent is also positioned within the same dataset by the system.

Looking ahead, georeferencing the model could provide additional value, such as comparing the scan with 3D city models to identify unscanned areas in a building. However, drift could cause issues in these implementations, potentially leading to inaccuracies. As illustrated in Figure 5.7, drift can be observed between datasets of the same environment acquired with or without loop closure.

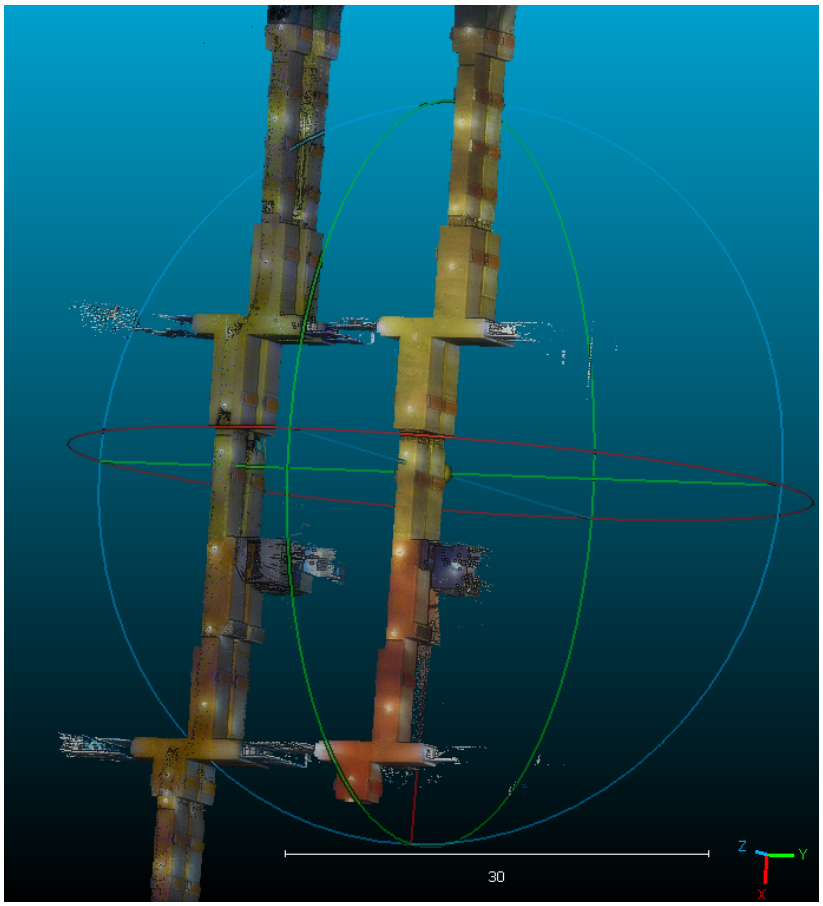


Figure 5.7.: Drift Visual - Without loop closure (Left) / With Loop closure (Right)

Path Smoothing

One of the primary challenges faced in visualizing the paths generated by the Rapidly Exploring Random Trees (RRT) algorithm stems from its inherent nature. The RRT algorithm uses a random sampling method, which often results in paths that are not the most direct or smoothest. While post-processing techniques such as path smoothing could theoretically produce a more visually appealing and practical route, they also risk introducing errors, such as path collision with obstacles. Consequently, smoothing adjustments were avoided to maintain the integrity and safety of the navigation paths, prioritizing directness and collision avoidance over aesthetic smoothness.

5.2.2. Future Works

Parallel Processing / Multi-threading

Processing the search from both trees simultaneously, instead of the current sequential methodology, could yield significant improvements in performance. This parallel processing approach could expedite the path finding process by allowing both trees to advance concurrently toward their objectives.

However, this implementation would require substantial changes to the existing system. Simultaneous tree expansion means that tree T_B would not be able to consistently advance toward the latest node of tree T_A. This discrepancy introduces additional challenges in ensuring that both trees are effectively working towards a common goal. Moreover, the connection check between the trees would necessitate a higher level of complexity, as it would involve continuously monitoring and aligning the progress of both trees to identify viable connection points.

Despite these challenges, the potential benefits of simultaneous tree expansion make it a promising area for further research and development. By addressing the increased complexity and ensuring synchronized progress between the trees, this approach could significantly enhance the efficiency and effectiveness of the path finding algorithm.

Continuous and Variable Collision Check

In this research, the bounding box and hence, the collision check is limited to a large rectangular box. In a scenario where there is an opening that only allows the user to crawl through, our method will fail. However, using a smaller bounding box that represents the minimum space required to pass could result in an unfeasible path (e.g., a path that requires crawling under tables when there is an opening large enough to walk next to it). Implementing a variable method that changes the size after a certain number of failed iterations could solve this problem.

In addition, a method that can constantly check around the proposed path instead of bound-based controls (without hindering the performance) can facilitate variable step size and provide a performance increase. By dynamically adjusting the bounding box size and employing a continuous path-checking mechanism, the system could navigate through varying spatial constraints more effectively, balancing feasibility and performance.

Visualization Improvements

Implementing methodologies such as Gaussian splatting to the point clouds could result in a visual that is much easier to comprehend for the observers (commanding agents) compared to the unstructured and semi-transparent look of the point clouds. Current state-of-the-art techniques deliver photo realistic 3D models; however, achieving this level of detail is a computationally expensive process, which could hinder the performance objectives of this research.

In addition, visualization improvements can be implemented for the path rendering as well. Smoothing the line by adjusting the path planning algorithm and implementing animated arrows along the path can enhance the effectiveness of the visual cues. These enhancements

can make the path more intuitive and easier to follow, thereby improving the overall usability of the system.

Dual Bounding Box

As discussed in Sub-question 2, point cloud output of the real-time modelling framework has low coverage especially further away from the trajectory. Implementing a second bounding box with a similar methodology to collision check to control the minimum point density and only expanding in regions over the limit could allow algorithm to establish feasible paths. However, this theory could be computationally expensive and does not mitigate the missing structural elements (e.g. ceiling)

A. Appendix

The extended abstract of the research paper provided in this chapter has been submitted to the 19th International 3D GeoInfo Conference 2024 in Vigo, Spain. The research will be published in the ISPRS Archives as part of the conference proceedings in July 2024.

A. Appendix

Direct Use of Indoor Point Clouds for Path Planning and Navigation Exploration in Emergency Situations

Algan Mert Yasar[1,2] , Robert Voûte[1, 2] , Edward Vebree[2]

[1] CGI Nederland B.V. George Hintzenweg 89, 3068 AX Rotterdam, The Netherlands
algan.yasar@cgi.com, robert.voute@cgi.com

[2] Delft University of Technology, Faculty of Architecture and the Built Environment, Architectural Engineering and Technology, Digital Technologies 2628 BL Delft, The Netherlands
a.m.yasar-1@student.tudelft.nl, r.voute@tudelft.nl, e.verbree@tudelft.nl

Keywords: Indoor navigation, Path planning, Game engine, RRT, Real-time, Point cloud.

Abstract

This study investigates the feasibility of directly utilizing 3D indoor point clouds for real-time indoor navigation, particularly to enhance emergency response processes. Traditional indoor navigation research primarily focuses on creating navigation systems from pre-existing indoor models, resulting in a graph representation that simplifies spatial relationships, requires post-processing, and delivers results only afterwards, often overlooking real-time obstacles and complex layouts such as those in modern office floors. This research proposes an original approach by leveraging real-time generated 3D models using HoloLens 2 sensors, which combine RGB images and depth sensor output to create a comprehensive point cloud. The study explores path planning directly within these point clouds without the need for extensive preprocessing or segmentation, aiming to provide immediate navigation support with minimal delay. Utilizing the Rapidly Exploring Random Trees (RRT) algorithm, the research seeks to minimize preprocessing and swiftly visualize navigable paths, evaluating the system's performance in terms of processing time and path viability. This approach addresses the limitations of traditional graph-based methods and the challenges posed by outdated or unavailable indoor models, offering a promising solution for real-time emergency navigation assistance.

