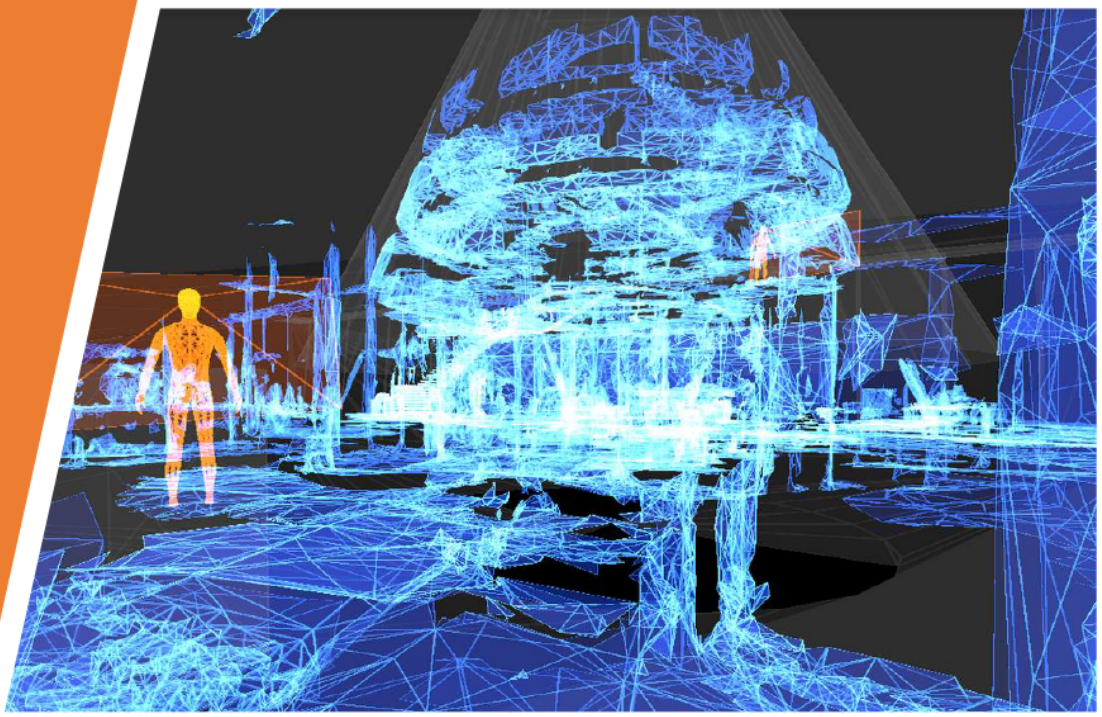
TU Delft

2020 Synthesis Project



Deployment of Indoor Point Clouds for Firefighting Strategy

Camille Morlighem
Charalampos Chatzidiakos
Jos Feenstra
Max van Schendel
Robin Hurkmans

Deployment of Indoor Point Clouds for Firefighting Strategy

GEO1011 - Synthesis Project

Camille Morlighem, *5156238*
Charalampos (Babis) Chatzidiakos, *5070465*
Jos Feenstra, *4465768*
Max van Schendel, *4384644*
Robin Hurkmans, *4370511*

Supervisors: Edward Verbree, Robert Voûte
Clients: Huib Fransen, Rob Braggaar

TU Delft
Delft University of Technology

Veiligheidsregio
Rotterdam-Rijnmond

CGI

Abstract

The Deployment of Indoor Point Clouds for Firefighting Strategy project was realised as a Synthesis Project of the Geomatics Master Programme of the Built Environment Faculty at the Technical University of Delft. This project was executed by a team of five Master students in collaboration with the Dutch response team collective Veiligheidsregio Rotterdam-Rijnmond.

The objective of this project is to develop an information system that makes use of indoor data to support tactical decision-making during fire emergency responses. The main challenge that response teams are facing when they develop deployment plans is the lack of appropriate information about indoor spaces. As a result, response teams may end up relying on inaccurate assumptions which can lead to dangerous situations. New technologies such as *SLAM* devices and *augmented reality* displays, combined with processing techniques, can be used to supply them with the information needed to make the right choices.

The result of this project is a prototypical information system containing an interactive, 3D environment that can receive updates, merge data from different data sources, and accommodate *mixed reality* information sharing in real-time.

Acknowledgements

Before we go into more details, we would like to thank our supervisors for their contributions to our Geomatics Synthesis Project. The realisation of this project would not have been possible without all these professionals to support and guide our team during critical moments.

First and foremost, we would like to thank Edward Verbree and Robert Voûte for their supervision. Mr. Verbree had a keen eye for project management and made sure we stayed on track. He also gave us feedback and advice on the product we developed and helped us with the general directions of this project. We would like to thank Robert Voûte and Bart-Peter Smit from *CGI* for providing us with a codebase we used as a starting point and for their advice on the technical parts of the product. We have had several meetings with Bart-Peter that really got us up to speed with his thesis code. Robert also entrusted us with *CGI*'s HoloLens and licences for Azure. On top of that, he also looked out for our mental well-being during the current Coronavirus crisis.

In addition, we would like to thank our clients Huib Fransen and Rob Braggaar from the Veiligheidsregio Rotterdam Rijnmond (VRR) for their feedback and advice on the demanded functionalities we have created. They really took a personal interest in our development, asking precise questions.

Lastly, we would like to thank the VR Zone TU Delft for the Microsoft HoloLens they lent us and the workspace they provided, and the TU Delft Library that allowed us to perform the product testing in their facility.

We would like to state that this whole synthesis project experience was very useful to all of us. Although we faced some difficulties in the beginning because working with a big group remotely is not an easy task, in the end we came up with a desirable and quite promising output. We hope that our project will be of big importance in the future and that it will help our clients to achieve their goals.

Contents

1

Project description

1.1 Introduction

The field of geo-information is currently undergoing a paradigm shift from maps as two-dimensional representations to full three-dimensional maps (Agugiaro, unpublished). Simultaneously, more and more end-users require systems that dynamically respond to changes in the environment. For this purpose, interactive digital maps are more suitable than traditional, static maps (Agugiaro, unpublished; Hardy and Field, 2011). The above-mentioned development means that a new area of geo-information must be explored: dynamic and real-time models of environments. It is vital to explore this field of study because it can be of tremendous use in many cases.

A particularly vital case to consider is fire emergency responses. In their work, firefighters often face stressful situations where they have to make important decisions in a limited timeframe (Sha et al. 2006). In van der Meer et al. (2018) it is stated that "*Information is key for a safe deployment of the fire brigade. An incorrect judgment could lead to decreased effectivity or even casualtie*s". The availability of information thus plays a crucial role for strategy development. During fire emergency responses, *information systems* are responsible for providing the information that is used to support strategy and decision-making. However, for such systems to be really supportive in emergency situations, they have to enhance *situational awareness,* otherwise these systems could lead to human and economic losses (Salmon et al. 2016; Prasanna et al., 2017). A challenge preventing the development of such a system is that indoor spaces can be regarded as terra incognita from a geo-data standpoint (Verbree, unpublished); Lidar and GPS cannot penetrate buildings and existing floor plans are often outdated. This means indoor data is sparse and a person entering a building cannot be properly tracked through conventional means (Mautz, 2012).

Technological progress in the fields of computer sciences and remote sensing has led to the emergence of head-mounted *Simultaneous Localisation And Mapping (SLAM)* devices combined with *mixed reality (MR)* displays. These devices can gather information about their environment, locate their user, and visually integrate virtual and real-world information (Hosch, n.d. ; Wu et al., 2013). Such devices allow users to have a better understanding of their surroundings, thus empowering *situational awareness*. Moreover, studies suggest that *MR* could immensely help firefighters in their tasks, as gathered data from such devices could be read, processed and refined in order to provide the firefighters with concise and useful information (Haase, 2017).

Previous studies have focussed on developing *MR*-based 3D *information systems* for supporting firefighters during emergency responses. These studies mainly developed helmets equipped with thermal imaging cameras to provide the user with thermal vision of his environment. However, it appears that until now there have been little attempts to develop a *MR information system* which supports collaborative, real-time data gathering, processing, distribution, and visualisation.

This project thus aims at developing such a system. The system consists of a centralised network containing two modules called the Navigator and the Explorer. The Explorer is responsible for gathering environmental data and providing *mixed reality* information to its user. It runs on a head-mounted *MR* device that is worn by a firefighter during deployment. The Navigator is a dynamic 3D environment which is used by the commander of the response team. It receives data from any number of Explorers, interprets it, and visualises it so as to maximise its value for tactical decision-making. It is also able to send information to Explorers, to direct their actions or to provide them with information that enhance their *situational awareness.* The modules' functionalities are specifically tailored to the needs of the Dutch response team collective Veiligheidsregio Rotterdam-Rijnmond (VRR).

This project report starts with a description of the research questions and project expectations in chapter 1. Chapter 2 provides an overview of the previous works and researches related to this project. Chapter 3 addresses the theoretical background related to our use case and the theoretical concepts. After this, the methodology implemented to develop our product is described in chapter 4 and the results achieved are shown in chapter 5. Chapter 6 includes the quality assessment and tests performed to test the product. Afterwards, chapter 7 includes the research conclusions and recommendations. Finally, chapter 8 addresses the project organisation.

1.2 Project expectations

To better express what features this product may or may not have the MoSCoW method was used. This method separates the features of the envisioned product into four categories: must-, should-, could-, and won't haves, to achieve a better understanding of the product that will be developed. *Must haves* are the essential features of the project without which the product cannot be considered finished. *Should haves* are not critical, but very much needed in support of the *Must haves*. *Could haves* would make good additions to the product, but are further away from the core of what the product should be. *Won't haves* are defined to clarify the scope of the project, to determine what we are not trying to do.

Must haves

- Merging of two HoloLens point clouds to create a unified, interactive 3D map
- A central hub where the map is visualised along with information about personnel
    - Point cloud or mesh display
    - Position / orientation of personnel
- Simple Hub to HoloLens communication (HuHo)
    - Place pings in map which are shown on the HoloLens
    - Show location and distance of other personnel

Should haves

- Add basic semantics / information manually (points, lines, areas, volumes)
    - Purpose: Users must be able to manually add information (as points, lines) to the environment
- Add basic semantics automatically (floor, wall, ceiling, etc.)
    - Principal component analysis
    - Purpose: Control room must be able to make sense out of the data they receive

Could haves

- X-ray vision, see mapped space on HoloLens
- Tracking of path & time walked (camera feed to control)
- Simple HoloLens to Hub communication (HoHu)
- Add advanced semantics automatically (object recognition)
- Holo to Holo communication, e.g. shouts: "Watch out!", "Over here!"
- An HUD to show extra information to personnel (current floor, distance to exit, minimap)
- Embedding georeferenced dataset with GIS data (showing context, adding indoor point cloud to the actual building within a city model)
- Make this dataset usable within a "*veiligheidsapp*" type app on mobile devices
- Fire detection
- Connect different sensors (IoT)
- Face / person detection to aid in finding casualties

Won't haves

- Audio communication
- Product testing with real response teams (surveys)
- Indoor wayfinding
- Data acquisition
- Researching the capabilities of using *SLAM* devices in harsh conditions (smoke, heat, wind, cold)

1.3 Objectives and research questions

Our objective in this Synthesis Project is to develop an *information system* making use of indoor point clouds to support real-time tactical decision-making during fire emergency responses. This *information system* should answer as best as possible the user needs. This goal leads us directly to the research questions that this project intends to answer. The main research question is as follows:

"*In what way can the use of indoor point clouds assist firefighter response teams in real-time tactical decision-making?*"

To answer this question there are multiple sub-questions that need to be answered as well. These sub-questions are listed below. Keywords are bolded.

- *"What **geoinformation** do the response teams **need**?"*
- *"In what **form** should we provide this **information** to firefighters so it can be effectively used?"*
- "*How can **information** be **communicated** within and between tactical response teams?*"
- "*How can the raw point clouds **data** acquired by response teams be **enriched** so as to maximise its value for tactical decision-making?*"
- "*How can we **process** raw **data** from multiple sources in real-time?*"
- *"How can we acquire real-time tracking of the firefighter team(s) and **share** that **information** with the control room?"*

Before going more into details into this project report, it should be mentioned that although the main research topic is about point clouds, they are rarely used directly within this project. Although some functionality uses point clouds, and the raw data that the HoloLens captures is a point cloud, we almost exclusively use meshes which are derived from point clouds. Because the HoloLens automatically generates meshes from its input, we found it would be a waste of computing power to discard these in favour of point clouds. Besides, as we will show later in this report, in most cases meshes are more suited for the purposes of this project.

2

Related work

This chapter gives an overview of the previous research and work related to the development of *information systems* to support firefighters during emergency operations. This chapter shows how our product differentiates itself from previous work in that field.

There have been several attempts to develop and implement *information systems* for supporting decision-making and helping firefighters in fire emergency situations. In that way, de Leoni et al. (2007) developed the WORKPAD architecture. This architecture consists of two levels; one level for the control room and a second level for response teams sent into the field. In that system, response teams are equipped with mobile devices and communicate with each other through a *MANET (Mobile Ad-hoc Network)* while the communication with the control room is made through *TETRA (TErrestrial Trunked RAdio).* Similarly, Klann (2008) implemented LifeNet, an Ad-Hoc Network accompanied by a wearable system that provides navigational support to firemen during a fire emergency. Wilson et al. (2005) and Wilson et al. (2007) implemented the FIRE project. They developed a head-mounted display that makes use of a *WSN (wireless sensor network)* for indoor tracking of firefighters. The *information system* makes it possible to share tracking information as well as health information about firemen and fire information to the *incident commander*. More recently, Roh (2015) created an architecture that allows transforming environmental information (temperature and humidity) and personal physical information into digital signals that are then sent by radio frequency to the *information system*. Firemen tracking is also implemented with the use of an *IMU*. On similar attempts, there is Jiang et al. (2004) with the Siren project that proposed a context-aware messaging application and Nguyen et al. (2015) that also developed a communication system. However, the similarity between all these researches is that the information provided by the system for assisting firefighters in their work is only two-dimensional.

Some research also focussed on developing 3D *information systems*. For instance, Held et al. (2019) designed the concept of Helon 360. It consists of a helmet equipped with a 360-degree thermal imaging camera that allows sending the thermal data to the *incident commander* who can visualise them through a tablet device. It helps firefighters in their tasks by providing them an augmented thermal vision, warning messages from the *incident commander*, and personal information. In the range of helmets, Rosenbauer (2017) proposes the HEROS-Titan Helmet also supporting a thermal imaging camera (see figure 1). Lastly, Qwake Tech (2018) designed a helmet integrating many technologies. Indeed, thermal imaging cameras are implemented into the helmet combined with *augmented reality* to provide an augmented thermal vision to the user. The data can then be sent through a wireless network to the *incident commander*. All those helmet prototypes are thus based on

thermal imagery. As a result, it appears that until now there has been no successful attempt to develop a 3D *information system* making use of smart glasses such as the Microsoft HoloLens and integrating data processing such as the semantic classification for supporting firefighters during fire emergency responses.

Figure 1: HEROS-Titan helmet with thermal imaging camera. Source: Rosenbauer, 2017

3

Theoretical context and concepts

The theoretical context and concepts are based on literature research and an interview conducted with the clients of the project. This knowledge is explained in the following chapter. This chapter is important, firstly to understand what requirements our product needs to take into account to best answer the firefighters response teams needs, and secondly, to help understand the technology on which our product relies.

3.1 Theoretical Context

The theoretical context is based on literature research and on an interview (see appendix A). During the development of the product, we sought out information regarding firefighter strategies, the user requirements, and actions taken during a building fire in order to follow the correct direction and derive a desirable output. In this chapter, this context will be discussed.