1. Introduction

Emergency response scenarios demand urgency and precision, as the nature of such events often involves saving lives and minimizing damage (Kapucu and Garayev, 2011). Navigating an unfamiliar location such as a high-rise building without prior knowledge, can be a significant challenge for emergency responders who need to quickly locate resources like fire extinguishers, safe spots, and alternative exits. Even a few seconds saved through an improved navigation system could have a substantial impact on the outcome of an emergency. Current indoor navigation systems primarily rely on satellite signals (such as GNSS) and local Wi-Fi or Bluetooth signals for positioning, which might offer great accuracy under normal circumstances. However, these systems often fail in emergency scenarios where power outages or structural damage can disrupt local networks and block satellite connections. Furthermore, these systems typically depend on pre-existing building data like floor plans or Building Information Models (BIM), which may be outdated or poorly maintained (Nikoohemat et al., 2020), leading to inaccurate and potentially dangerous guidance during emergencies. Most indoor navigation solutions represent the environment in 2D and simplify the spatial complexity into nodes and connecting lines, which is efficient for everyday use but inadequate for the intricate layouts and unexpected obstacles found in emergency scenarios (Boguslawski et al., 2022).

This study aims to leverage the capabilities of an existing system (Morlighem et al., 2020; Smit et al., 2021; van Schendel, 2022) where 3D indoor point clouds are generated and expanded in real-time as field agents navigate through the environment. We utilize visual HoloLens 2's internal SLAM system for positioning, which does not depend on external connectivity other than local network connections provided with the system for data transfer.

This research emerged from an effort to find a methodology that could utilize available 3D point cloud data with minimal processing to keep up with the real-time data influx. An unconventional approach was adopted, using a methodology popular in the robotics industry that focuses on speed and iterability. Additionally, the study uses data obtained from three different scanners: one is the Microsoft HoloLens 2, used to evaluate what could be achieved with the current system, and the other two are survey-grade mobile laser scanners (MLSs), namely GeoSLAM ZEB Horizon RT and Leica BLK2GO. Even though their workflows do not fit with the real-time mapping system, they are employed in this research to see what is achievable if sensors with higher point density and wider fields of view become available.

As sensor technology continues to advance rapidly, the potential for integrating higher-quality sensors into real-time navigation system becomes increasingly feasible. Notable developments, such as the introduction of LiDAR sensors in consumer devices like Apple's mobile phones and tablets starting in 2020, highlight the trend towards more accessible and powerful sensing technologies (Díaz-Vilariño et al., 2022). The recent release of Apple's Vision Pro during the course of this research further underscores the progress in wearable sensor technology. These advancements suggest that features typically reserved for high-end mobile laser scanners could soon be available in smaller, more affordable, and wearable formats.

Employing the modified bi-directional Rapidly Exploring Random Trees (RRT) algorithm for pathfinding, this method treats the points in the point cloud as obstacles, using any available empty space to establish a path from the start to the endpoint. The RRT algorithm is chosen for its efficiency in rapidly processing and adapting to new environmental data, crucial for navigating dynamically changing indoor environments during emergencies.

Unlike traditional navigation systems that simplify environmental data into 2D maps, this solution maintains the three-dimensional complexity of the environment. This approach

allows for effective navigation through and around physical obstacles within complex indoor settings, which is essential in emergency scenarios where environments are prone to sudden alterations such as obstructions from debris or changes in accessible paths.

For example, in emergency scenarios requiring navigation through complex and altered indoor layouts, traditional graph-based navigation systems might not swiftly adapt to such changes. These systems often require time-consuming reprocessing of environmental data to reflect new conditions. In contrast, the use of a real-time 3D model in this research enables the immediate adaptation of navigation paths to the current state of the environment, providing responders with up-to-date and reliable routing.

The proposed navigation system can swiftly adapt to the dynamically changing environment of an emergency. By enhancing decision-making processes and providing efficient paths to safety or to other agents in distress, the system optimizes the use of real-time data with minimal processing.

This navigation solution thus offers an alternative to conventional methods by leveraging advanced scanning technology and pathfinding algorithms, facilitating more effective and reliable navigation in complex indoor environments during emergency responses.

2. Related Works

Indoor navigation is often considered under few main topics: indoor positioning, and pathfinding, mapping (modelling), and communication of the found path. The indoor positioning aspect of this project has been addressed by preceding studies (Morlighem et al., 2020; Smit et al., 2021; van Schendel, 2022). Therefore, our study will focus on the latter. This section follows the natural progression of indoor navigation: first, modelling; second, pathfinding; and lastly, navigation support/communication. Each of these subsections will be considered from the perspective of emergency response.

2.1 3D Indoor Mapping Techniques

The importance of accuracy in indoor modelling has been previously emphasized; there is a need for an up-to-date model, whether it be a floor plan, BIM, or 3D model. An up-to-date 2D floor plan would still represent a complex office floor as a single room. Therefore, the requirement is an up-to-date, 3D indoor model. Building Information Models (BIMs) could be an option. However, a case study by the European Commission (Carbonari et al., 2020) indicated that out of 21 building logbook initiatives, only 8 require updates after renovations, and only 3 are digitalized and accessible. Consequently, even if an up-to-date 3D model exists, it might not be available. To summarize, 2D models are inadequate for representing complexity, and existing 3D models could be inaccessible or outdated. Therefore, real-time 3D reconstruction appears to be the most viable option for emergency response scenarios.

2.2 3D Path Planning

Using a 3D model for navigation is often achieved with a navigation graph. This method requires segmenting and classifying the 3D model to extract the navigable surface of the floor plane. An example is the work of Balado et al., (2019), where they use a 3D point cloud to calculate the route. While this is a robust methodology, it takes computational time. Flikweert et al., (2019) proposed a methodology that automates the process from point clouds to navigation graphs, which can perform even better with high-capacity hardware. However, simplifying the obstacles and navigable surface into a 2D representation is not feasible in a dynamic environment such as emergency response.

Another approach by Broersen et al., (2016) proposes extracting the 3D empty space and structuring it into a graph to achieve pathfinding. This also requires time and processing, but their implementation of an octree approach allows them to quickly index their graph, even though it is a 3D structure. Lastly, the work of Fichtner et al., (2018) combines the use of octree and navigation graphs to streamline the process even more. However, these methodologies would require reprocessing as the dataset expands in real-time and depend on a secondary product. Therefore, we sought a method that could work with an unstructured, unprocessed point cloud.

Path planning methodologies for humans, in contrast to robotics, require much more structure and classification to be understood by the end-user, which does not fit with our objective. Therefore, we adopted a methodology that emerged in the field of robotics: Rapidly Exploring Random Trees (LaValle, 1998). This simple algorithm is not computationally demanding yet effective. In addition to the base algorithm there are various studies that proposed improvements such as bi-directional search, target oriented exploration and adaptive extension (Wang et al., 2022; Zheng et al., 2022).

2.3 Visualization and Path Communication

The final part of this section involves the visualization and communication of the path. We use the term "communication" because the output of our pathfinding methodology is a list of coordinates, which might be adequate for an autonomous drone but not for a field agent. Our end user, a field agent, has limited time and resources to comprehend complex data. Patel and Grewal, (2022) conducted a user study comparing augmented reality (AR)-based indoor navigation with traditional 2D maps. Most users found the AR-based method to be less mentally and physically tiring and were able to reach their objectives faster. The study noted that a limitation was the need for users to hold a mobile phone in front of them for navigation support. The authors proposed the use of an AR-headset as a further improvement, which could enhance the user experience by allowing hands-free navigation. Additionally, preceding studies related to this project (Morlighem et al., 2020; Smit et al., 2021) utilized the Unity game engine for visualizing 3D graphics, and the current workflow is optimized around this system. Therefore, we decided to visualize the path in Unity in a way that can be overlaid onto the real world using the HoloLens's mixed reality view. This approach ensures that the path is easily understandable and can be followed by the field agent in real-time, enhancing their navigation efficiency during emergency responses.

3. Methodology

3.1 System Overview

Diagram and overall explanation of the system is seen in Figure 1 below. Point cloud processing and coordinate picking actions are only applicable for the proof of concept and are due to the limitations of mobile laser scanners and the development environment. The 3D modelling system that this project is designed for can provide point clouds as well as the location of field agents (initial coordinates) and allows for the selection of goal coordinates. Therefore, the pre-processing section will not be required.

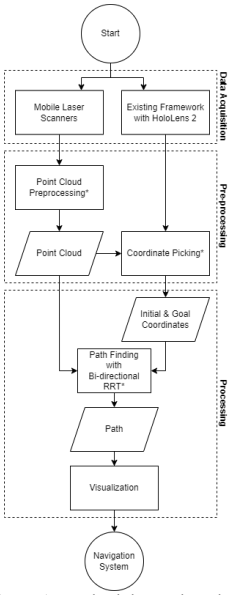


Figure 1. Methodology Flowchart

3.2 Data Collection Equipment and Environments

Our study employed the HoloLens 2 and two mobile laser scanners, the GeoSLAM Zeb Horizon RT and Leica BLK2GO, across three environments: an office building, an apartment complex, and a basement. The office environment was selected for its complex layout and reflective surfaces like glass and monitors, which pose challenges for scanning accuracy. The apartment complex and basement, chosen for their simpler structures and minimal pedestrian traffic, facilitated a focused evaluation of the system's routing and visualization capabilities under controlled conditions. All environments were scanned with the appropriate permissions to comply with data privacy regulations, ensuring GDPR (General Data Protection Regulation) compliance.

3.3 Scanning Workflows and Data Preprocessing

The HoloLens 2 was used to represent real-time 3D modelling system, capturing RGB and Depth images along with the scanner's pose. This data was stored in an SQL database and processed at the edge server into point clouds, though specific details remain outside the scope of this document due to proprietary restrictions. The mobile laser scanners required initial preprocessing using their respective software suites to convert proprietary data formats into standard files (.ply, .las), necessary for subsequent visualization and analysis. This preprocessing included noise filtering and SLAM processing, streamlined to mirror potential future scenarios where more advanced sensors might be used without extensive manual intervention.
Furthermore, preprocessing included coordinate picking prior to path finding. This is an additional step required in the development process in contrary to the 3D modelling framework where initial location will be the field agent's location and goal location will be determined from the system interface (Figure 2).

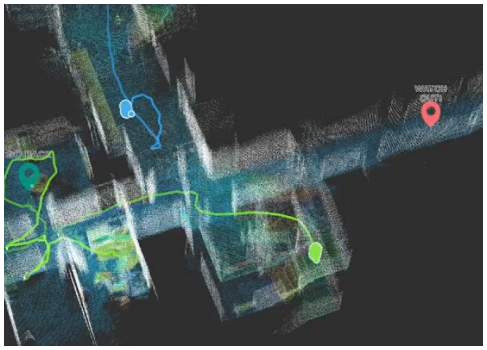


Figure 2. POI and Field agent view from 3D modeling system

3.4 Data Handling and Processing

While this research did not focus on sensor fusion, it emphasized the individual capabilities and data processing workflows of each scanner type. The HoloLens 2 and mobile scanners operated independently, capturing distinct datasets that were used to evaluate the system's performance across different technological capabilities and environmental complexities.

3.4.1 Data Structuring: As mentioned in previous chapters, high computational demand of 3D point clouds is often a bottleneck. In this study we addressed this in a similar method to Fichtner et al., (2018) and Broersen et al., (2016). An octree was used to structure the search space, enhancing the steering and collision checks, while the search trees of the RRT algorithm were structured with a KD-tree to achieve feasible iteration speeds.

3.4.2 Pathfinding and Route Planning: Pathfinding was conducted using a modified bi-directional Rapidly Exploring Random Trees (RRT) algorithm. Inputs included initial and goal coordinates, accompanied by the point cloud file either directly exported from the 3D modelling system or acquired with mobile laser scanners and pre-processed. The following sections explain the functions involved in path planning. The generalized process is illustrated in the flowchart below (Figure 6).

3.4.3 Collision Check: One of the most crucial functions of the algorithm is collision checking. In most RRT implementations, obstacles are depicted as 'regions,' so the collision check involves determining if the sample coordinates fall within these obstacle limits. However, in this implementation, obstacles are the points of the point cloud, representing an indoor environment. Therefore, the aim is to stay 'in' the obstacle.
To achieve this, an axis-aligned bounding box that roughly represents the size of an average adult human is used to check if there are 'points' in the path. Another challenge is defining the threshold for a collision. Since the data was not cleaned or filtered, noise in the scans could result in false collisions. Thus, we calculated the density of the obstacles to decide on this threshold. This density is sensor-specific and should be adjusted according to the data source.

3.4.4 Target-Bias: In standard RRT, sampling is uniformly random across the search space, which can result in inefficiencies in large spaces or narrow openings. A performance-enhancing solution is target-biased sampling. The algorithm still randomly explores but, with a probability of p, the tree is expanded from the nearest node to the goal, in the direction of the goal.

3.4.5 Step Size: Defined as the distance between current node (n) and sample node (nsample), step size is set as the width of the bounding box used for collision checking. As the algorithm checks for collisions around nsample, which is at the center of the bounding box, the previous check for n ensures the first half of the step is obstacle-free. Therefore, checking for nsample, results in a clear path. Bounding boxes and step size is seen in **Figure 3**.

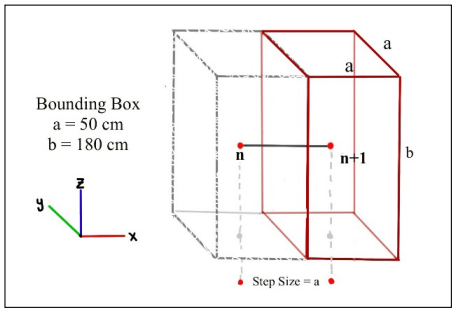


Figure 3. Bounding box and step size.

3.4.6 Bi-Directional Search: Search process involves two RRT trees: Tree A (TA) and Tree B (TB). TA starts the search from initial coordinates and expands towards the goal coordinates (due to bias). Meanwhile, TB starts at the goal coordinates and expands to the latest node of TA in a sequential search process.

3.4.7 Path Optimization, Normalization & Export: When a connection is established between initial and goal points, rewiring is executed which then checks if there is a path with lower cost. Furthermore, as the path represents a connection between crucial points of collision check bounding boxes, the coordinates are normalized to the floor level by reducing the Z component by 0.7 meters (**Figure 4**). This process is done to optimize the visualization results. Lastly, the coordinate list of nodes is exported in a plain text file.

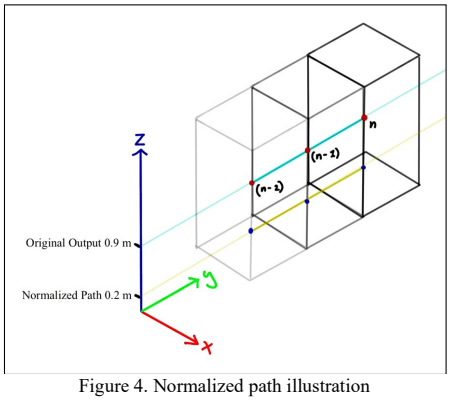


Figure 4. Normalized path illustration

3.4.8 Visualization: The user interface of the real-time 3D modelling system is developed with a game engine, which is also used for visualizing the 3D model. Consequently, the path visualization is done within the same environment. A C# script that accepts the resulting text file from the path planning is used to visualize the path. Since the path shares the same coordinate system with the point clouds, the final path, after normalization, appears approximately 0.2 meters above the surface and provides visual guidance that does not require additional directions (Figure 5).