3.1.1 Actions During Building Fires

To make sense of how and when our product would be useful, it is essential to look into the actions that take place during firefighters operations. We will explain the strategies used by the Dutch fire brigade. As stated by van der Meer et al. (2018), the deployment of an emergency response includes several phases in the fire department of the Netherlands. The first phase is the **alert phase** during which some basic information is provided such as the address and the name of the building, the type of incidents and which units are needed (van der Meer et al., 2018). The second phase is the **en-route** phase. In this phase, the firefighters get in the fire truck and put on their protective suits. This phase is followed by the **on-site/exploration** phase. At this point, the fire brigade collects information to determine their strategy and decide whether to approach the building from outside or from indoors (van der Meer et al., 2018; interview). This phase is more related to the product we developed since the availability of information about the building characteristics plays a crucial role during this phase. Indeed, building characteristics affect the development of building fires and the safety of the fire brigade deployments (van der Meer et al., 2018). Besides the building characteristics, fire and human characteristics are also analyzed in that phase. Finally, the fire brigade can determine their strategy and decide whether to tackle the problem from outside or from indoors. If an indoor deployment is chosen, then the firefighters have to explore the indoor space of the building, which means that they have to move towards the fire. The attacking team tries to find the fire source, determines attack routes, and verifies whether the fire can be extinguished with hose reels already available (van der

Meer et al., 2018). In cases like these, dynamic 3D maps of the building would be very useful and it would help firefighters avoid relying on assumptions, which can be dangerous.

3.1.2 User requirements

Assessing the user requirements for an *information system* supporting decision-making during building fires is a challenging task. Compared to other emergency-related jobs, it can be claimed that firefighters face the most stressful and life-threatening situations where they have a limited timeframe to make a life and death decision (Sha et al., 2006). Moreover, as stated by van der Meer et al. (2018), not only the user requirements vary among the different firefighter functions but they also vary among the people occupying the same function. Taking that into consideration, the information should be provided to the right person at the appropriate time (near real-time). In this project, we tried to assess as best as possible the user requirements based on a literature review and an interview conducted with the project clients (included in appendix A).

About the *information system* itself, research realised by (Prasanna et al, 2017) shows that such a system does improve decision-making in emergency cases only if the system :

- "*Consists of a network of technologies to capture data from the incident environment;*
- *Consists of a mechanism to process data;*
- *Is capable of making unprocessed data more meaningful by organisation or classification; and*
- *Can cater for individual needs and display different categories of information in a way that is meaningful for those individuals.*"

However, besides the quality and capabilities of the *information system*, usability should also be taken into account. Explanations to the users why and how the functionality of the innovation can help them are also important as well as easy access and start-up of the system (van der Meer et al, 2018 ; interview).
The information that should be acquired from the *information system* for safe deployment and decision-making was also assessed during the interview and consists of:

- Real-time tracking of the firefighter team and fire location and expansion;
- Location awareness of building features;
- A combination of 2D and 3D representations that could result from the implementation of ToggleMaps (van der Meer et al, 2018); and
- Filtered information as the information displayed should be meaningful, but too much information might confuse the users.

According to the conducted interview, the most important building features that the firefighters need to know during the deployment of their strategy are:

- The geometry of the building;
- Selected (dedicated) fire entrances;
- Stairways and elevators; and

- Adjacent buildings or wings of the building.

Finally, the display device also matters as firefighters need to keep their hands free on the field. For those reasons, devices mounted on the wrist or on the head could be suitable (van der Meer et al, 2018 ; interview).

3.1.3 Challenges of the use case

Time at which information is provided

The time during which firefighters can look at the available information is limited. This is a challenge to take into consideration when developing our application. Indeed, if the information offered needs too much time to be in the right form and available to the right person, then people will most likely not use it. Thus, our goal was to make the information available in real-time or near real-time.

Dangerous environments in which firefighters work

Firefighters have to operate under risky circumstances and they have to enter quite complex and dangerous environments. During rescue operations, they enter buildings and they are often exposed to physical and physiological risk factors including smoke inhalation, heat, stress, and fatigue (Richmond et al., 2008). Such environments are usually surrounded by heavy smoke which has a significant effect on the time required to locate a person (Abulhassan and DeMoulin, 2017) or even sometimes makes it impossible. Vision-based tracking technologies cannot operate in environments where flames and smoke are present. Moreover, GPS signals are blocked by the walls of the building so it is not an option either (Mautz, 2012). Tracking the path and keeping track of the time walked by firefighters is something that we wanted to include in our product so we needed to overcome those obstacles.

Acceptance of new technologies

For our application to work, some hardware is required (more details in chapter 4). Sometimes, this hardware can be rather complex, especially when the intended users are not familiar with such devices (Prasanna et al., 2017). As noted by van der Meer et al (2018), the firefighters prefer to visit the building in reality to experience the building and their environment. During the implementation of our product, we assumed that not all firefighters are familiar with complex hardware and that they are unlikely to take extra time to consult digital information. Thus, we tried to develop an intuitive, user-friendly application.

Privacy

Privacy is also an important challenge. It should be noticed that not all emergency workers are allowed access to the same information. Authorisations for all different levels and types of emergency workers would have been necessary to take into account (van der Meer et al, 2018).

3.2 Theoretical Concepts

Theoretical Concepts are based on literature research. Those concepts are needed to better understand the theory and the scope related to the developed product. Those concepts also highlight what components the implemented application must have. They are discussed in the following section.

3.2.1 Augmented reality and mixed reality

Klopfer and Squire (2008) define *Augmented Reality (AR)* as "*a situation in which a real-world context is dynamically overlaid with coherent location or context-sensitive virtual information*". Real-world and virtual world are thus blended in a meaningful way and in real-time. In other words, real world videos or images are overlaid with computer-generated visuals in a way that facilitates and increases user interactions and understanding of its surrounding environment (Hosch, n.d. ; Wu et al., 2013). The user is thus able to sense and visualise information about his environment that they would not be able to detect without the use of *AR* (Kipper and Rampolla, 2013). Based on these definitions it can be said that *AR* has the three following characteristics (Kipper and Rampolla, 2013) :

1. *AR* combines real and virtual information;
2. *AR* is interactive in real-time;
3. *AR* operates and is used in a 3D environment.

The first *AR* applications date back to the early 1990s and were mainly heads-up-displays developed for military purposes (Hosch, n.d.). Since then, *AR* applications have been developed and used in various domains (Craig, 2013a). For instance, *AR* is commonly used in the gaming domain to add environmental information to players' viewpoints but *AR* applications were also developed for smartphones to display environmental information (addresses, streets, restaurants, etc.) to users (Hosch, n.d.). In the left part of figure 2, an example where *AR* applications are used for gaming is displayed. The right part of the same figure shows an example of an *AR* application that helps users to see points of interest that are available near their location.

Figure 2: AR application which supports physical interaction with a computer game (left), application that allows users to see what restaurants are available near their location (right). Source: Craig, 2013a

Usually, *AR* devices are based on the two following steps (Craig, 2013a) :

1. In the first step, the application needs to find out the current state of both real and virtual worlds;
2. In the second step, the application must display the virtual (computer-generated) information on top of the real-world information in a way that allows the user to integrate that virtual information as part of his physical environment. Then, the application moves back to step 1.

To support those steps, *AR* devices have to include three main components (Craig, 2013a): sensors, a processor, and a display. The sensors are used to determine the real-world state. Those sensors usually include sensors for tracking, sensors for gathering user input data, and sensors for gathering environmental data about the real world. A processor or computing system is needed for processing the sensor data, carrying out the application tasks, and generating the signal driving the display. Finally, the *AR* device includes a display suitable for integrating the virtual information and the real-world information (Craig, 2013a).

Mixed reality (MR) is often used interchangeably with *AR* although those terms have slightly different meanings. *Mixed reality* actually defines a broader interpretation than *AR*, including both the digital world and the real-world as a whole (Craig, 2013b). Although *MR* has really gained in popularity these last years, there is no common understanding of what *MR* actually is and experts thus have different definitions (Speicher, 2019). In that way, lots of experts define MR as a more advanced version or evolution of *AR* that allows for a more advanced understanding of the user environment (Speicher et al., 2019). Real-world information and digital information are mixed in a way that allows users to interact with virtual objects and virtual objects to interact with the surrounding environment (Craig, 2013b; Speicher et al., 2019). It is that ability to have interactions with the virtual world that makes the difference with *AR*. In fact, all *MR* applications are *AR* applications but the reverse is not always true (Craig, 2013b). In this interpretation of *MR, MR* can be made possible only through the use

of specific devices. The Microsoft HoloLens lies in this range of *MR* devices (Speicher et al., 2019).

3.2.2 Simultaneous Localisation and Mapping (SLAM)

Simultaneous localisation and mapping (SLAM) addresses the problem of placing a device into an unknown environment in order to perform localisation of the robot and mapping of its environment at the same time. The device thus has to produce a map of its environment and simultaneously localise itself within that map (Durrant-Whyte and Bailey, 2006). Although this problem has been solved theoretically and conceptually it still poses some technical challenges nowadays. Robust methods have been deployed for mapping static and structured environments but the mapping of dynamic and unstructured environments is however still an open problem. At first, *SLAM* was used mainly in the robotic domain but nowadays its application has extended to other domains such as in navigation and odometry for virtual or *augmented reality* (Thrun, 2007). Figure 3 displays a *SLAM* device that is mainly used for indoor mapping.

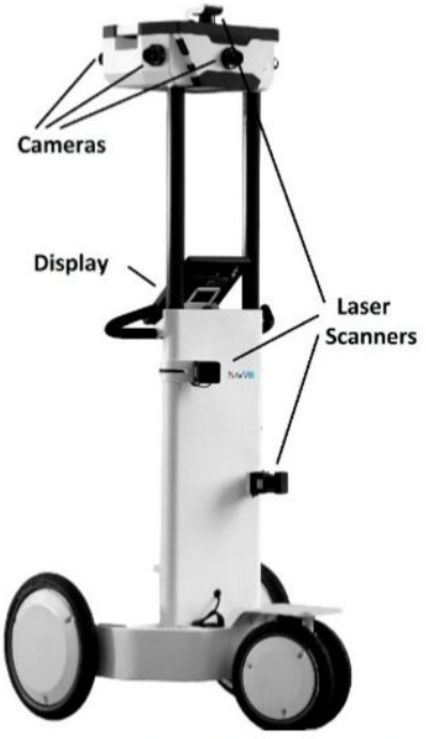


Figure 3: Example of SLAM device. Source: Lemmens, unpublished

4

Methodology

The following chapter consists of the methodology developed for implementing the desired product and thereby achieving the objectives of this project. This chapter shows the essential components our product is made of and how those components interact to form the *information system* as a whole.

4.1 Hardware & Software Prerequisites

The hardware required to implement the methodology described in the next sections is as listed below:

One or multiple Microsoft HoloLenses are used to run the Explorer application. This one sends meshes to the Navigator application and supplies an *augmented reality* overlay to the user. Microsoft HoloLens are smart glasses developed by Microsoft (see figure 4). It consists of a head-mounted display unit featuring an *inertial measurement unit (IMU)*, a depth camera with a view angle of 120°×120°, a video camera, a microphone, audio speakers, a light sensor and four "environment understanding" sensors. The lenses are located in the visor, at the front part of the unit. The unit also contains an internal battery that lasts 2 to 3 hours in active use and two weeks in standby time. It also includes a finger-operating device for interface scrolling and selecting, called the Clicker. This device is paired with the headset through Bluetooth 4.1 Low Energy wireless connectivity. In addition, the HoloLens also features IEEE 802.11ac Wi-Fi. The whole system is worn on the user's head and is connected to an adjustable headband. We chose to use this device because it has the ability to provide users a larger amount of information about their environment while they are executing their tasks allowing them to keep at the same time visual contact with that environment (Arif, 2019).

Figure 4: Microsoft HoloLens. Source: HoloLens 2, n.d.

Unity is a cross-platform game engine developed by Unity Technologies and downloadable for free from the Internet. It offers users the possibility to build 2D and 3D virtual reality or *augmented reality* games but it also has applications in other domains outside gaming as in film, automotive, architecture, engineering and construction domains. The programming language used in Unity is primarily C#.

An existing codebase was provided by *CGI* as part of Bart Peter Smit' s ongoing thesis. This showed how to communicate meshes from a HoloLens to Unity.

A desktop PC is needed to run the Navigator application. This computer links the HoloLenses together and performs the necessary data processing. A laptop computer is sufficient for achieving these tasks.

A Microsoft Azure Queue is used for providing asynchronous communication between HoloLenses and the hub over the Internet.

A Microsoft Azure Spatial Anchors account is used for creating a common coordinate system for multiple HoloLens users.

4.2 Setup

As stated in the previous section, an existing codebase was provided by *CGI*. This codebase contained around 2.000 lines of code, linking together Unity, the HoloLens, and the Azure Cloud. To answer the research questions and to fulfill the needs of the clients, two applications were developed: one application used for the HoloLens, and one for a desktop environment. These applications are called Explorer and Navigator respectively.

Both of these applications are developed within the Unity engine. Besides the source code, both projects contain features like shaders, materials, 3D models and 2D sprites. In this report, we will focus almost exclusively on the codebases within these projects, but it should be noticed that Unity Projects contain more than just code.

The codebase of the Explorer application focuses on data acquisition, sending this data to the Navigator app, receiving data from the Navigator, and visualizing all of this within an *AR* environment. It also takes care of local to global coordinate transformations.

The codebase of the Navigator application, on the other hand, serves as a central hub. Only one application is meant to be running at any time. From multiple Explorer applications it receives data like the HoloLens' position and orientation, and the meshes it just created. It then sends data back to the Explorer, like the position of other Explorers. The Navigator takes care of properly storing, classifying, and visualizing the data. Scanned scenes created by continuous listening to the HoloLens can be saved afterwards, and loaded in an emulator for later inspection. These functions are also further explored, in section 4.4.

Both codebases must share a large amount of the same code. For example, if a message becomes too large, the message is split into multiple smaller messages. The receiver of those messages needs the exact same classes in order to know what this message means, how this split is performed, and thus how to reconstruct the original message. A third codebase was thus created, containing all general code of our messaging system, and the data model, to ensure that these elements remained consistent. This codebase was called the Backbone codebase as it can be seen as the backbone of the Navigator and Explorer. The whole architecture is shown in figure 5.

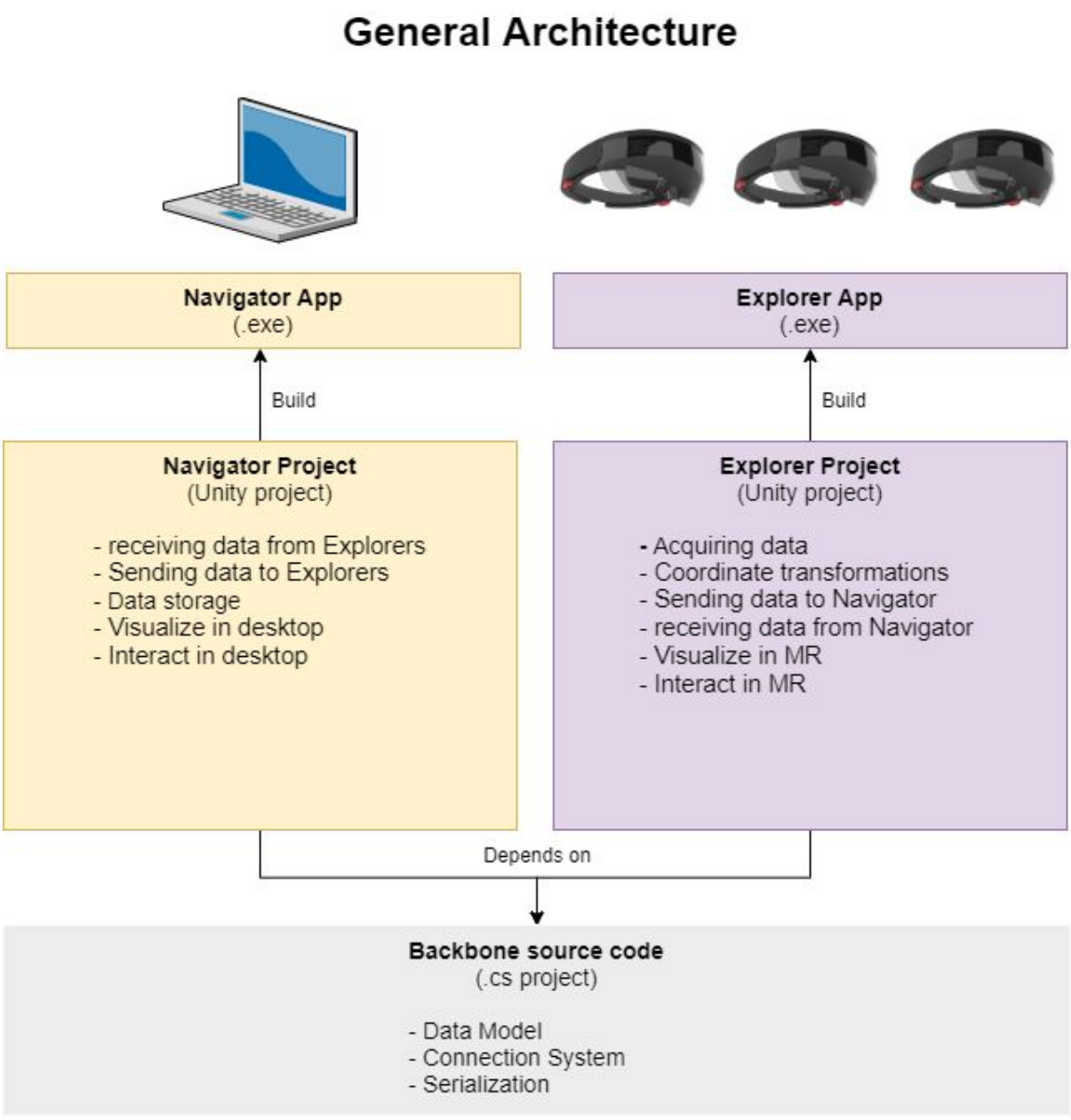


Figure 5: our general setup. Source: Author

4.3 Software architecture

The inherited codebase from CGI was developed as an exploratory proof of concept, which made it difficult to expand upon. For this project, a robust and scalable base was needed to ensure an implementation of all the envisioned features during the development phase, and for facilitating future changes.

4.3.1 Code Architecture & Design Patterns

To solve the troubles related to the existing codebase, we rewrote it from scratch. To avoid rebuilding a similarly complex codebase, proper code architecture, and proper utilisation of programming design patterns were needed. Some literature research in that field was then required. The most helpful books were “Design Patterns: Elements of Reusable Object-Oriented Software" of Gamma et al. (1994), and the more recent "Game programming Patterns” of Nystrom (2014).

Code architecture and Design patterns exist mainly for two reasons. First, it aims at making any script more adaptive to changes. Whenever a new feature needs to be implemented to any existing codebase, usually, the code first needs to be disrupted. Space is created for the new feature as it were. The more adaptive, modular, and decoupled a codebase is, the less disrupted a codebase needs to be in order to add functionality.

The second reason has to do with the cognitive limitations of programmers working on code. It is inefficient if a programmer needs to understand the inner workings of an entire codebase only to add one small feature in one very specific domain. What would be better is to split up the code into functions, sections, classes and namespaces, and the idea is that these areas should have as little as possible to do with other areas. This makes sure a programmer can focus on just that class, or just that function, without needing to worry about auxiliary troubles. This process is called "Decoupling strategies" in the Design Pattern books.

This knowledge helped us in creating a modular and highly reusable code. This focus on code design highlighted that a lot of features needed special attention in order to be properly scalable and modular. For example, we created a serializable base class, which generalised saving, loading, sending, receiving procedures for all data in our system.

The Design Patterns book also helped understand why the original codebase was not suitable for change. It contained heavy usage of something akin to the “Singleton” design pattern, which, in short, aims to make everything accessible anywhere within the code. This may sound convenient, but an overuse of static and global state classes results in non-modular code, and conflicts with the two aims of Design Patterns mentioned above.

Again for modularity purposes, we also aimed at linking our product to the Unity engine to a minimum. In that way, the implemented codebase could be easily converted to some other 3D visualisation environment. This had been a challenge and in the end, our product heavily relies on the 3D library of the Unity Engine for elements like Vectors, Matrices, and Meshes.

If a conversion to another platform has to be made in the future, substitutes will need to be found, or these features will need to be implemented ad hoc.
We managed to implement an important decoupling strategy concerning Unity: the Unity visualisation classes are kept separate from the data storage classes themselves. This way, scenes can be loaded and saved without Unity: it is just there for visualisation purposes, and for providing a 3D library.

4.4 Backbone

The Backbone is the body of code present in both the Navigator and Explorer apps. It contains two main elements: the data model, which stores the data captured by the HoloLenses in a format that suits our needs, and the messaging system, which converts this data into messages that can be transferred using Azure queues. This chapter addresses the *why* and *how* the Backbone codebase was created. This is explained by describing the particular technical challenges that required its implementation, which is followed by the several solutions used to tackle these issues.

4.4.1 Technical challenges

The HoloLens, as well as Microsoft's Azure systems, have a lot of peculiarities. Documentation of these systems is sometimes sparse. This meant that during this project we encountered several technical challenges which are related to how data is transferred. The first problem was that transferring data over Azure queues requires serializing it to a binary format. Due to limitations in the HoloLens' .NET libraries, we were unable to achieve this using conventional means, such as by using the C# built-in serializer. The second problem was that Azure queues have a size limit of 64 kB per message which prevented us from sending meshes with large amounts of vertices.

4.4.2 Solutions

Several low-level implementations had to be made to properly work around many peculiarities of the HoloLens and Azure. The term "Low-level" is used but it should be noted that the code of these low-level elements is written in C#, which is a high level programming language.

The code created to overcome the stated challenges turned out to be needed on both the Navigator and Explorer apps, and should therefore be a separate body of code. This culminated to about 22 classes, organised within two modules: "Azure", dealing with connections and serialization, and "Data", dealing with the data model. These two spaces together is what we now refer to as the Backbone. The following paragraph will cover the rough shape of both of these modules, and the design decisions behind them.

Data
A data model was necessary to manage all the different elements we wanted to keep track of, record, and share, in both the Explorer and the Navigator apps. Explorers, for example, want to keep track of the positions of other Explorers, as well as the position of pings placed by the Navigator.

The model starts with a Scene class. This class represents a full scenario of one or multiple HoloLenses walking through an environment, and all data captured and generated along with that. In most scenarios, there will be one scene, but in the occurrence when two HoloLenses are not part of the same environment, or not properly synchronised, multiple scenes can be created.

A Scene contains three lists, holding Explorers, MeshUnits, and Pings respectively. Each one of these lists is accompanied by a dictionary, for instant retrieval based upon a certain key. The Explorer is a data model used to internally keep track of where one Explorer-user is, including a history of where it has been. A MeshUnit contains a mesh, alongside metadata about when this mesh was created, and what the classification is. A Ping represents a point-like feature in the scene. These are sometimes added as a result of classification, like a “door”, but they can also be added by the users themselves, in the shape of a “move here” command for example.

The decision to make all of the captured and generated data hierarchical children of a single Scene as a root, and to then allow multiple scenes, enables important interactions. This, for example, makes it possible to use the scenes as more or less “layers”. We can emulate and overlay an earlier captured scene over a scene being captured in real-time, without issues on the back end. The new scene can then be compared to the old one. Also, by making this data model mostly separate from Unity functionalities, we can load, capture and save scenes without the need of the Unity engine.

Lastly, to make the connection to the Azure module, all classes within this data model implement the SerializableData base class. This enables all data to be saved, loaded, serialized and deserialized from raw byte arrays in a recursive fashion.

This is now why scenes can be saved as a singular binary file: the scene just asks its Explorers to serialize themselves, and each Explorer then calls its internal components to do the same, and so on. To accomplish this serialization we created our own binary formatter as a workaround for the technical challenges mentioned above.

Azure

Azure covers the full process of sending the data of our data model from one device to another. This is organised into the classes Connection, Message, and Poller, among others. Messages can contain any object derived from a certain SerializableData class, to ensure the data can actually be converted to raw bytes. Azure also contains the low-level implementations mentioned earlier of the BinarySerializer and the chunk system.

This means that the system is technically versatile enough to be able to send an entire scene through the azure cloud in an efficient manner. We never actually use this, but it could be useful in the future to enable a secondary Navigator to receive what the first Navigator is seeing.

To resolve the issue of the 64 kB per message limit of the Azure Cloud, we created a custom Chunk loader: large messages are split up into several chunks, sent through the queues, and reassembled on the other end. This system is made so that it can handle messages of any size.

4.5 Navigator

In this section, the implementation of the Navigator project is discussed. Its main function is to act as a control centre which can combine environmental and positional data from multiple Explorers, as well as automatically classify and visualise this data. It also supports sending information and commands to Explorers. Its main features are first covered. From there, the classification procedure is explained with technical details. Lastly, topics like data visualisation and data interactivity are covered.

4.5.1 Main features

This application serves as a central environment to which multiple Explorers are connected. It runs on a desktop environment, but could be ported to other devices such as tablets or smartphones. Only one Navigator application is meant to be running at any time.
The Navigator offers two ways of receiving data: streaming and emulation. During streaming, the application connects to all available Explorer applications and starts receiving data like the HoloLens' position, orientation, and the meshes it created. The Navigator then sends data back to the Explorer, such as the position of other Explorers. The Navigator records and stores all received data, semantically enriches it (separating floors, ceilings, walls and furniture), and visualises it to give the user an overview of all data gathered. A scene gathered during streaming can be saved to a binary file.
The emulator allows us to load a previously created scene and to play it back the way it was captured in the first place. This allows users to re-evaluate a recorded dataset at a later time. If the classification was turned off during the actual capturing of a scene the emulator can also be used to re-apply the classification.
Streaming and emulation are not mutually exclusive: both can be active at the same time. Technically, any number of streams and emulations can be active, but for the sake of keeping the application clear to use, we have limited the use to only one active stream and one active emulation. Doing so could be useful in cases where a user would want to compare an old scan to one that is currently being captured.

4.5.2 Data processing: Semantic enrichment of meshes

Both the streaming and emulation modes have the option to classify the meshes into walls, ceilings, furniture and floors. This semantic enrichment aims to make the meshes acquired by the HoloLens more readable. This sub-chapter consists of two main sections. The first one consists of the methodology followed to add basic semantics. Basic semantics were added by classifying the acquired meshes. The second section describes the methodology developed to add more complex semantics to the model.

Classification of meshes

In this section, the meshes generated by the HoloLenses are enriched with semantics and thus classified as wall surfaces, ground surfaces, ceiling surfaces and furniture. To achieve this, the first step of the method consists of the clustering of the meshes into different surfaces based on the surface normal. Then, a label (ground/wall/furniture/ceiling) is assigned to each surface by comparing some properties of that surface (height of the surface, area and orientation) with dynamic threshold values. The overall approach is based on the research of Shi et al. (2019).

A) Region-growing

Region growing is an algorithm used for simple shape detection (Ledoux et al., 2019). The aim here is to detect the different planar surfaces into an input mesh so that the surfaces can later be classified. The algorithm implemented starts from a random triangle, known as the seed triangle, of the input mesh and computes its normal. The first planar region starts from that 'seed triangle' and other triangles are added to that region by investigating the neighbours of the starting triangle. Those neighbours are considered as candidate triangles. Their normal is computed and if the angle between their normal and the normal of the seed triangle is small (below a preset threshold), the neighbours are added to the planar region and their neighbours become the next candidate triangles to be investigated. Once no more candidate triangle fitting with the current region can be found, the algorithm starts growing a new region from a new seed triangle. The algorithm stops when all triangles of the mesh have been assigned to a region.

For accessing the neighbours of a given triangle in the mesh, a dictionary structure is implemented before executing the region-growing algorithm, for optimisation purposes. The dictionary structure thus maps from the mesh vertices to their incident triangles. Figure 6 summarises the overall methodology that is used at that step.

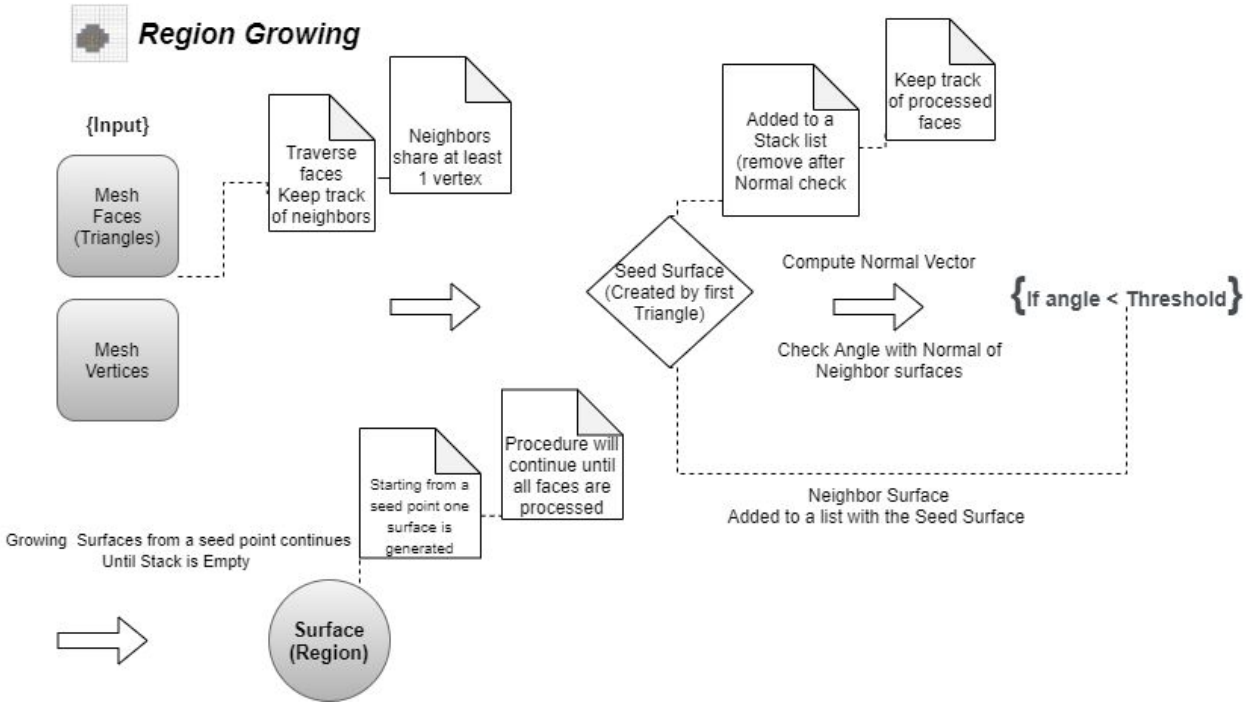


Figure 6: Region Growing algorithm leading to surface generation from a mesh. Source: author

B) Semantic assignment

Once a planar surface is detected in the input mesh, it can be classified between ground, ceiling, wall and furniture surfaces. This classification is done based on the following properties of the surface: the area, the height and the orientation (parallel or perpendicular to the ground). Those properties are then compared with four thresholds :

- Area threshold: below that area threshold, surfaces are classified as furniture. Indeed, furniture surfaces are usually smaller than wall, ground or ceiling surfaces.
- Angle threshold: by computing the angle between the normal of the ground surface and the normal of the surface to be classified, it is possible to determine whether the surface is parallel to the ground (angle = 0) or perpendicular (angle = 90). With this information, it is possible to make the distinction between wall surfaces (perpendicular to the ground) and ground or ceiling surfaces (parallel to the ground).
- Height ceiling threshold: above that height threshold, surfaces are considered to be part of the ceiling.
- Height ground threshold: below that height threshold, surfaces are considered to be part of the ground.

The whole classification process based on thresholds is shown in figure 7.