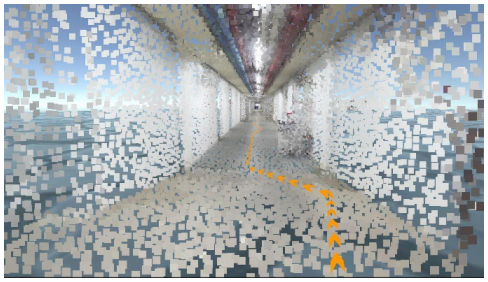


Figure 5. Path visualized over the dataset in game engine.

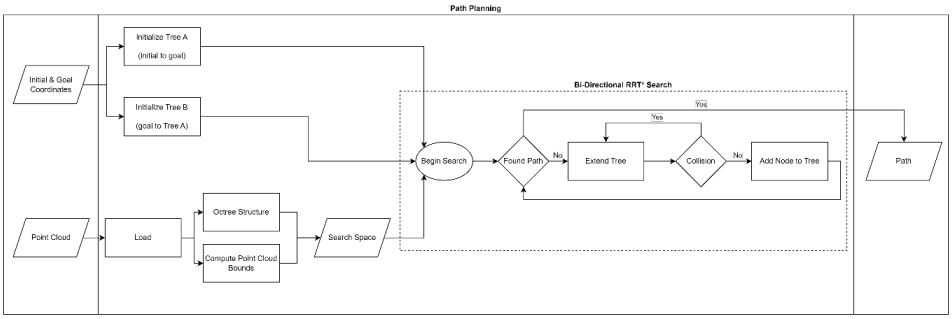


Figure 6. Path Planning Flowchart.

4. Results & Discussion

4.1 System Performance and Future Directions

The system's effectiveness was primarily measured by the speed and success of the pathfinding process, with the RRT algorithm tailored to prioritize rapid, safe route generation over finding the shortest path. This approach was chosen to reflect the urgent need for emergency navigation, where conditions can change unpredictably. The current implementation serves as a proof of concept, with plans for future tests to validate and refine the system under a broader range of real-world conditions. The aample results for the longest test path is given below.

Testing hardware:

- Processor: Intel i7-10850H CPU @ 2.70GHz
- RAM: 32 GB DDR4
- GPU: NVIDIA Quadro T1000 4 GB DDR6

Dataset:

- Environment: Basement (Figure 7)
- Sensor: GeoSLAM Zeb Horizon RT
- Point Cloud Size: 1,407,430 points
- Path Length: 83 meters
- Average Time per 1000 Iterations: 0.61 seconds
- Total Run Time (octree generation and ~15,000 iterations): 11.89 seconds

Note that the hardware capacity used in testing is significantly lower than that of the edge computer. Therefore, we consider these rates promising.

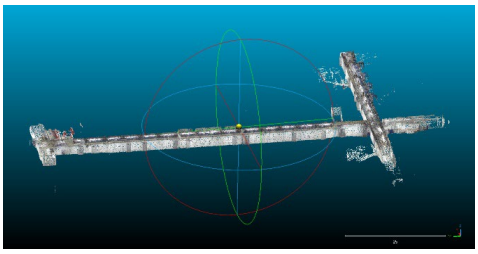


Figure 7. Basement dataset (Scale bar = 25 m).

4.2 Challenges

4.2.1 Sensor limitations: On one hand, it is possible to generate a point cloud in real-time. However, the resulting point cloud often has gaps where the algorithm ends up leaving the point cloud, or there are obstacles that are not completely scanned, hence considered as viable openings. On the other hand, some sensors can quickly scan the environment with much higher coverage, resulting in a dataset that is perfect for this methodology. However, it is not yet possible to access the point cloud without preprocessing. A comparison of the same environment with different scanners is shown below (Figure 8). Due to field of vision limitations, the ceiling is not covered by the HoloLens scan.

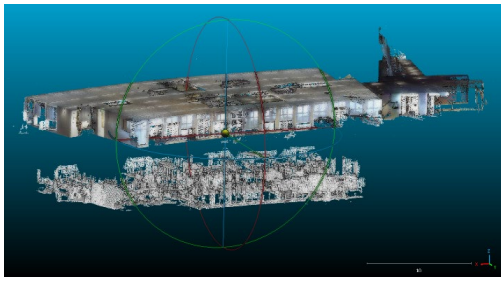


Figure 8 Office environment with Mobile Laser Scanner (top), Hololens 2 (bottom).

4.2.2 Step size: As mentioned in the methodology (3.4.2), the step size is set to 0.5 meters to accommodate the bounding box for collision checks. While this is effective for collision checking, a small step size negatively affects performance in large areas (e.g., the basement dataset). Variable step size implementations have proven to be useful in such scenarios (Zhang et al., 2019). However, using a variable step size would cause the collision check method to fail.

4.2.3 Target bias: Another method to overcome the challenges of large areas is introducing a target bias. This improves performance in simpler layouts; however, the bias probability also introduces a failure probability in complex layouts. As seen in Figure 9 below, both trees quickly advanced to their objective yet resulted in a failure due to iterating towards opposite sides of a wall and not being able to connect. The red and yellow dots represent the nodes that are closest to each other (0.2 meters apart). This challenge is especially evident in scenarios with multiple floors.

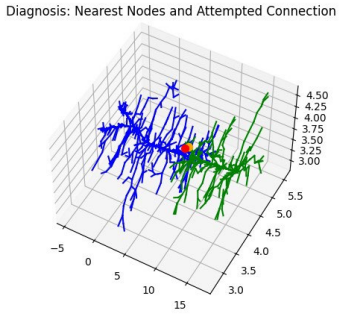


Figure 9. Faile due to biased sampling.

4.3 Discussion

Results show that it is possible to achieve efficient path planning by direct use of 3D indoor point clouds. However, it is not feasible with the current 3D modelling system due to limitations of the sensors. First of all, coverage of the HoloLens 2 FOV (field of view) and Range is inadequate for this methodology as HoloLens 2 data has gaps and unscanned obstacles which is then considered as empty space and used for path planning by the RRT search. Second, data acquired by mobile laser scanners were suitable for search. On the other hand, streaming the point clouds in real-time with these devices is not yet possible.

Another aspect to mention is the existence of drift in the scans. Especially with the HoloLens 2 and BLK2GO scans. Loop closure is practiced with Zeb Horizon RT scans while BLK2GO scans conducted without it to observe the effects. There is visible drift even in comparably short scans (>1 m drift in scans less than 10 minutes). For our methodology it does not effect the results as the path finding and visualization share the positioning and coordinate system of the point cloud. However, it could result in inaccuracies if georeferencing is required.

4.4 Future works

4.4.1 Parallel Processing / Multithreading: Processing the search from both trees simultaneously instead of the current sequential methodology could yield greater improvements in performance.

4.4.2 Continuous and Variable Collision Check: In our research bounding box hence, the collision check is limited to a large rectangle box. In a scenario where there is an opening that only allows the user to crawl through, our method will fail. However, using a smaller bounding box that represents the minimum space required to pass could result in an unfeasible path (e.g. a path that requires to crawl under tables when there is an opening large enough to walk next to it). Implementing a variable method that changes the size after certain amount of failed iterations could solve this problem. In addition, a method that can check around the proposed path constantly instead of bound based controls (without hindering the performance) can facilitate variable step size and provide performance increase.

4.4.3 Visualization Improvements: Implementing gaussian splatting to the point clouds could result in a visual that is much easier to comprehend for the observers (commanding agents) compared to the unstructured and semi-transparent look of the point clouds.

A. Appendix

Acknowledgements

We would like to acknowledge that we have utilized ChatGPT to enhance the readability and clarity of this document.

References

Balado, J., Díaz-Vilariño, L., Arias, P., Frías, E., 2019. POINT CLOUDS TO DIRECT INDOOR PEDESTRIAN PATHFINDING. Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. XLII-2/W13, 753–759. https://doi.org/10.5194/isprs-archives-XLII-2-W13-753-2019

Boguslawski, P., Zlatanova, S., Gotlib, D., Wyszomirski, M., Gnat, M., Grzempowski, P., 2022. 3D building interior modelling for navigation in emergency response applications. Int. J. Appl. Earth Obs. Geoinformation 114, 103066. https://doi.org/10.1016/j.jag.2022.103066

Broersen, T., Fichtner, F.W., Heeres, E.J., Ivo, de L., Rodenberg, O.B.P.M., Verbree, E., Voûte, R., 2016. Project Pointless: Identifying, visualising and pathfinding through empty space in interior point clouds using an octree approach., in: Proceedings of the 19th AGILE Conference on Geographic Information Science. Presented at the 19th AGILE Conference on Geographic Information Science., Helsinki, Finland.