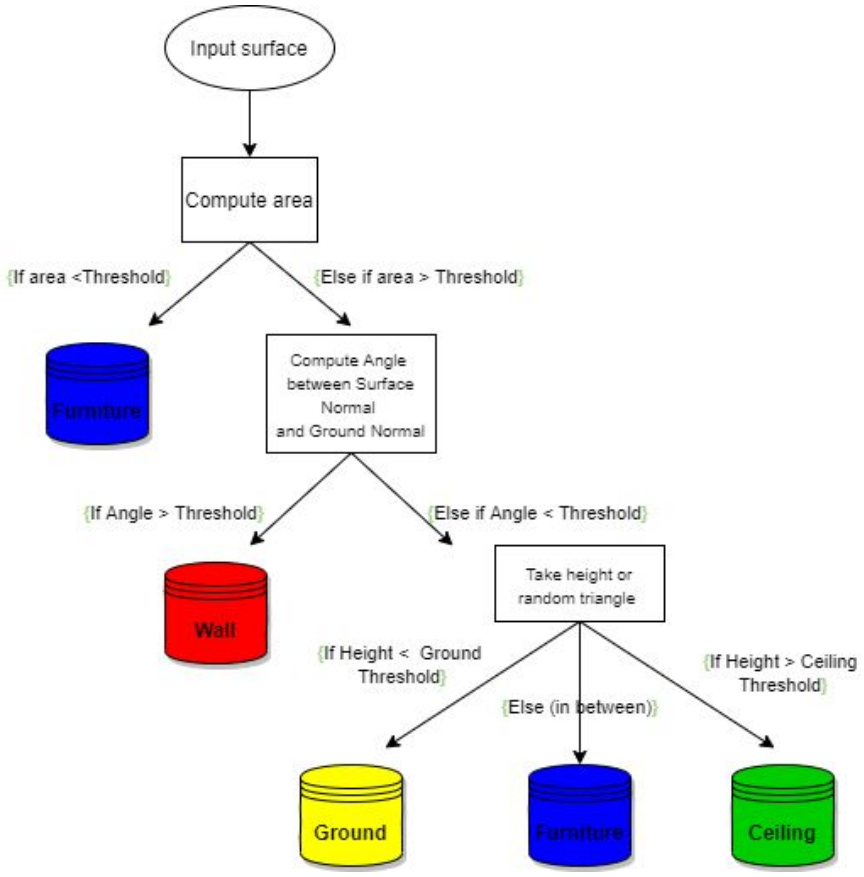


Figure 7: Classification of meshes based on the different thresholds. Source: author

In order to make the meshes classification work for any kind of room regardless of its dimensions, the three first thresholds used in the classification have to be input-dependent. This does not regard the angle threshold as perpendicularity and parallelism properties stay the same regardless of the input data. To make the other thresholds input-dependent, the meshes generated by the HoloLenses are used to compute them beforehand. In this way, the average minimum height and the average maximum height of all meshes are respectively used as height ground threshold and height ceiling threshold. Finally, the area threshold is based on the average area of the parts of the meshes that are located in between the two other height thresholds, thus excluding the ceiling and ground surfaces from the computation.

C) New meshes generation

Once the planar surfaces of the input mesh have all been classified, all surfaces labeled with the same semantic are assembled together and a new mesh is generated per semantic. Based on its label, a color is assigned to the mesh for visualisation purposes and the newly generated meshes replace the input mesh.

D) Other approaches tested

Other approaches were tested before implementing the region growing method but they were not chosen as a final approach for quality or time efficiency purposes. Those approaches are described in the coming paragraphs.

- Semantic assigned per triangle face

This method makes use of the same classification based on thresholds than the region growing approach. However, instead of detecting planar regions into the input mesh, all triangles of the mesh are classified independently of their neighbours based on their area, height and orientation properties.

- RANSAC plane detection

In this approach, the detection of planar surfaces into the input mesh is also performed but it is based on RANSAC plane detection (Ledoux et al., 2019). The algorithm starts from a random triangle of the input mesh and instantiates a plane instance from the triangle vertices. Then, for every other triangle with a normal parallel to the plane normal, the algorithm checks whether they fit with that plane or not. If it is the case, the triangles are added to the instantiated plane. When all triangles have been checked, the algorithm creates another plane instance and repeats the process. This procedure continues until all triangles in the mesh have been assigned to a planar surface. The output planar surfaces are then assigned a semantic based on the surface properties (area, height and orientation).

Door Detection

The door detection procedure starts after the classification was performed for a certain number of meshes. The input needed for door detection is the surfaces classified as walls and the track points of the HoloLens users. Figure 8 displays the overall methodology. This methodology was based on the research of Flikweert et al. (2019).

A) Planar partition

In that step, each surface classified during the previous step as a wall is turned into a planar partition. In order to remove outliers (misclassified wall surfaces), we only keep the "wall-like" planes. Those are actually the planes having a normal angle of around 90 degrees.

B) Merge partitions

The next step consists of checking if some planar partitions created in the previous step actually correspond to the same. If two planar partitions are considered to be the same, an average plane is computed from those two planes. That average plane is kept into the planar partitions list and the two input planes are discarded.

C) Find doors locations

In the last step, door detection is performed. This procedure consists of the following steps: i) First, we take all the points of the HoloLens users track (as a list of 3D Vectors, (x,y,z)). Each point is connected with the next point in order to create a line segment. ii) Then for each line segment, the algorithm checks whether it intersects with any of the planar partitions created in the previous step. iii) If any intersection point is found, the algorithm needs to check whether that intersection point is located in between two coplanar wall planes (in which case it is a door) or if it is at the left or right of a wall plane (can be a point in a corridor next to a wall). For doing that, the points of the plane are projected to the ground and two convex hulls are computed; one with only the points of the wall plane and one with those points plus the intersection point. If the intersection point is in between two walls which means that the user passed through a door then the two convex hulls will be the same. If the intersection point is at the side of a wall (corridor) then the convex hull that contains that point will be larger than the convex hull without it. In short, we compare the areas of the two convex hulls. If they are the same (within a small tolerance) then a door is spawned at the intersection point position

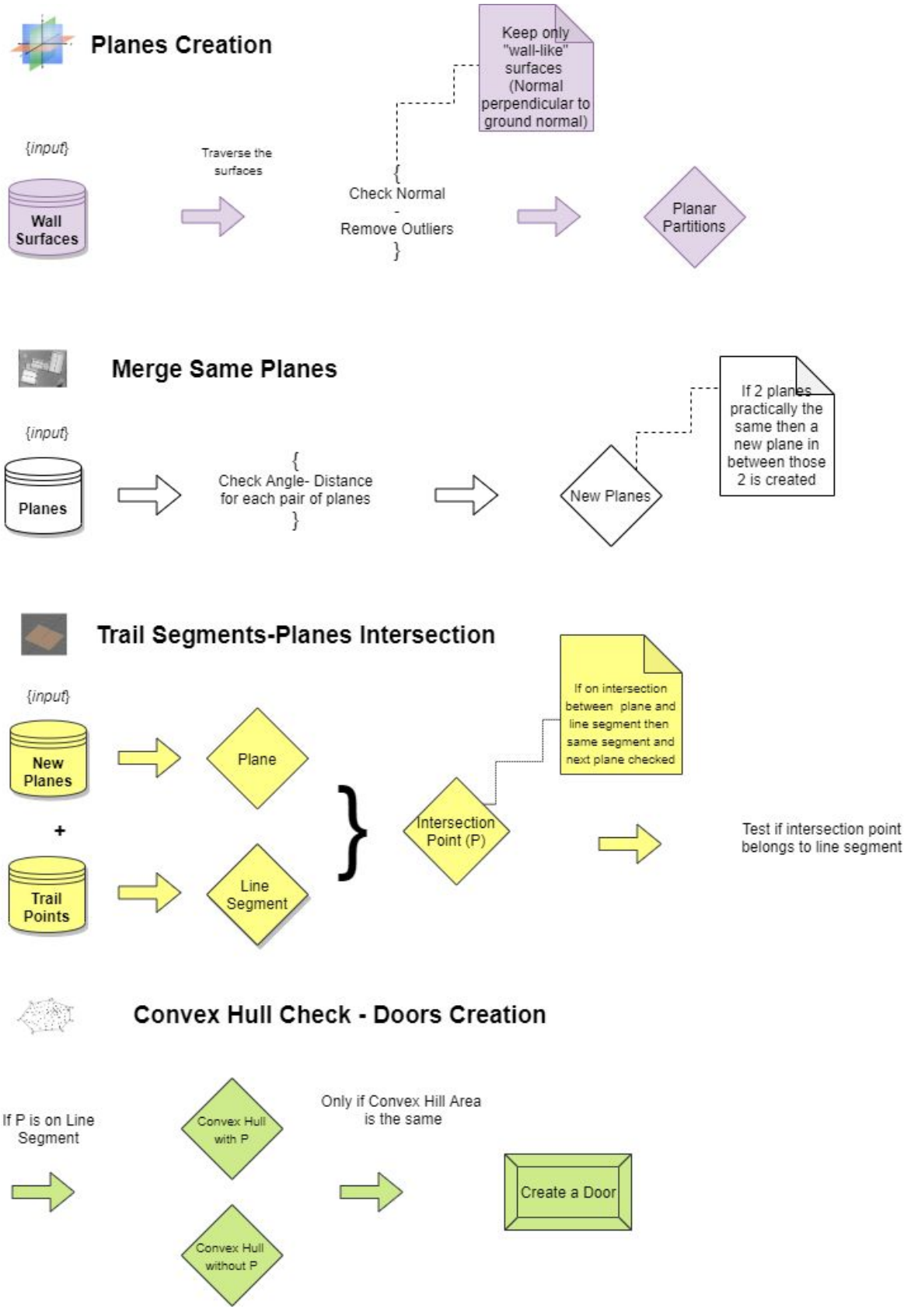


Figure 8: Overall methodology for door detection. Source: author

D) Other approaches tested

Another approach was tested before the main approach described above was implemented. The main concept of that approach was similar to our main method; finding intersections between walls and users track. The main difference is that the vertices of the wall meshes and the track of the HoloLens users were first projected to the ground plane. In theory, when the wall's points are projected to the ground, it is then possible to perform line detection to detect the wall's footprint on the ground. The key was then to find the intersections between those detected lines and the users track as those intersections would actually be door positions.

Two methods were implemented for line detection of wall footprints:

- RANSAC line detection

RANSAC works by taking two random points from the wall projected points and instantiates a line (Ledoux et al., 2019). Then for every other point in the wall points, the algorithm checks the distance between the point and the instantiated line. If that distance is smaller than a user-defined threshold, then the point is assigned to that line. After all wall points are assigned to one line instance, then the detected lines are the ones with the biggest number of fitting points.

- Hough transform

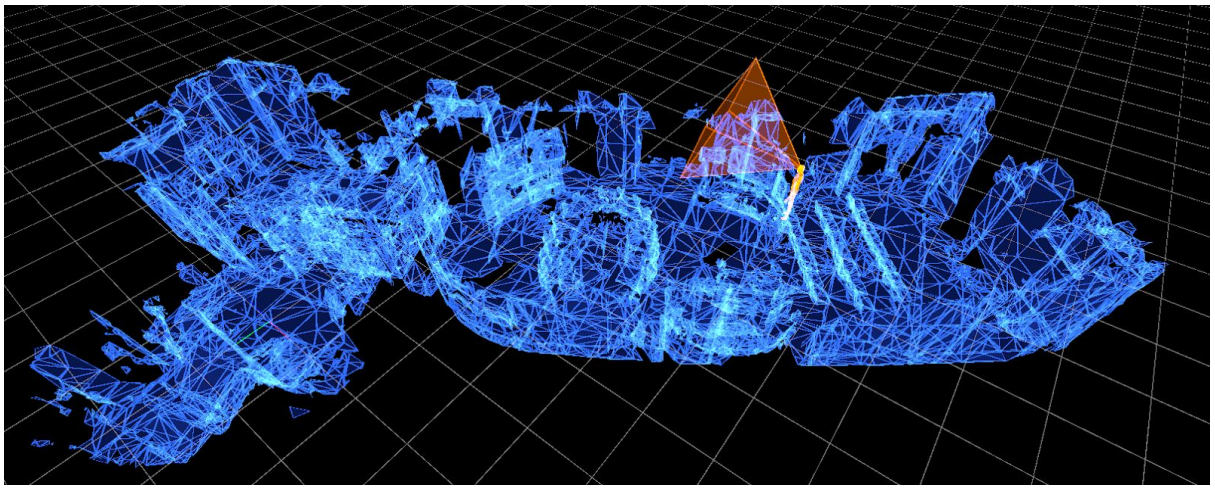
The hough transform line detection is based on a voting mechanism (Ledoux et al., 2019). The algorithm works on generating several line instances by using all possible parameter combinations that give a line instance. Then for each line instance generated, the algorithm counts the number of wall projected points that fit with that instance and that number is stored as a vote. The line instances detected are the ones with a number of votes higher than a preset threshold.

This approach is thus point cloud based, unlike the other methods that are using the derived meshes. The procedure after the line detection is simply to check the intersection between the detected lines and the user tracks. The algorithm then spawns a door at those intersection points. The main problem of those approaches is that if some points are misclassified as wall points (outliers) then the lines that are created do not always correspond to wall footprints, so the intersection between trail line segments and RANSAC lines does not lead to door detection.

4.5.3 Data Visualisation

During all of these features, it is vital that we can actually look at the data. In order to correctly visualise meshes within the Unity engine, we had to look into materials and shaders. Shaders in Unity use the Higher Level Shading Language, or HLSL, common for DirectX applications. We had to write a couple of these shaders ourselves, mostly based on examples, to enable us to use vertex and edge colouring, as well as transparency.

With the materials and shaders in place, we wrote a couple of small classes to enable us to switch between kinds of visualisations for meshes. The four main visualisation methods implemented are technical view, time view, point cloud view, and classification view (see figure 9). Technical view shows the meshes as a transparent wireframe. This helps when the user wants to actually read the topology of the meshes. Time view colourises the meshes based on their age: the older a mesh gets, the more it shifts to darker tones. This can represent the validity of a mesh in a dynamic environment. Older meshes are less valid in a sense, as there is a high change that the ground truth underneath is not the same anymore. The point cloud visualisation is useful when overlapping datasets, and might be more familiar and readable to Geomatics experts. Finally, we have implemented a visualisation based on the classification of meshes. This can make a scene overall more readable, especially since the user can toggle off certain classifications. Interpreting visualisations is subjective, but by only showing walls it becomes arguably better to read the individual rooms of a building.



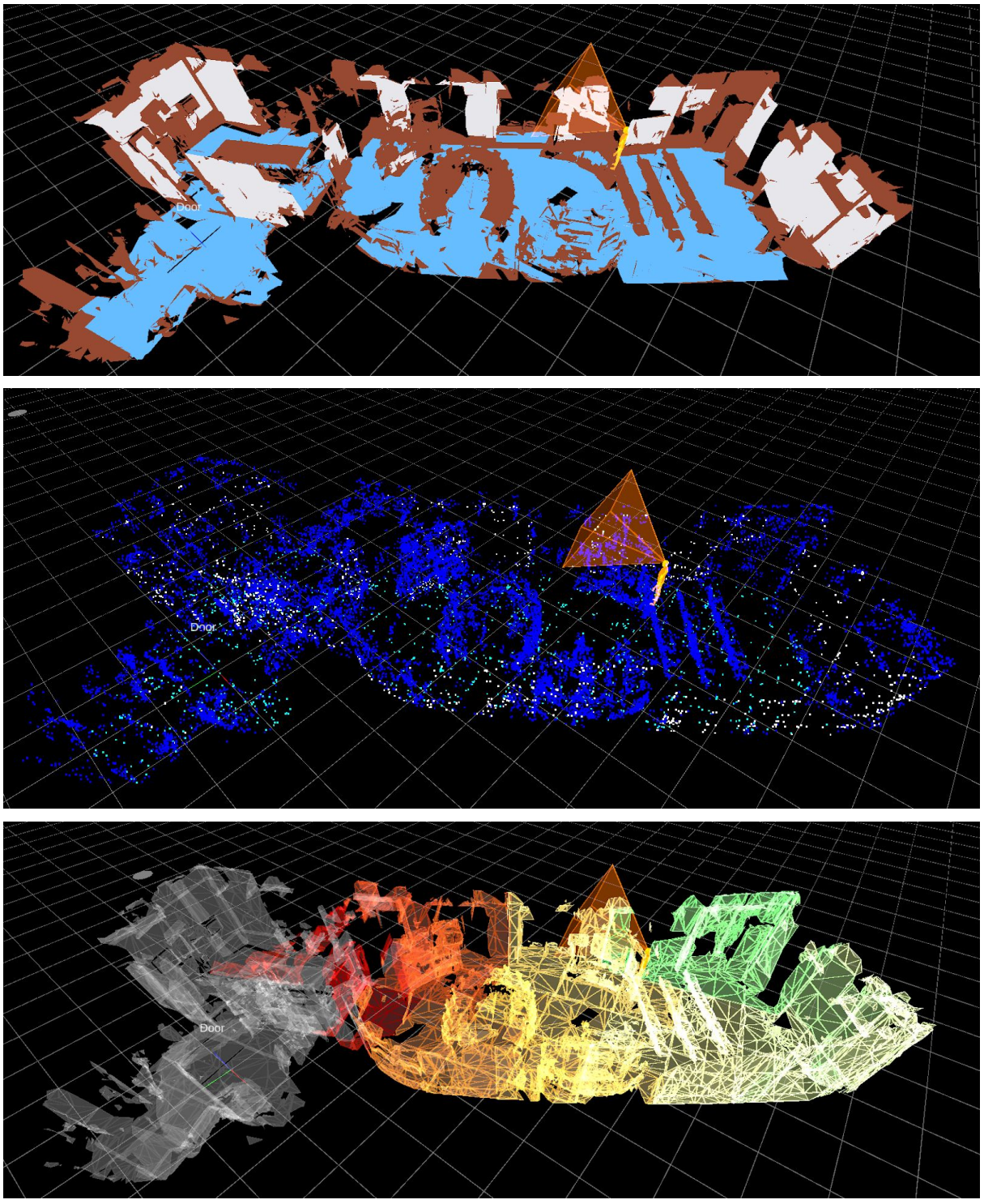


Figure 9: Wireframe view, classified view, classified view as a point cloud, and time view. Source: author

4.5.4 Data Interaction

The point of capturing and visualising data in real-time, is to directly make the data usable as a means of communication in the field. This means that the users of both the Navigator and Explorer should be able to interact with the data in some manner.

The main way to do this within the Navigator, is by creating and sending “pings”. Pings are a combination of a point in space, a time, a type, and a string containing an optional message. These can be created with the mouse wheel, and while streaming is active, are directly sent to all hololenses connected to that particular scene. This way, the user of the Navigator can signal the positions of objectives to go to, or calamities to stay away from.

4.5.5 User Interface

To create a user interface for user interaction, we made use of the components within the Unity UI toolkit. To make it truly interactive, the interface had to be very intuitive and it had to constantly give feedback on what the user is doing. The interface needed to have a menu bar framework from which all the possibilities of the system could be managed. For the resulting components of the UI, look in the results section (chapter 5).

4.6 Explorer

In this section we discuss the implementation of the explorer module, which is the application that runs on the HoloLens. Its purpose is to create a 3D mapping of an environment and to provide *mixed reality* information to users to support tactical decision-making. The application consists of multiple modules, the most important of which are described below.

4.6.1 Shared coordinate system

Each HoloLens, being a *SLAM* device, is capable of locating and orienting itself relative to the space it maps. While the device is active, location and space are recorded relative to the session’s starting location. Consequently, every device records data in its own coordinate system. To allow data from multiple devices to be combined this means we first need to unify their coordinate systems. We do this by establishing a point in space that serves as a common point of reference. If each device can position itself relative to the common point, it can then be used as a shared coordinate system. This allows multiple devices to map the space and be tracked at once. Figure 10 illustrates the principle behind the coordinate system transformation.

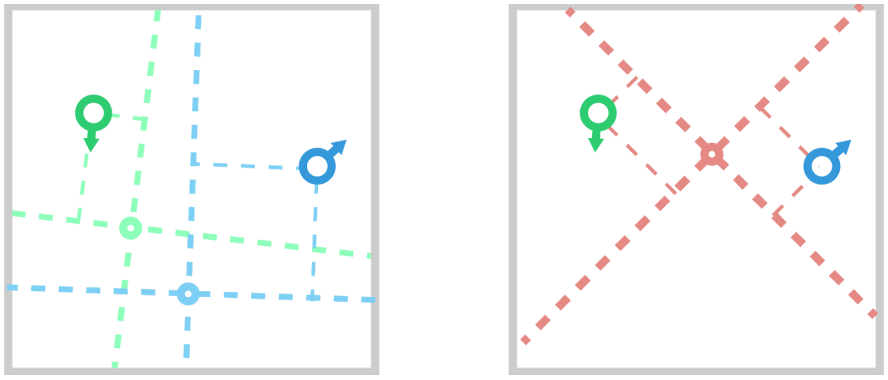


Figure 10: Left: two HoloLens users in their own, separate coordinate system. Right: Two users after establishing a shared coordinate system. Source: author

We accomplish this by using a spatial anchor. A spatial anchor is essentially a description of a point space. Much like how humans can describe the position of an object or themselves relative to a landmark, e.g. "*The bus stop is 100m to the left of the statue"*, or "*The wallet is on the far end of the table*", SLAM devices can use spatial anchors as a common reference point for describing their positioning and environmental data.

Azure Spatial Anchors

Luckily, Microsoft provides a cloud service called *Azure Spatial Anchors (ASA)*. This service is tightly integrated with the HoloLens hardware and allows us to create and store spatial anchors through its Unity *SDK*. Although there are many benefits to this system, such as its ease of use and reliability, it also has some significant downsides. The most important downside is that it is closed source. This means that we cannot know for certain how *ASA*s work. However, it is still possible to make an educated guess about how it functions.

The primary component of an *ASA* is a geometric description of the space in the form of a point cloud. This is confirmed in the official Microsoft documentation (Frequently asked questions - Azure Spatial Anchors, 2020). In this document they also state that a sparse version of the point cloud is used and they provide the following figure as an example.

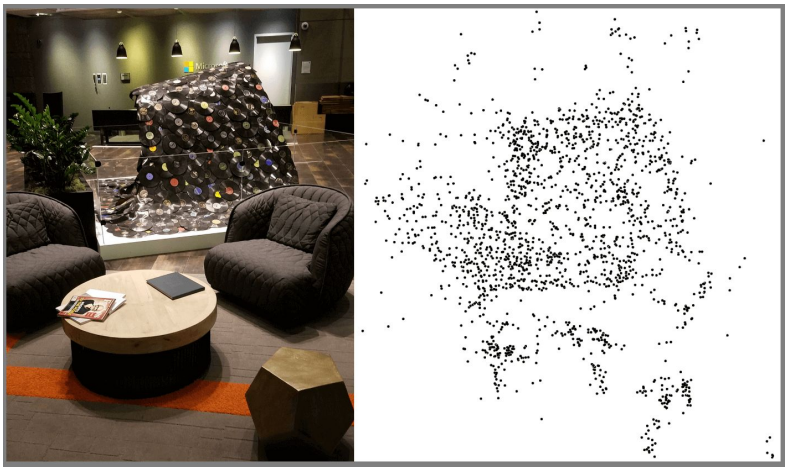


Figure 11: Environment and corresponding sparse point cloud used for ASA. Source: Frequently asked questions - Azure Spatial Anchors (2020)

When trying to identify a spatial anchor, the HoloLens will try to match its current environment to the spatial anchor geometry by finding the transformation that minimises the difference between the environment and the anchor. There are multiple ways to achieve this. Some methods, such as the iterative closest point algorithm described in Yuan et al. (2016), only use the geometric properties of the point cloud. Others are also able to use the point cloud's photometric properties (Huang et al., 2012).

Another piece of sensor data that spatial anchors are confirmed to use is the HoloLens' stereo camera feed (Frequently asked questions - Azure Spatial Anchors, 2020). It could do this by finding key points in the spatial anchor data and checking whether these are present again when locating the anchor. There are a variety of well-established algorithms available to achieve this, such as those described by Hannah (1988).

The techniques described above are used in tandem to find the position of the HoloLens relative to the *ASA.* However, the HoloLens needs to know in advance which *ASAs* might be nearby to start looking for them. This can be done by manually specifying their identifier. However, actively watching for *ASAs* is computationally expensive. If there are a large number of them the computational cost could become prohibitive. We therefore use a technique called coarse relocalisation to roughly determine which *ASAs* might be nearby and only watch for those that are. Once again, this functionality is integrated in the *ASA SDK,* which supports Bluetooth, *WiFi fingerprinting* and GPS. Because the HoloLens does not have a GPS sensor and the use of Bluetooth fingerprinting requires specific Bluetooth beacons, we opted to use *WiFi fingerprinting*. This means the HoloLens only watches for *ASAs* which were created on the current WiFi network, with an associated WiFi signal strength that approximately matches the current measured signal strength. This method only works under the assumption that during deployment there is a constant WiFi signal available to the HoloLens. To achieve this, we propose equipping fire trucks with high strength, high frequency WiFi antennas, both to provide the Internet connection that is necessary for the Explorer to function as well as for coarse relocalisation purposes.

Coordinate transformations

In our implementation we create the shared coordinate system using a single *ASA*. After finding this *ASA,* the Explorer starts transmitting data to the Navigator. This data consists of two parts: the scanned environment in the form of meshes and the user's position and orientation. Before this data can be sent, it needs to be converted to the *ASA's* local coordinate system. This involves multiplication by two transformation matrices. We do this in an efficient way by concatenating the two transformation matrices, reducing the multiplications required by nearly half.

4.6.2 Optimisation

To make head-mounted *augmented reality* applications comfortable to use, they need to run at a high frame rate. In the case of the HoloLens, the ideal frame rate is 60 frames per second (FPS) because its display works by switching between four colour fields at 240 Hz. This means every frame can only take a maximum of 16 ms to compute. Failure to do so causes holograms to jump around, which ruins the illusion that they are actually present in the scene. It can also cause the user to become nauseated and disoriented which can be dangerous during high intensity operations. The code we received from *CGI* initially ran at around 30 FPS but dropped to 5 FPS when performing coordinate transformations or sending data. To improve the application's frame rate we use two techniques: coroutines and multithreading.

Coroutines

A limitation of the Unity Engine is that all computations that interact with the engine need to be performed on the main thread. Rendering the actual image is done on the GPU, but to do so it needs to be directed by the CPU. This means that if the main thread is occupied by some other background process, it won't be able to direct the GPU to render new frames, causing the frame rate to drop. To solve this problem, we use a method called coroutines.

Coroutines are a type of function that is able to halt its operation until certain criteria are met, after which it can continue where it left off. This means that we can direct an operation to only run during the part of the frame after the scene has already been rendered. An example of where we use this method is when updating the minimap.

A downside of using coroutines is that when a large part of every frame is dedicated to other processes, operations can take a long time to complete. This means that coroutines are only suitable in situations where operations do not have to complete as soon as possible.

Multithreading

For computations where it is not necessary to interact with the Unity Engine, it is possible to use multithreading. The HoloLens CPU has four cores, which each support a single thread, meaning it is possible to perform 4 calculations in parallel. One of these threads is used by the engine, leaving 3 threads for other computations. Multithreading is thus ideal for increasing the frame rate because parallel computations do not interfere with rendering. We use this method to perform the mesh coordinate transformations and to send data to the cloud, which were the principal causes of the low framerate in the original code.

Multithreading does come with its own set of limitations. For example, parallel operations cannot modify the same data at the same time, nor can they directly interact with the scene.

This means that we had to use additional data structures such as queues for inter-thread communication.

4.6.3 User interface

As stated in the user interview, assisting firefighters in the field is not just about how data is processed, but also about how it is presented to them and how they interact with it. In this section, we discuss the different elements of the user interface. Screen captures of the interface in use are included in appendix C.

Menu

During operations, firefighters need to be able to add information to the environment manually. They also need to be able to filter out information which is not necessary to reduce visual clutter. For this purpose, we implemented a menu which the user can access by performing a pinching gesture with their right hand. The menu appears one meter in front of the user and follows their movement. Figure 12 shows the menu viewed directly from the front.

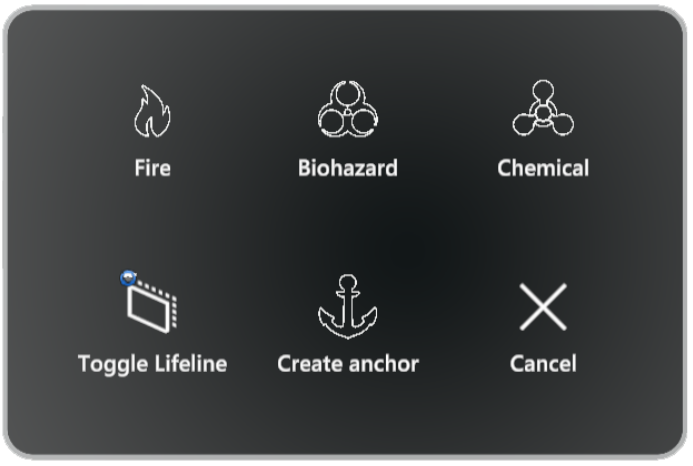


Figure 12: Menu shown head-on. Source: Author

The top row is dedicated to three different types of pings, representing hazards that the user might encounter during operations. In the bottom left, the user has the option to toggle the lifeline, which is described later in this section. The bottom middle button allows the user to create a spatial anchor. The bottom right button closes the menu. When the user places a ping or a spatial anchor, it is placed at the point where the user was pointing towards when they opened the menu. The amount of buttons were significantly reduced from the original application so as to not overwhelm the user.

Scene objects

There are two types of objects that can be present in the scene: avatars and pings. Avatars represent other firefighters, pings are manually added information, such as the position of a fire. These objects have a position in space and are displayed as two-dimensional icons that always point towards the user. We chose this way of displaying them so that objects are recognisable from any direction. The icons are shown in figure 13. The icons have

intentionally contrasting colours and have a white outline. This makes them identifiable against any background.

Figure 13: The icons for pings and users. From left to right: spatial anchor, chemical hazard, biohazard, user, and fire hazard. Source: Author.

Heads Up Display

Besides showing information in the environment through the use of holograms, a heads up display (HUD) is also provided. The information in the HUD is always in the same position in the user's view, in our case on the bottom left. The HUD consists of a minimap and textual information about the user's current height and the duration of the operation.

The minimap is a concept we borrowed from video games, where they are usually used to show the player's position and nearby objectives. Our minimap, shown in figure 14, mostly has the same purpose. It shows the user's orientation relative to their starting point and where nearby pings and other users are. If a tracked ping or user's distance to the user is further than a predefined threshold, it appears on the edge of the minimap, otherwise it appears somewhere inside it. Only pings or users that are within a predefined height range of the user are shown on the minimap.

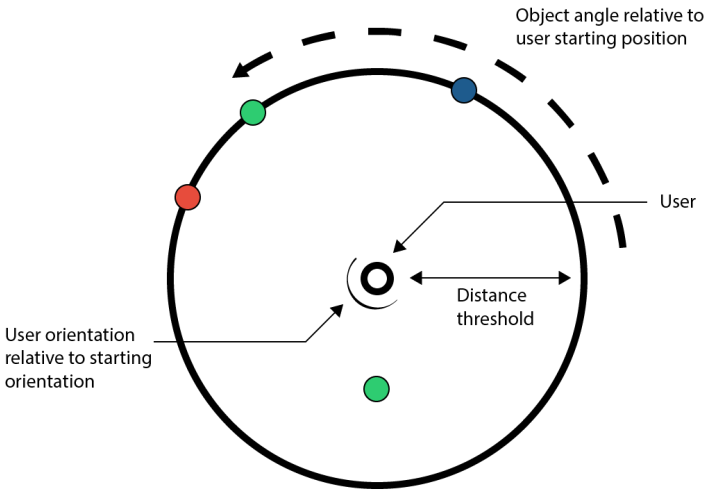


Figure 14: Minimap concept. The colours are inverted relative to how it is displayed on the HoloLens. The coloured circles are placeholders for the icons shown above. Source: Author.

Lifeline

To assist firefighters in finding their way back to a previous location we implemented a lifeline. Every n meters the user's position is recorded and a line is drawn between the position history. If the user is lost, he can then simply follow the line to get back to where he was. The line is displayed holographically and it is translated one meter downwards to prevent it from blocking the user's view.

5

Results

In this chapter, the results of this project are explained in terms of its achievements and the implemented codebases. The capabilities of the developed product are described as well as its limitations and other elements that could be improved.

5.1 Project expectations versus achievements

At the beginning of the project, using the MoSCoW method, we defined our project expectations. In the following section, we assess whether those project expectations were met.