Carbonari, G., Ricci, M., Calderoni, M., Dourlens-Quaranta, S., 2020. Building logbook state of play: report 2 of the study on the development of a European Union framework for buildings' digital logbook. Publications Office, LU.

Díaz-Vilariño, L., Tran, H., Frías, E., Balado, J., Khoshelham, K., 2022. 3D MAPPING OF INDOOR AND OUTDOOR ENVIRONMENTS USING APPLE SMART DEVICES. Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. XLIII-B4-2022, 303–308. https://doi.org/10.5194/isprs-archives-XLIII-B4-2022-303-2022

Fichtner, F.W., Diakité, A.A., Zlatanova, S., Voûte, R., 2018. Semantic enrichment of octree structured point clouds for multi-story 3D pathfinding. Trans. GIS 22, 233–248. https://doi.org/10.1111/tgis.12308

Flikweert, P., Peters, R., Díaz-Vilariño, L., Voûte, R., Staats, B., 2019. AUTOMATIC EXTRACTION OF A NAVIGATION GRAPH INTENDED FOR INDOORGML FROM AN INDOOR POINT CLOUD. ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci. IV-2/W5, 271–278. https://doi.org/10.5194/isprs-annals-IV-2-W5-271-2019

Kapucu, N., Garayev, V., 2011. Collaborative Decision-Making in Emergency and Disaster Management. Int. J. Public Adm. 34, 366–375. https://doi.org/10.1080/01900692.2011.561477

LaValle, S.M., 1998. Rapidly-exploring random trees : a new tool for path planning. Iowa State Univ. Annu. Res. Rep.

Morlighem, C., Chatzidiakos, C., Feenstra, J., van Schendel, M., Hurkmans, R., 2020. Deployment of Indoor Point Clouds for Firefighting Strategy.

Nikoohemat, S., Diakité, A.A., Zlatanova, S., Vosselman, G., 2020. Indoor 3D reconstruction from point clouds for optimal routing in complex buildings to support disaster management. Autom. Constr. 113, 103109. https://doi.org/10.1016/j.autcon.2020.103109

Patel, V., Grewal, Dr.R., 2022. AUGMENTED REALITY BASED INDOOR NAVIGATION USING POINT CLOUD LOCALIZATION. Int. J. Eng. Appl. Sci. Technol. 6, 67–77. https://doi.org/10.33564/IJEAST.2022.v06i09.009

Smit, B.-P., Voûte, R., Verbree, E., 2021. CREATING 3D INDOOR FIRST RESPONDER SITUATION AWARENESS IN REAL-TIME THROUGH A HEAD-MOUNTED AR DEVICE. ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci. V-4–2021, 209–216. https://doi.org/10.5194/isprs-annals-V-4-2021-209-2021

van Schendel, M., 2022. Merging of Topometric Maps of Indoor Environments.

Wang, X., Wei, J., Zhou, X., Xia, Z., Gu, X., 2022. AEB-RRT*: an adaptive extension bidirectional RRT* algorithm. Auton. Robots 46, 685–704. https://doi.org/10.1007/s10514-022-10044-x

Zhang, Z., Wu, D., Gu, J., Li, F., 2019. A Path-Planning Strategy for Unmanned Surface Vehicles Based on an Adaptive Hybrid Dynamic Stepsize and Target Attractive Force-RRT Algorithm. J. Mar. Sci. Eng. 7, 132. https://doi.org/10.3390/jmse7050132

Zheng, Z., Bewley, T.R., Kuester, F., Ma, J., 2022. BTO-RRT: A rapid, optimal, smooth and point cloud-based path planning algorithm.

B. Appendix

B.1. Line Renderer with Unity

B.1.1. C# Script for Visualization

In this section, we explain the C# script used for visualization in our research. The script is responsible for translating the coordinates from the path planning algorithm into visual elements within the Unity game engine. It uses the Line Renderer tool to draw the path based on the coordinates provided in a `path.txt` file.

Script Overview

The script initializes the necessary components, loads the path data from a text file, and uses the Line Renderer to create a visual representation of the path. The primary functions are:

- **Start()**: Initializes the components and loads the path data from the `path.txt` file.
- **LoadPath()**: Reads the path data and stores it in a list.
- **RenderPath()**: Uses the Line Renderer to visualize the path with a chevron-shaped texture.

Script Listing

```
using System.Collections.Generic;
using UnityEngine;
using System.IO;

public class PathVisualizer : MonoBehaviour
{
    public LineRenderer lineRenderer;
    public string pathFileName = "path.txt";
    private List<Vector3> pathPoints;

    void Start()
    {
        lineRenderer = GetComponent<LineRenderer>();
        pathPoints = new List<Vector3>();
        LoadPath();
        RenderPath();
    }

    void LoadPath()
```

```
20      {
21          string path = Path.Combine(Application.dataPath, pathFileName);
22          if (File.Exists(path))
23          {
24              string[] lines = File.ReadAllLines(path);
25              foreach (string line in lines)
26              {
27                  string[] coords = line.Split(',');
28                  if (coords.Length == 3)
29                  {
30                      float x = float.Parse(coords[0]);
31                      float y = float.Parse(coords[1]);
32                      float z = float.Parse(coords[2]);
33                      pathPoints.Add(new Vector3(x, y, z));
34                  }
35              }
36          }
37          else
38          {
39              Debug.LogError("Path file not found: " + path);
40          }
41      }
42
43      void RenderPath()
44      {
45          lineRenderer.positionCount = pathPoints.Count;
46          lineRenderer.SetPositions(pathPoints.ToArray());
47          lineRenderer.material = new Material(Shader.Find("Sprites/
                Default"));
48          lineRenderer.material.SetTexture("_MainTex", Resources.Load<
                Texture2D>("ChevronTexture"));
49          lineRenderer.startColor = Color.white;
50          lineRenderer.endColor = Color.white;
51          lineRenderer.startWidth = 0.1f;
52          lineRenderer.endWidth = 0.1f;
53          lineRenderer.textureMode = LineTextureMode.Tile;
54          lineRenderer.alignment = LineAlignment.TransformZ;
55      }
56  }
```

Listing B.1: C# Script for Path Visualization using Line Renderer

C. Appendix

C.1. Dataset Figures

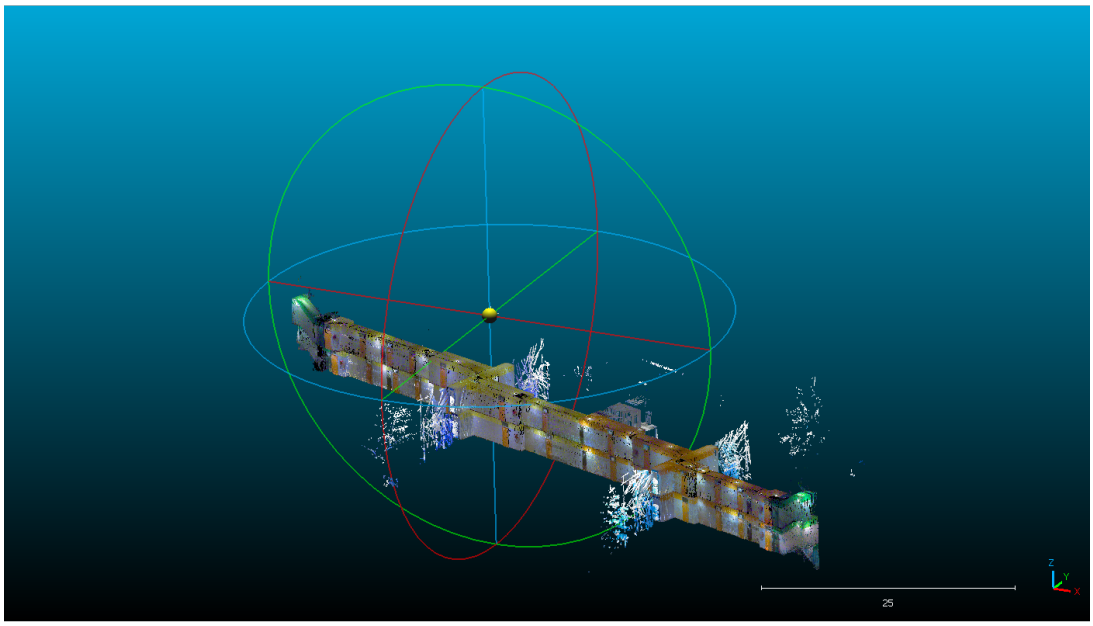


Figure C.1.: BLK2GO Apartment Complex

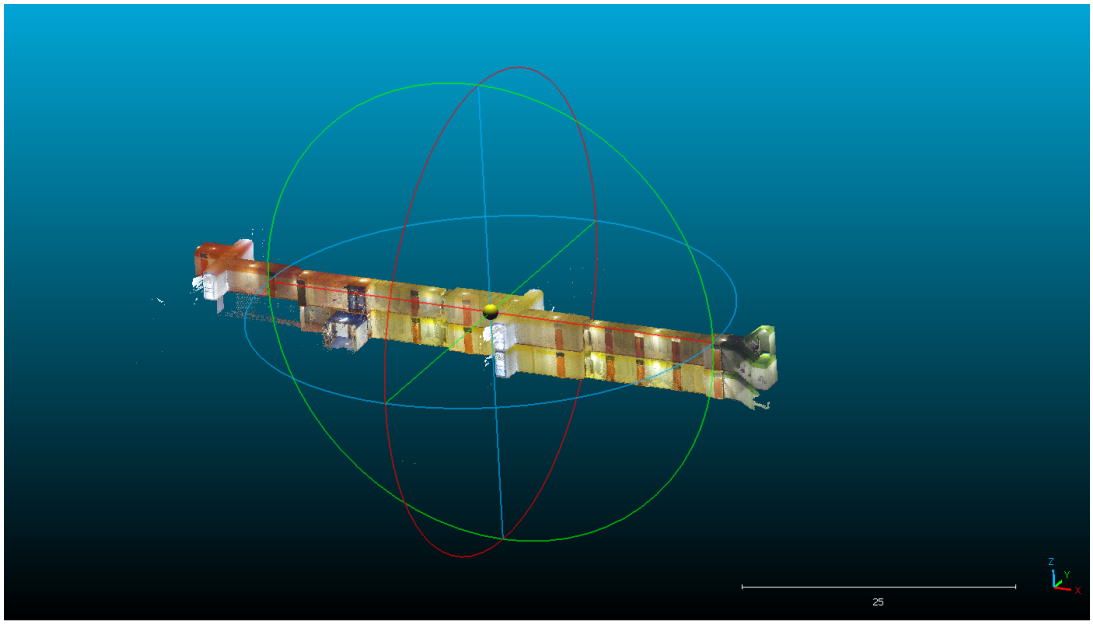


Figure C.2.: Zeb Horizon Apartment Complex

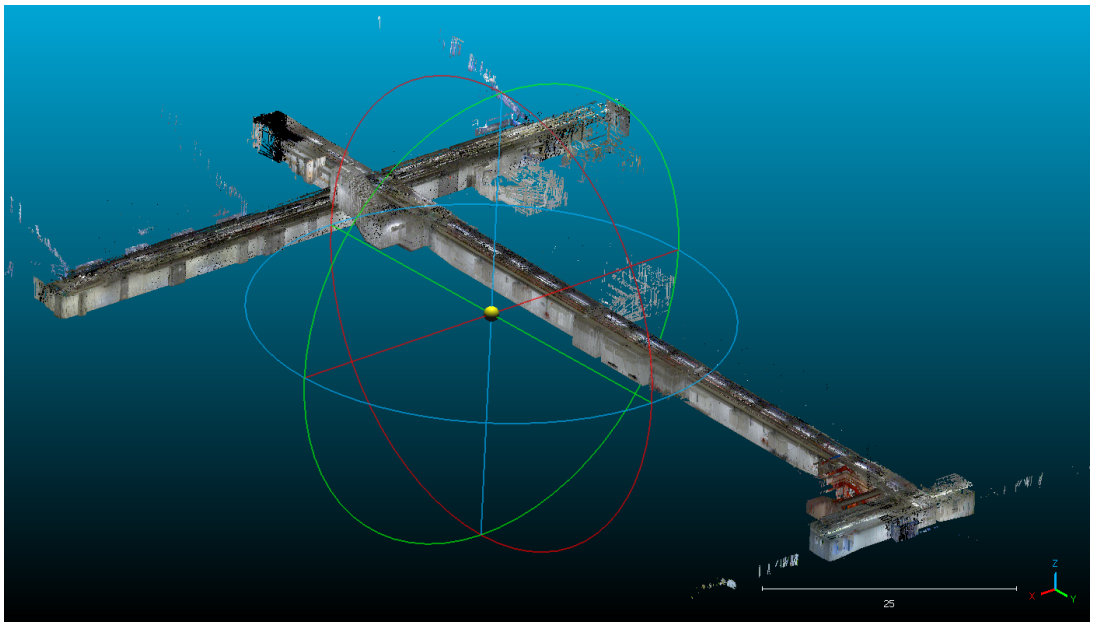


Figure C.3.: BLK2GO Basement

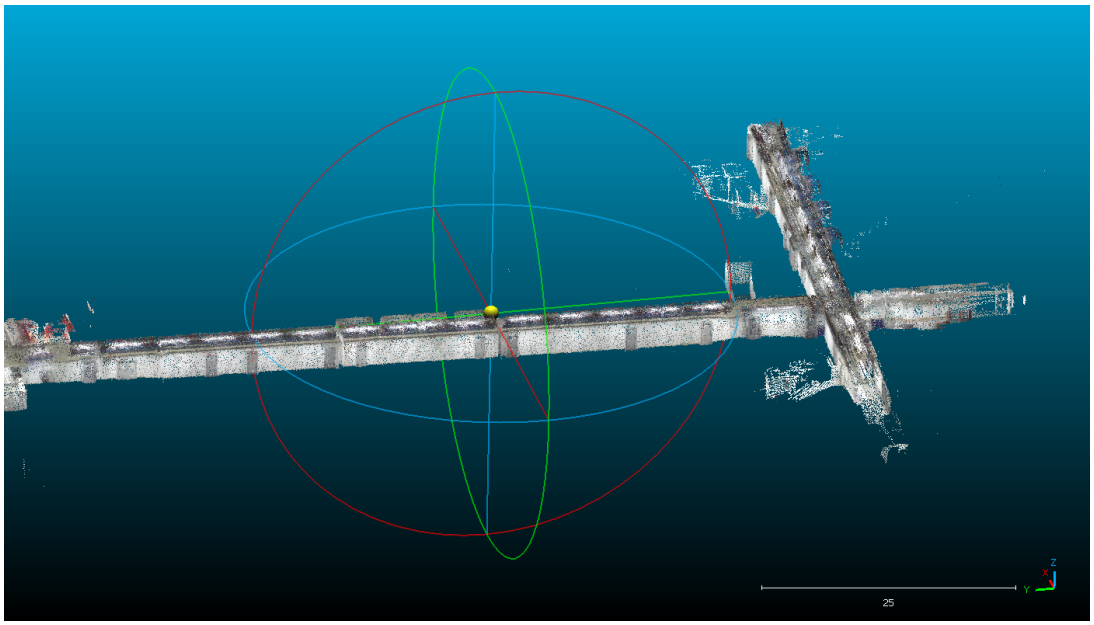


Figure C.4.: Zeb Horizon Basement

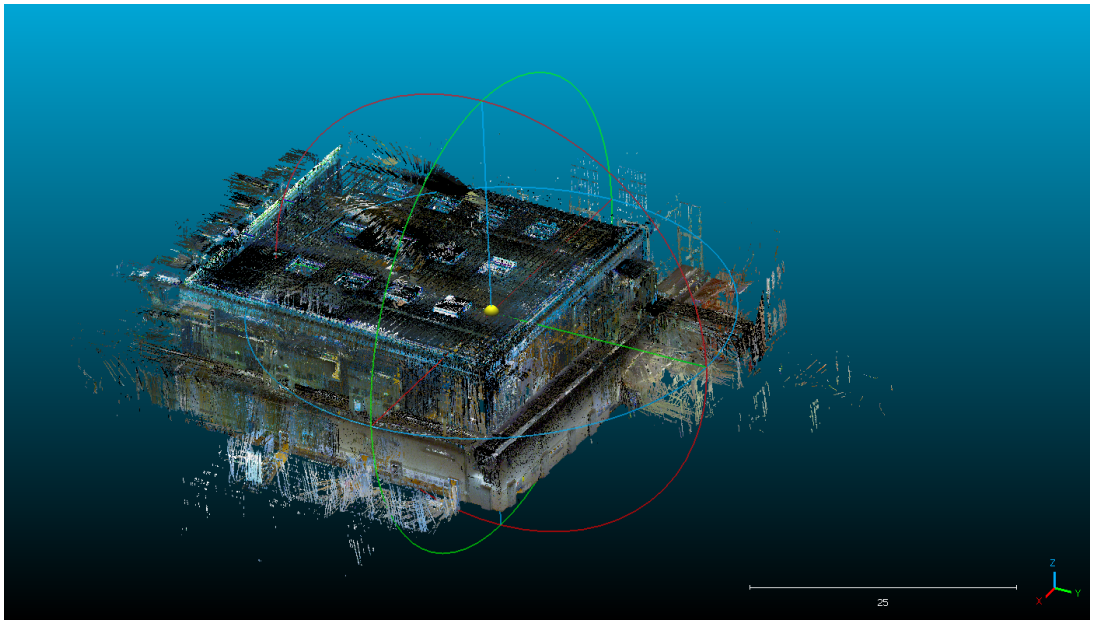


Figure C.5.: BLK2GO Hall

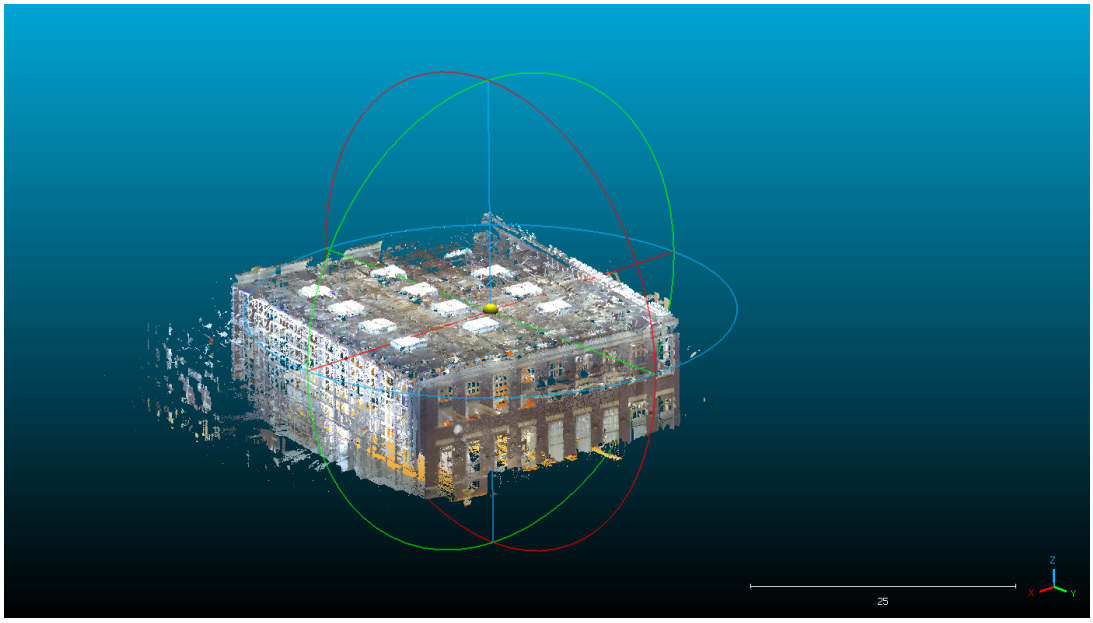


Figure C.6.: Zeb Horizon Hall

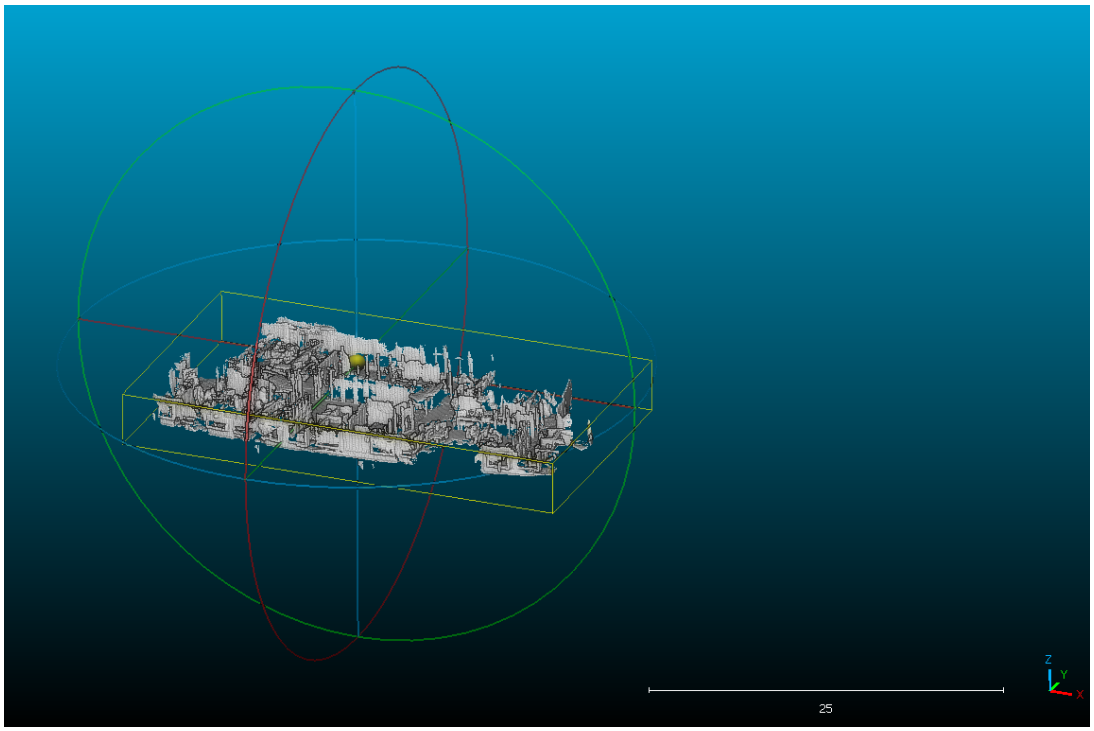


Figure C.7.: HoloLens 2 Office

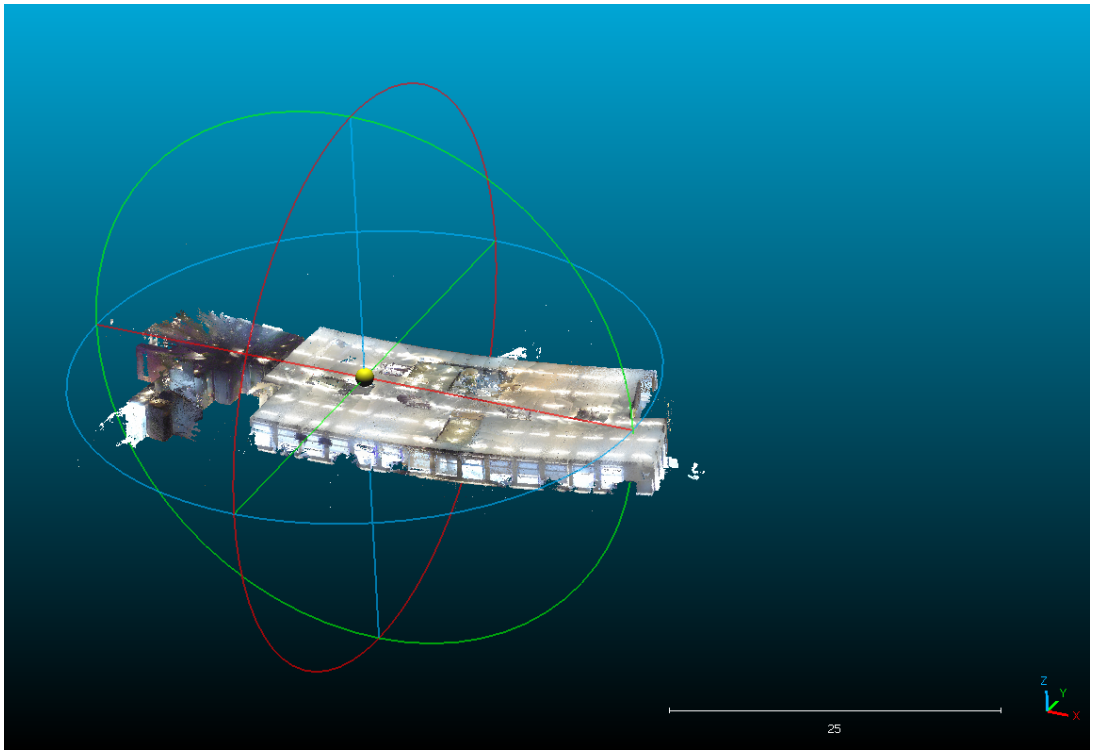


Figure C.8.: Zeb Horizon Office

Bibliography

Aernouts, M., Bellekens, B., and Weyn, M. (2018). MapFuse: Complete and Realistic 3D Modelling. *Journal of Robotics*, 2018:1–13.

Angelats, E. and Navarro, J. A. (2017). TOWARDS A FAST, LOW-COST INDOOR MAPPING AND POSITIONING SYSTEMFOR CIVIL PROTECTION AND EMERGENCY TEAMS. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W8:9–15.

Balado, J., Díaz-Vilariño, L., Arias, P., and Frías, E. (2019). POINT CLOUDS TO DIRECT INDOOR PEDESTRIAN PATHFINDING. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W13:753–759.

Boguslawski, P., Zlatanova, S., Gotlib, D., Wyszomirski, M., Gnat, M., and Grzempowski, P. (2022). 3D building interior modelling for navigation in emergency response applications. *International Journal of Applied Earth Observation and Geoinformation*, 114:103066.

Broersen, T., Fichtner, F. W., Heeres, E. J., Ivo, d. L., Rodenberg, O. B. P. M., Verbree, E., and Voûte, R. (2016). Project Pointless: Identifying, visualising and pathfinding through empty space in interior point clouds using an octree approach. In *Proceedings of the 19th AGILE Conference on Geographic Information Science.*, Helsinki, Finland.

Carbonari, G., Ricci, M., Calderoni, M., and Dourlens-Quaranta, S. (2020). *Building logbook state of play: report 2 of the study on the development of a European Union framework for buildings' digital logbook.* Publications Office, LU.