Must haves

	Expectations	Results
1	Merging of two HoloLens point clouds to create a unified, interactive 3D map.	**Achieved**
	A central HUB where the map is visualised along with information about personnel - Point cloud or mesh display - Position / orientation of personnel	**Achieved**
3	Simple Hub to HoloLens communication (HuHo) - Place pings in map which are shown on the HoloLens - Show location and distance of other personnel	**Achieved**

Table 1.a Must haves of project (expectations and results)

Should haves

	Expectations	Results
1	Add basic semantics / information manually (points, lines, areas, volumes)	**Semi-Achieved (points)**
2	Add basic semantics automatically (ground, wall, ceiling, furniture)	**Achieved**

Table 1.b Should haves of project (expectations and results)

Could haves

	Expectations	Results
1	X-ray vision	**Achieved**
2	Tracking of path & time walked.	**Achieved**
3	Simple HoloLens to Hub communication (HoHu)	**Achieved**
4	Add advanced semantics automatically (object recognition)	**Achieved (door detection)**
5	Holo to Holo communication, e.g. shouts: “Watch out!”, “Over here!”	**Not achieved**
6	A HUD to show extra information to personnel (current floor, distance to exit, minimap)	**Achieved (minimap)**
7	Embedding georeferenced dataset with GIS data. (showing context, adding indoor pc to the actual building within a city model)	**Not achieved**
8	Make this dataset usable within a “*veiligheidsapp*” type app on mobile devices.	**Not achieved**
9	Fire detection	**Not achieved**
10	Connect different sensors (IoT)	**Not achieved**
11	Face / person detection to aid in finding casualties	**Not achieved**

Table 1.c Could haves of project (expectations and results)

Won’t haves

	Expectations	Results
1	Product testing with real response teams (surveys)	**-**
2	Indoor wayfinding	**-**
3	Data acquisition	**-**
4	Researching the capabilities of using *SLAM* devices in harsh conditions (smoke, heat, wind, cold)	**-**
5	Audio communication	**-**

Table 1.d Won’t haves of project (expectations and results)

5.2 User requirements versus achievements

During the whole project realisation, one of the main focuses was to create a product that would answer user needs and really help firefighters during fire emergency responses. In this section, we assess how our product achieves this goal.

To make sure, our project would answer user needs, an interview about user requirements was made at the beginning of the project (see appendix A). The main user requirements that were highlighted during the interview and the way we answered them are listed below (see figure 15):

- Real-time tracking of the firefighters teams and fire location and expansion

The implemented application allows for tracking of HoloLens users in real-time. The track of the users inside the fire building is displayed in the surrounding environment on the UI. However, fire detection was not directly implemented but the functionality of the application that allows users to add pings makes it possible for the HoloLens users to depict the fire location on the displayed 3D map.

- Location awareness of building features (building geometry, fire entrances and stairways/elevators, adjacent buildings or wings for the building)

Not all required building features are identified and displayed in our application but some semantic classification is performed for detecting basic features like walls, ceiling, ground and furniture. Door detection is also implemented so that the doors traversed by the HoloLens users are displayed on the 3D map.

- A combination of 2D and 3D representations that could result from the implementation of ToggleMaps

ToggleMaps concept was not directly implemented. However, our product produces a dynamic 3D map of the indoor building. In addition, a 2D representation (minimap) is also displayed to the users.

- Filtered information as the information displayed should be meaningful but too much information might confuse the users

To avoid user confusion, the developed product displays essential and basic information for supporting firefighters in their tasks. Moreover, a system of layers was implemented in the navigator so that the application user can select and filter the information he wants to display on the UI. In this way, he can visualise only the information he needs. For instance, the user can choose to display only a certain type of semantics (only the walls,...).

- Devices that would let the firefighters their hand free

The device chosen for our information system is the Microsoft HoloLens. This head-mounted device allows users to have an augmented vision of their environment having at the same time their hands free so that they can perform their tasks without restriction of movements (Arif, 2019).

.

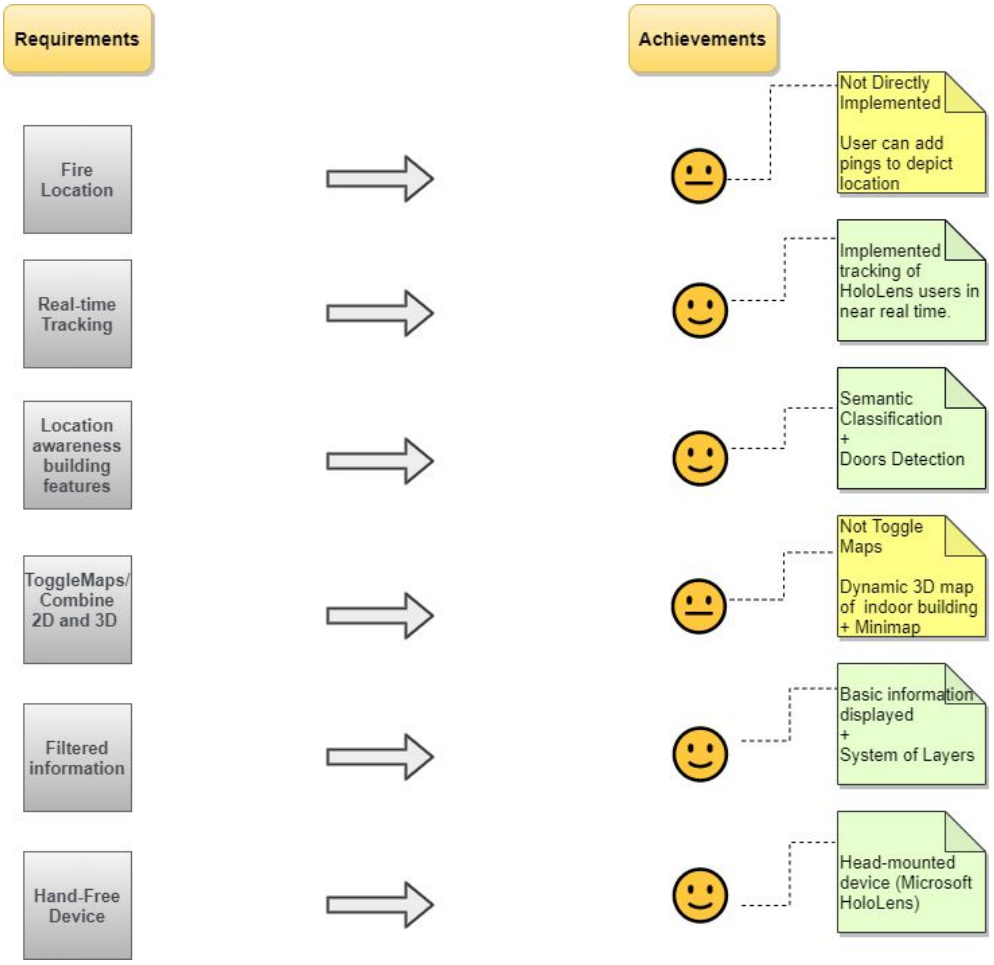


Figure 15: User Requirements and Product Achievements. Source: Author

5.3 Navigator Results

The Navigator codebase we ended up with is quite sizable for the scale of a synthesis project (10.000 lines of code) and covers several topics. In this section, the results of the most notable features of the navigator are discussed.

5.3.1 Semantic Enrichment of the meshes

In this chapter, we present some results regarding the output of our classification algorithm, taking as an example meshes that were generated in a *CGI* office. Capabilities and

limitations of the method are detailed. It is worth to say that this method works in near real-time but the results shown are based on later processing of saved meshes using the emulator.

Capabilities

The aim of classifying the meshes into ground, walls, furniture and ceiling surfaces is to help firefighter response teams to make a better sense of the indoor space. Figure 16 shows two different views of the raw meshes generated by the HoloLens user. The scene represents an office at *CGI* company. From that figure, it is obvious that it is hard for the response teams to have a clear understanding of the room features. It is hard to distinguish different furniture from the ground or from the walls. All the meshes have the same color regardless of their semantics and the interpretation of the scene is quite tough.

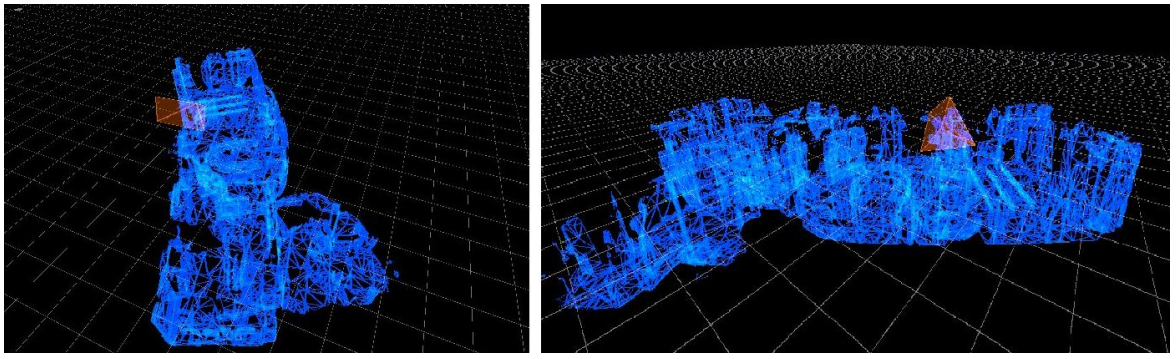


Figure 16: Two different views of unclassified meshes generated in an office at CGI company. Source: Author

This is where the semantic enrichment procedure takes place. By using the methodology described in chapter 4, each mesh is classified and displayed with different color based on the class it belongs to, as shown in figure 17. The red color corresponds to the wall meshes, the light blue color corresponds to the furniture meshes and the gold color corresponds to the ground meshes. Ceiling has a green color but in our scene the ceiling of the room was not scanned. The results obtained are quite promising. The structure of the indoor space is clearer and the distinction between furniture, wall and ground is easily made. The chairs, the tables and other furniture can be detected easily. Moreover, an assumption regarding the width of corridors can be made based on the ground and wall meshes.The response teams can now efficiently interpret the scene and they can use this knowledge to develop a proper deployment strategy without relying on risky assumptions.

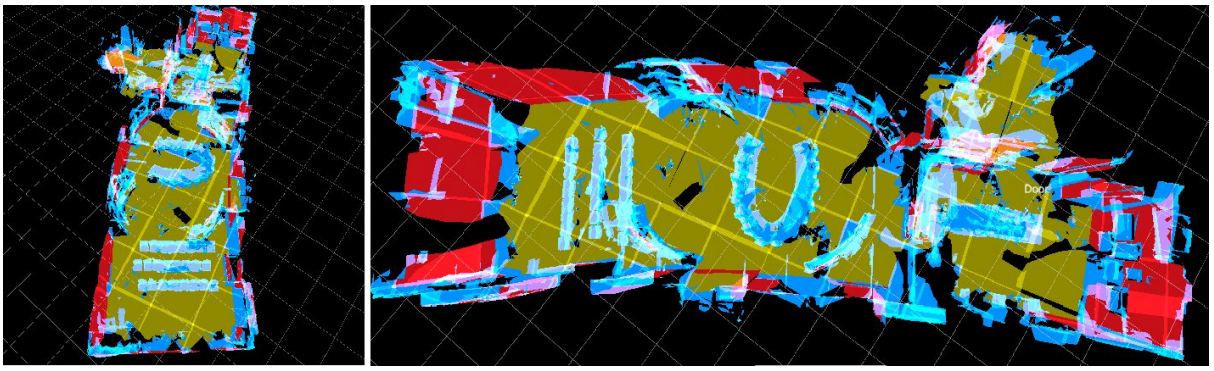


Figure 17: The meshes generated in the office at CGI company after adding basic semantics. Source: Author

In addition, it is also possible for the user to display only a certain type of meshes. For instance, in figure 18, only the walls and the ground are displayed.

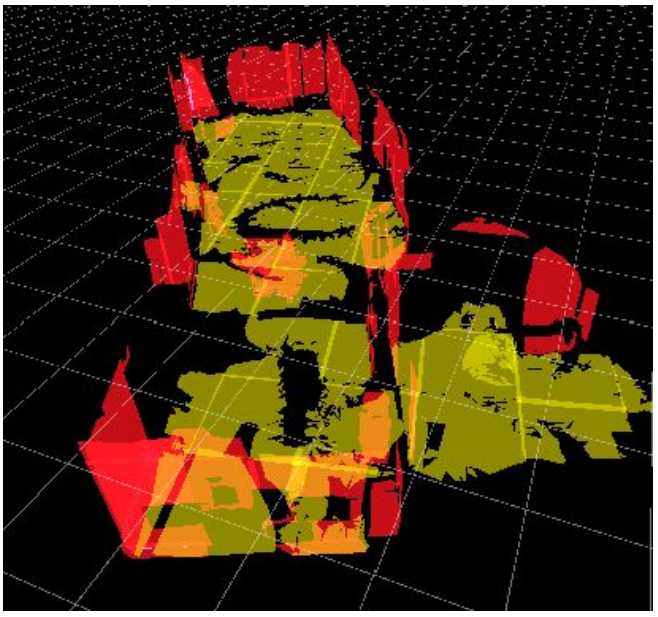


Figure 18: CGI room with only walls and floor meshes displayed. Source: Author

Besides the meshes classification, when the classifier script runs, doors are also detected. As stated by the commanding officer in the interview we conducted, the knowledge of doors/entrances locations can lead to a safer deployment strategy. Figure 19 shows a door that is detected in the *CGI* office. The door in the model is displayed as a label “door” with white color just above the ground. In that figure, furniture meshes are not displayed.

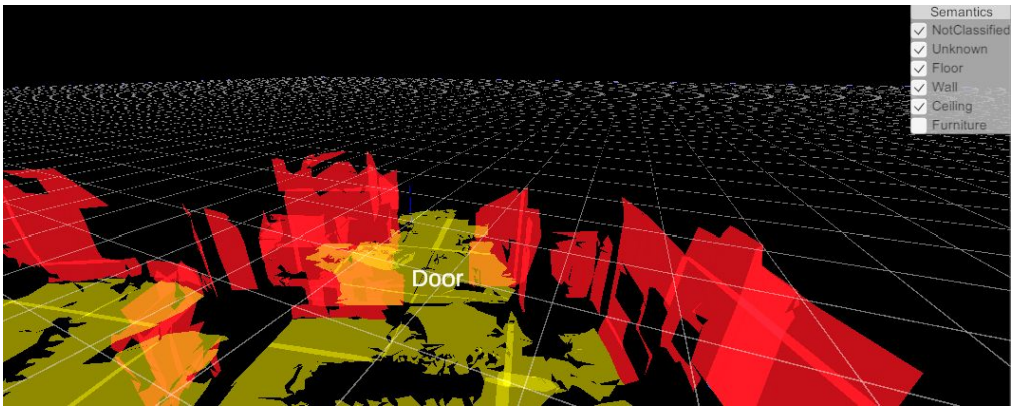


Figure 19: Door detected in the CGI office. Source: Author

Limitations

Although the results of the semantic enrichment of meshes are quite promising, there are still some limitations. Time, accuracy and functionality in different environments are essential in our use case, thus optimisation regarding those will increase the value of our product.

A) Time needed for meshes classification in big/complex environments

During the quality assessment made at TU Delft library (see chapter 6), at some points we realised that the classifier algorithm needed a bit more time to run than when we tested it in other less complex/big environments. From this, we can state that semantic enrichment works in semi real-time.

B) Thresholds not updated with saved data

When the meshes are generated in real-time by a HoloLens user, the thresholds are redefined every x newly added meshes. The recomputation of thresholds every x meshes is essential especially for buildings with different structures per room and with different floors (to avoid the ground and ceiling thresholds being defined based on the first floor heights only). When the classifier does not run in real-time but through the emulator to classify saved meshes, the recomputation of those thresholds is not taking place since all the meshes are passed together as input and not in acquisition order. In the specific cases described, meshes can end up misclassified. An example is shown in figure 20 where the ground of the second floor is misclassified as furniture.