Díaz-Vilariño, L., Tran, H., Frías, E., Balado, J., and Khoshelham, K. (2022). 3D MAPPING OF INDOOR AND OUTDOOR ENVIRONMENTS USING APPLE SMART DEVICES. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLIII-B4-2022:303–308.

Endsley, M., Bolte, B., and Jones, D. (2003). *Designing for Situation Awareness: An Approach to User-Centered Design*.

Fichtner, F. W., Diakité, A. A., Zlatanova, S., and Voûte, R. (2018). Semantic enrichment of octree structured point clouds for multi-story 3D pathfinding. *Transactions in GIS*, 22(1):233–248.

Flikweert, P., Peters, R., Díaz-Vilariño, L., Voûte, R., and Staats, B. (2019). AUTOMATIC EXTRACTION OF A NAVIGATION GRAPH INTENDED FOR INDOORGML FROM AN INDOOR POINT CLOUD. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W5:271–278.

Kapucu, N. and Garayev, V. (2011). Collaborative Decision-Making in Emergency and Disaster Management. *International Journal of Public Administration*, 34(6):366–375.

Karaman, S. and Frazzoli, E. (2011). Sampling-based Algorithms for Optimal Motion Planning. arXiv:1105.1186 [cs].

LaValle, S. (1998). Rapidly-exploring random trees : a new tool for path planning. *Iowa State University Annual Research Report*.

Morlighem, C., Chatzidiakos, C., Feenstra, J., van Schendel, M., and Hurkmans, R. (2020). Deployment of Indoor Point Clouds for Firefighting Strategy.

Nasir, J., Islam, F., Malik, U., Ayaz, Y., Hasan, O., Khan, M., and Muhammad, M. S. (2013). RRT*-SMART: A Rapid Convergence Implementation of RRT*. *International Journal of Advanced Robotic Systems*, 10(7):299.

Nikoohemat, S., Diakité, A. A., Zlatanova, S., and Vosselman, G. (2020). Indoor 3D reconstruction from point clouds for optimal routing in complex buildings to support disaster management. *Automation in Construction*, 113:103109.

Noreen, I., Khan, A., and Habib, Z. (2016). A Comparison of RRT, RRT* and RRT*-Smart Path Planning Algorithms.