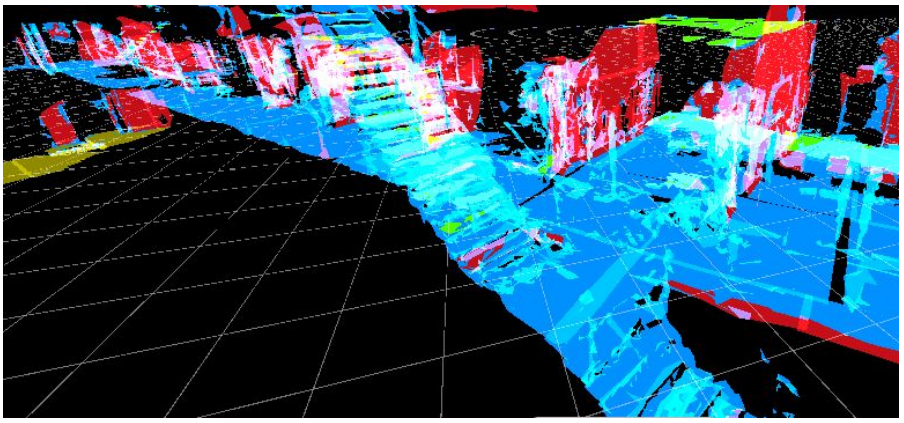


Figure 20: TU Delft library room, ground misclassified as furniture due to missing necessary updates of thresholds. Source: Author

C) Door detection limitations

First of all, we were not able to test the door detection algorithm many times with different indoor space structures.Thus, we cannot claim that our approach will work in every situation, especially because this method is really sensitive to noise and outliers in the input classified meshes.

In the *CGI* office, there are actually two doors but only one is detected when performing door detection with the saved meshes of that room (see figure 21). In order for a door to be detected, two things are necessary. First, the user needs to pass through the door and not only to scan it and second, the two walls next to the door need to be scanned properly. In figure 21, the second requirement is not fulfilled since the walls next to the second door are not properly scanned. The yellow arrow on the figure shows the place where the second door is located while the pink arrow shows the wall part that is not scanned by the HoloLens.

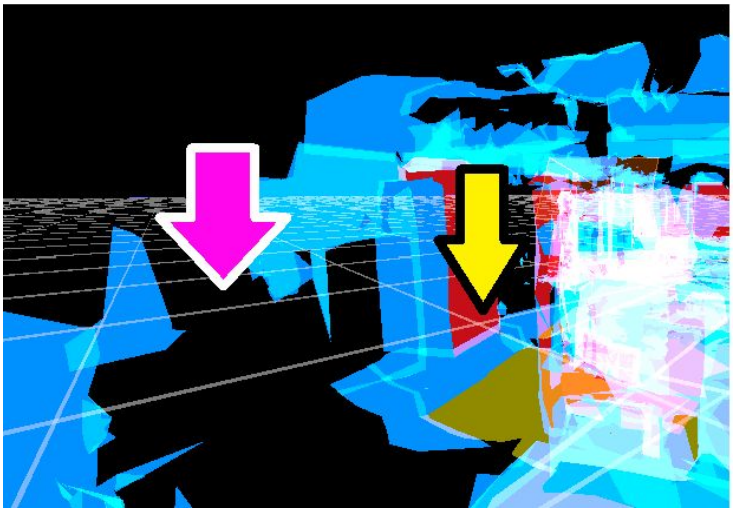


Figure 21: Location of missing door (Yellow arrow), not scanned area (Pink arrow). Source: Author

Other approaches for meshes classification

Two other approaches were tested for classifying the meshes. The first approach labels each triangle of the input mesh one by one (instead of labelling collections of triangles).

Figure 22 shows the result of that algorithm from two different views. The classification with this algorithm is not that accurate. The main problem is that most of the vertical parts of furniture surfaces are misclassified as walls. Moreover, small furniture close to the ground (chairs in the left part of figure 22) are also misclassified as ground. It is worth mentioning that this was the least time consuming approach. Misunderstandings regarding the structure of the indoor space can be dangerous in our use case and that is the reason why this approach is not our main approach.

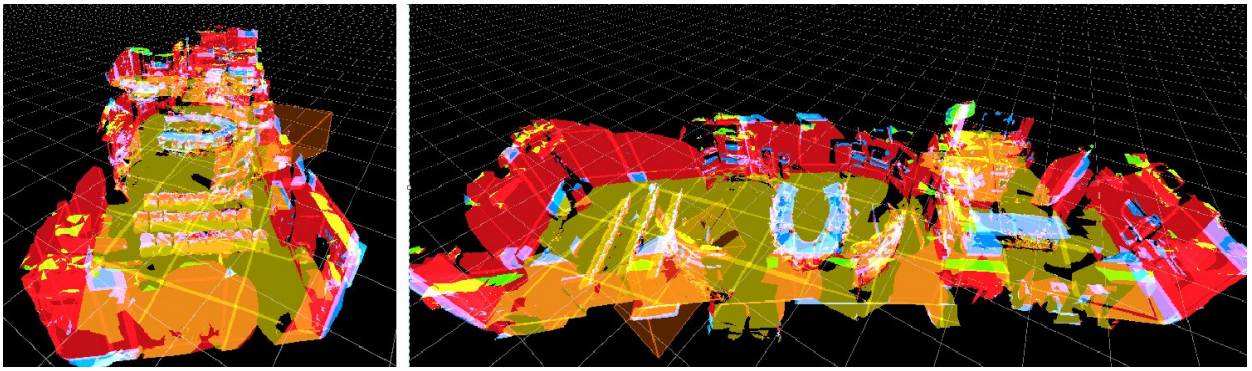


Figure 22: Meshes classified with "LabelPerTriangle" Algorithm. Source: Author

The second approach uses the RANSAC method to detect planar surfaces. Figure 23 shows the result of the RANSAC algorithm from two different views. With this approach, we get more noisy results compared to our main approach (region growing algorithm). Still the response teams might be able to make sense of the indoor space but there is no advantage of this approach over our main approach as they take the same time to run.

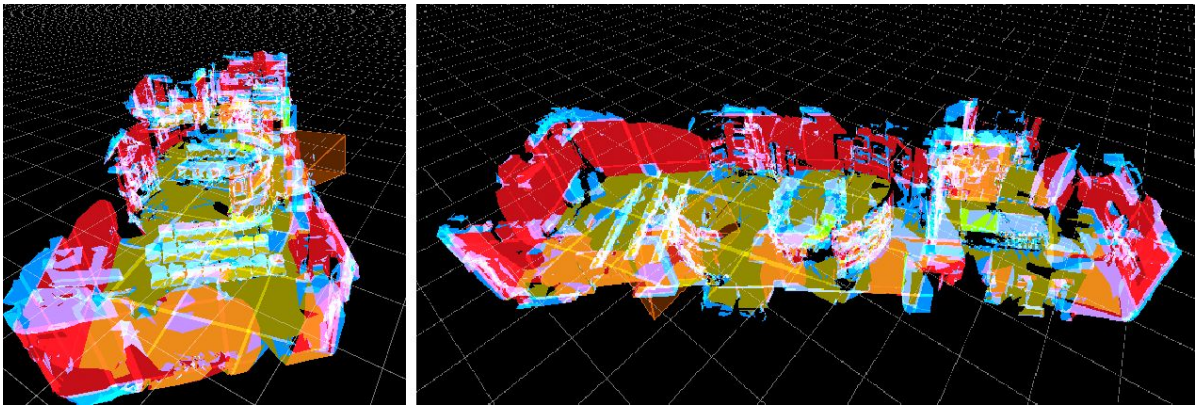


Figure 23: Meshes classified with RANSAC Algorithm. Source: Author

5.3.2 UI

In order to make the navigator application interactive, the back end needs to be connected to the front end. The front end is represented by the User Interface (UI). The implementation of the UI is built in a modular way, meaning that UI components can easily be added to the unity scene while coding. The UI consists of a menu bar framework, minimap, log system, emulator options and toggle panels.

Menu bar framework

At the top of the screen, a menu bar is placed to easily access the various functionalities of the application (see figure 24). When clicking on a button, a dropdown menu with other buttons appears. In this section, for each of the buttons the functionality of its child buttons is explained.

Figure 24: menu bar of the UI. Source: Author

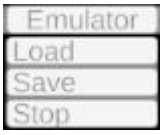
App button:

- Reset: this resets the currently active emulation or HUB connection to a blank grid again.
- Quit: if the application is built as an executable, then the application will be closed.

HUB button:

- Start: starts a connection with an active HoloLens in the Explorer application.
- Stop: ends a connection with the active HoloLens.
- Save: saves the mesh generated in the current HUB session as a binary file, which can be loaded by the emulator again.
- Clear: clears the current session and removes the generated mesh.

Emulator button:

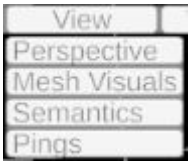


- Load: opens a file panel in which the user can choose a binary file to load into the application for visualisation.
- Save: saves the emulation.
- Stop: stops the emulation and removes the emulated mesh from the screen.

Edit button:

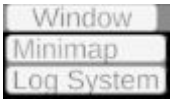
- Resolution: spawns a window that allows the user to choose a desired screen resolution. This only works if the application is built as an executable.

View button:



- Perspective: toggles a list of the perspective options in another panel.
- Mesh Visuals: toggles a list of the mesh visual options in another panel.
- Semantics: toggles a list of the semantics options in another panel.
- Pings: toggles a list of the available pings in another panel.

Window button:



- Minimap: toggles the minimap component. More information on the minimap below.

- Log system: toggles the log system component. More information on the log system below.

Minimap

The minimap shows the 3D model from a top down view oriented in one direction (see figure 25). It follows the cursor that moves the scene around, but it can also follow a HoloLens in the field.

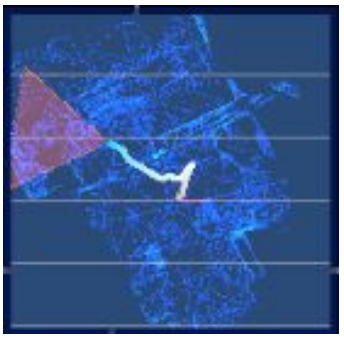


Figure 25: minimap. Source: Author

Log system

The log system is used to display messages and information about the 3D model and about the actions the user makes. It can also be used to transfer messages between the Navigator application and a HoloLens in the Explorer application, but that is not yet functional (see figure 26).

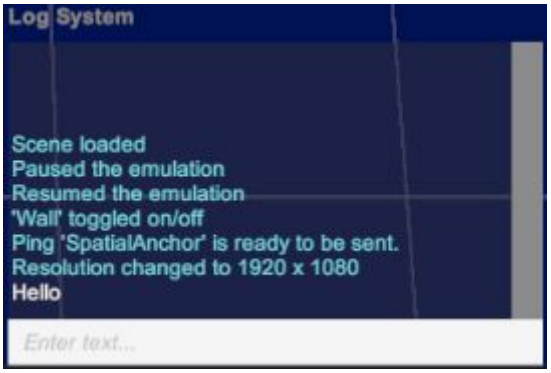


Figure 26: log system of the UI. Source: Author

Emulator

When the emulator is running, a panel with a play/pause button and a time slider is displayed to manipulate the emulation.

Toggle panels

These are the panels that can be activated by the buttons underneath the view button (see figure 27). The perspective options change the point of view of the camera, but this is not implemented. The mesh visuals options change the way the mesh model is displayed on the

screen. The semantics checkboxes toggle the different classified meshes (floors, walls, ceilings, furniture) of the mesh model on or off. The ping options let the user choose which ping to activate, which can be placed by pressing the mouse wheel at a specific location in the mesh model.

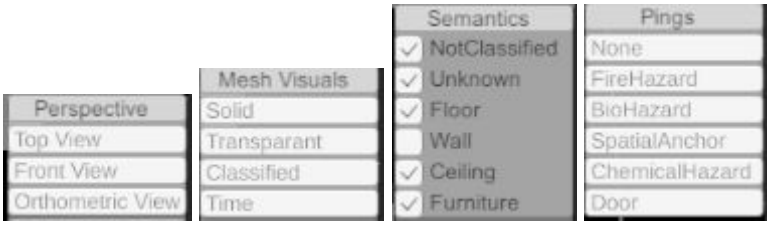


Figure 27: Toggle panels. Source: Author

Other functionalities

All the panels spawned on the screen are customisable by the users. This means that the visibility of most panels can be toggled on or off. Also the panels are draggable allowing the user to place them at a desired location on the screen.

5.4 Explorer

5.4.1 Shared coordinate system

By using the *ASA* system, we were able to establish a shared coordinate system. This makes it possible to track multiple users in the navigator at once, as well as display their position on the HoloLens. The spatial anchor system is not perfect, though. The main flaw is that we only use a single *ASA* to establish the shared coordinate system. This causes small deviations in how two HoloLenses position the *ASA* and leads to larger deviations the further they move away from it. Another problem is that the explorer sometimes fails to detect *ASAs*. This problem is usually resolved by rebooting the HoloLens. We suspect this issue arises from multithreaded processes failing to halt when exiting the application. Yet another problem is that it is quite easy to accidentally create multiple spatial anchors through the in-game menu. Due to the coarse relocalisation method used for finding nearby *ASAs* this means sometimes two HoloLenses will detect different spatial anchors. This causes them to have different coordinate systems, causing an offset in their positioning and orientation, as well as the mapped environment.

5.4.2 Optimisation

The optimisation techniques we implemented had the desired effect of increasing the frame rate. The original application ran at around 30 FPS with regular dips to around 5 FPS. Despite our implementation performing significantly more calculations it usually runs at around 60 FPS, although notable dips to around 40 FPS happen when large amounts of sprites are being rendered at once. This is most likely a limitation of the HoloLens' GPU, though, and not due to our implementation.

5.4.3 User interface

The user interface mostly functions as it should, but there are a few minor issues that reduce its usability. The first is that it is possible to accidentally lock the pop-up menu into place without noticing it by dragging it with a hand gesture, and no way to reverse this without restarting the application. Another issue is related to the minimap. The position of an object on the minimap edge is computed by interpolating between its last known position and its current position. If the angle between the object and the user changes from 0 to 359 degrees, the object will move 359 degrees around the border of the minimap counter clockwise, while ideally it would only move one degree clockwise. This problem may cause disorientation for the user.

6

Quality assessment / Testing

This chapter discusses a testing scenario around using the Navigator application and two HoloLenses and its results. The chapter also presents a quality assessment of different categories for the HoloLens.

6.1 Testing scenario

In order to test our implemented product and the HoloLens in a real life situation, VRR developed a test scenario as part of this project. The scenario takes place in the library of the TU Delft on Monday 15-06-2020. As the current accessibility of the library building is limited due to the current lockdown related to the Coronavirus crisis, we needed to adapt the initial testing scenarios. The initial scenarios with maps are included in appendix B. The scenario we developed coping with the environmental limitations is depicted below.

In our adapted scenario there are “chemical spills” reported somewhere in the library building. These spills need to be found and removed. Two emergency response teams, both equipped with HoloLenses are deployed for this operation. The *incident commander*, who is connected to the two HoloLenses of the teams through the Navigator application, is situated in the VR Zone, which is also the starting point of the two teams.

When the operation starts, the first team moves as fast as possible through the library in order to find the location of the spills. When the spills are found, this team places chemical hazard pings at these locations which can be viewed by the commander through the Navigator application. As soon as the first location is found, the second team holding heavy equipment is deployed. This team needs to navigate to the first location by using the pings of the chemical spills displayed on the HUD of the HoloLens. When the spills are found, the team starts removing them.

The paths of the two teams are shown in figures 28 and 29. The green stars depict the locations of chemical hazard pings.