Nosrati, M., Karimi, R., and Hasanvand, H. A. (2012). Investigation of the * (Star) Search Algorithms: Characteristics, Methods and Approaches. *World Applied Programming*.

Patel, V. and Grewal, D. R. (2022). AUGMENTED REALITY BASED INDOOR NAVIGATION USING POINT CLOUD LOCALIZATION. *International Journal of Engineering Applied Sciences and Technology*, 6(9):67–77.

Rodenberg, O. (2016). *The effect of A* pathfinding characteristics on the path length and performance in an octree representation of an indoor point cloud.* PhD thesis.

Sithole, G. and Zlatanova, S. (2016). POSITION, LOCATION, PLACE AND AREA: AN INDOOR PERSPECTIVE. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, III-4:89–96.

Sliney, D. (1995). Risk assessment and laser safety. *Optics & Laser Technology*, 27(5):279–284.

Smit, B.-P., Voûte, R., and Verbree, E. (2021). CREATING 3D INDOOR FIRST RESPONDER SITUATION AWARENESS IN REAL-TIME THROUGH A HEAD-MOUNTED AR DEVICE. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, V-4-2021:209–216.

Van Oosterom, P. (1999). Spatial Access Methods. In *Geographical Information Systems Principles, Technical Issues, Management Issues, and Applications*, volume 1, pages 385–400. Wiley.

van Schendel, M. (2022). Merging of Topometric Maps of Indoor Environments.

Wang, X., Wei, J., Zhou, X., Xia, Z., and Gu, X. (2022). AEB-RRT*: an adaptive extension bidirectional RRT* algorithm. *Autonomous Robots*, 46(6):685–704.

Zhang, Z., Wu, D., Gu, J., and Li, F. (2019). A Path-Planning Strategy for Unmanned Surface Vehicles Based on an Adaptive Hybrid Dynamic Stepsize and Target Attractive Force-RRT Algorithm. *Journal of Marine Science and Engineering*, 7(5):132.

Zheng, Z., Bewley, T. R., Kuester, F., and Ma, J. (2022). BTO-RRT: A rapid, optimal, smooth and point cloud-based path planning algorithm. arXiv:2211.06801 [cs, eess].

Colophon

This document was typeset using LaTeX, using the KOMA-Script class `scrbook`. The main font is Palatino.