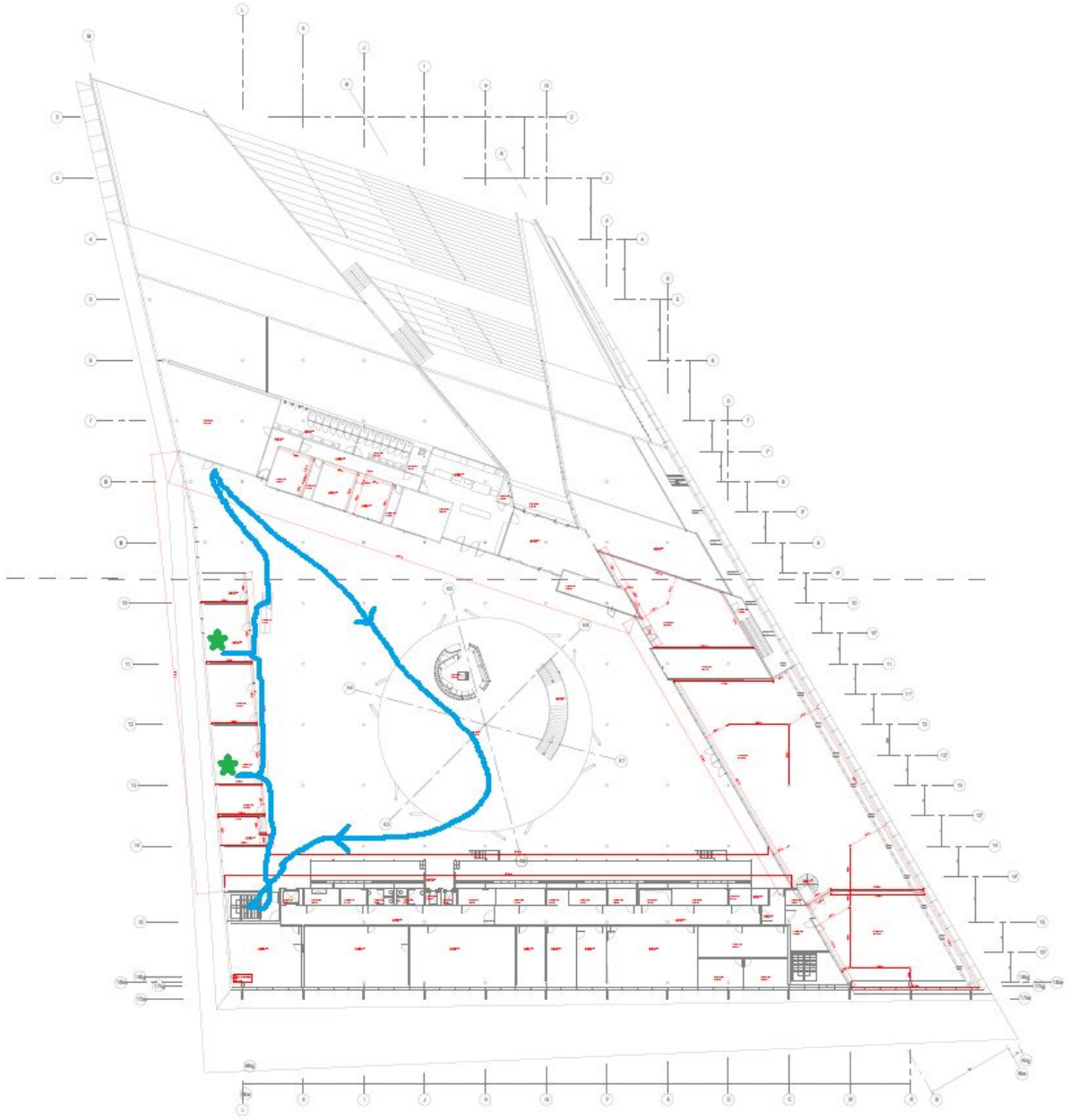


Figure 28: Path of team 1. Source: adapted from clients' scenario

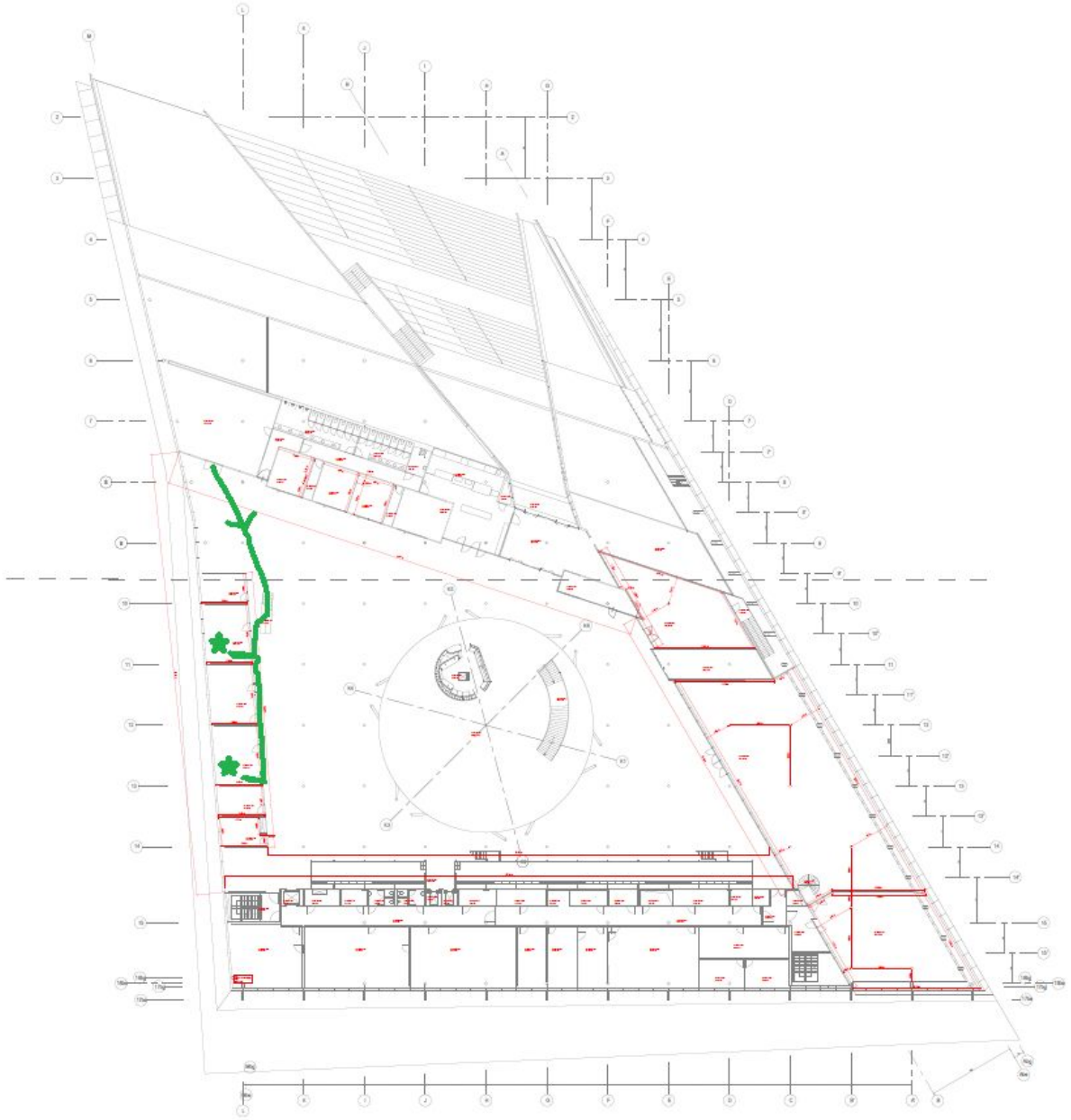


Figure 29: Path of team 2. Source: adapted from clients' scenario

6.2 Test results

The testing scenario was executed as described above and shown in figures 28 and 29. The scanned mesh displayed in the Navigator application for the commander during the testing scenario is recorded (https://www.youtube.com/watch?v=D3ISTzrMcaw).

During execution of the testing scenario we ran into some issues:

- The placement of the initial spatial anchor, meant to combine the coordinate reference systems of the two HoloLenses into one space, was sometimes troublesome. This is probably because the spatial anchor needs to be placed at a position with much variation in the environment or because of a weak WiFi signal. This meant that the eventual scenario was only executed with one HoloLens, used sequentially by the two teams. Before the execution of the testing scenario, we managed to create a mesh of the library with two HoloLenses in one space.
- We wanted to record the video of the HoloLens, but due to the large amount of bandwidth needed for this recording, the framerate of the video feed dropped significantly and could easily cause the HoloLens to crash. Therefore we decided to skip the recording of the HoloLens video feed.
- The placement of the chemical hazard drifted away from the location it was initially placed on.
- The classification and visualisation of the semantics have not been tested in real-time since there were some limitations (as explained in the previous chapter) but it was tested later with the saved meshes generated during our testing implementation. Figure 30 shows the results of classification of two different scenes that were acquired in TU Delft library.

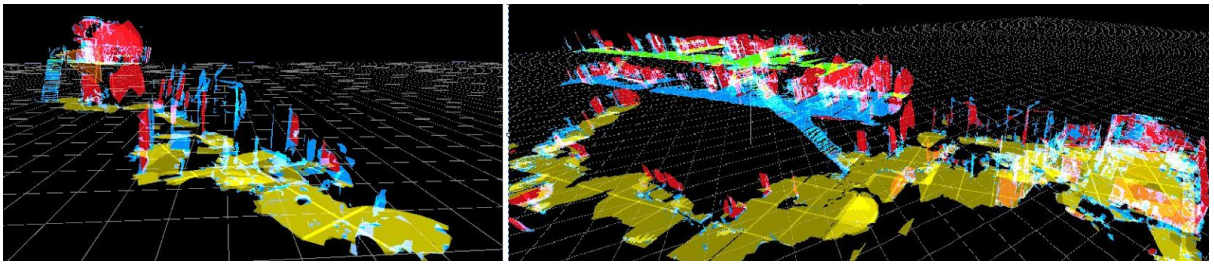


Figure 30: Classified meshes generated in TU Delft library during testing scenario. Source: Author

Besides the issues, we could say that the eventual scenario was quite successful. We were able to display the mesh generated by team 1 and save it as a binary file. Team 2 was able to rapidly be dispatched to the chemical hazard pings placed in the scene by team 1 by making use of the HUD in the HoloLens for navigation.

We learned from this that the HoloLens is not always reliable when placing the initial spatial anchor to be shared among two HoloLenses. We also experienced the limiting factors of the HoloLens: rapidly depleting batteries and long deployment time after adjusting something in the code.

6.3 General Quality Assessment

To assess the quality and limitations of the HoloLens in general different test categories are determined. These are user motion, lighting, environment, spatial awareness and user comfort. This categories can be described as follows:

- User motion: Does the HoloLens still work properly when the user starts accelerating or running or does it require a steady state? What happens when the user makes fast moves (suddenly turns back) or shakes the HoloLens?
- Lighting: Does the HoloLens still work properly when the lighting changes, for instance in darker rooms or in rooms with reflectives surfaces as windows or mirrors? Does smoke influence the HoloLens in any way?
- Environment: Does the HoloLens work properly in any kind of room (small VS large)? How does it react to stairs? What about unusual surfaces as curved geometries? What happens when people/animals pass in the view field of the HoloLens?
- Spatial awareness: How does the HoloLens "react" when it is lost? What happens when the WIFI signal is reduced?
- User comfort: How does the user feel after having used the HoloLens for a long period of time? Also, is the user limited in his movements or embarrassed in its activities?

The test results for those different testing categories are shown in table 2.

Tests	**Test results**
User motion	
1. Make sudden moves : quickly turn back or move head 2. Walk ⇒ run 3. Shake the HoloLens	1. The Explorer application might crash completely. During testing, Max stumbled, causing the HoloLens to crash. This only seems to occur sometimes and is hard to reproduce. 2. Moving too quickly sometimes causes the HoloLens to fail to map the environment, sometimes causing loss of tracking. 3. Same as 1.
Lighting	
1. Test the HoloLens in front of windows/mirrors 2. Test the HoloLens in dark VS light rooms 3. Does smoke affect the HoloLens?	1. It sees the reflection of the mirror as extra space, as if you could step into the mirror. Transparent surfaces are not mapped. 2. Dark rooms cause tracking loss, the more light the better it works. 3. We were not able to test ourselves. Other people have performed tests of this. From this test we can conclude that dense fog or smoke is picked up by the depth sensor as geometry, making it fail

	to map the environment.
Environment	
1. Test the HoloLens in small room 2. Test the HoloLens in large room 3. Walk stairs with the HoloLens 4. Test the HoloLenses in room with curved geometries 5. Test the HoloLens with moving object (people/animals/opening a door) passing in front of the user 6. Test the HoloLens outside	1. The HoloLens can map the entire room. Extremely tight spaces might cause the HoloLens to lose tracking. 2. It only maps a local part of the room around the HoloLens user. 3. Moving up stairs causes the holograms to shake but pose no further problems. 4. Curved geometries pose no problem, although they may be scanned less accurately due to the low resolution of the HoloLens depth sensor. 5. Changing the scene too much might cause *ASAs* to not be registered. If an object restricts the HoloLens' field of view it might also lose tracking. 6. The HoloLens works well outside, given that an Internet signal is available.
Spatial Awareness	
1. Test the HoloLens in a room with reduced WiFi signal 2. Do placed holograms stay in place during usage?	1. When the WiFi signal is lost, the application does not work anymore. In order to send the information captured by the HoloLens via the Azure Cloud to the Navigator application, a stable Internet connection is essential. 2. Information placed in the scene may drift slightly when the user moves their head. This depends on the current frame rate and how well the environment has been mapped. Loss and regaining of tracking sometimes causes holograms to be moved to a slightly different position. The difference is usually not enough to significantly reduce usability.
User comfort	
1. Test the HoloLens for a long period (more than 20 min at least) to see how the user feels afterwards. 2. Is user movement limited?	1. Some people become nauseated after a long period of use. This is mostly caused by the HUD. Long periods of use might also cause eye strain, also due to the HUD. Headaches might also occur. All of these are helped by calibrating the HoloLens for each user. 2. User movement is not limited.

Table 2 : results of the quality assessment

7

Conclusions and recommendations

This chapter is dedicated to the conclusions and recommendations of this Synthesis Project. In the following sections, first the research questions will be answered. The conclusions and future work will then be addressed.

7.1 Research questions

The main research question this project aimed to answer was:

"*In what way can the use of indoor point clouds assist firefighter response teams in real-time tactical decision-making?*"

This main research question was subdivided and answered through the following sub research questions:

- *"What **geoinformation** do the response teams **need**?"*

To answer this research question and to make sure our project would answer the user's needs, an interview was conducted with the intended users to know what information they needed from the developed *information system* (see appendix A). During this interview, the users stated that the information needed by the response teams for safe deployment and decision-making consisted mainly of the following information:

- Real-time tracking of the firefighters team and fire location and expansion;
- Location awareness of building features;
- A combination of 2D and 3D representations that could result from the implementation of ToggleMaps (van der Meer et al, 2018); and
- Filtered information as the information displayed should be meaningful but too much information might confuse the users.

The building features are the building characteristics needed to perform a safe operation. According to the conducted interview, the most important building features that the response teams need to know to perform a safe deployment consist mainly of the geometry of the building, the fire entrances, the location of stairways and elevators and information about the adjacent buildings or wings of the building.

- *“In what **form** should we provide this **information** to firefighters so it can be effectively used?”*

The information needed by the firefighters during the emergency responses is provided to them into different forms. First of all, the HoloLens users environment is displayed to the Navigator application through a combination of 3D classified meshes with a minimap supporting the user navigation into that environment. Real-time positions and orientations of the HoloLens users are also provided to the Navigator application users and displayed over the 3D environment. Besides, the UI was designed with the possibility for the application user to filter the information he needs as he may tick on and off layers of information. Finally, information about that environment is also given through pings that are virtually placed in the real world. Those pings can be added both by the Navigator application users and by the HoloLens users and they are thus visible for these two kinds of users as well. In that way, pings can be used to inform firefighters on the field and people in the control room about fire location and expansion for instance.

- “*How can **information** be **communicated** within and between tactical response teams?*”

The communication between the firefighters when they are inside the building is crucial. In the *information system* we developed, a log system was implemented to allow sending text messages from the control room to the HoloLens users. However, this tool is not functional yet. Nevertheless, response teams can still communicate with each other by making use of the pings functionality of our product. In that way, HoloLens users can use preset pings (chemical hazard, biohazard, fire hazard,...) or create their own pings and display them within the scene. Those 2D icons will be visible for all HoloLens users on their HUD as a hologram but also within their minimap. These pings are also visible for the commander in the control room within the 3D interactive environment in the Navigator application. The same way, the Navigator application user can place pings within the 3D model and those pings will be visible for HoloLens users as well. As a result, they can inform each other about the condition, location and progress of the fire and exchange other important information by making use of the pings functionality.

- “*How can the raw **data** acquired by response teams be **enriched** so as to maximise its value for tactical decision making?*”

Before the response teams determine their strategy, they need information about the indoor space in order to make sense of that indoor environment. As shown in our results, we enrich the raw data acquired by the HoloLens users with some basic semantics. The classification of the acquired meshes into walls, ground, ceiling and furniture meshes and display of those meshes into different colors help the commander of the response team to understand the basic structure of the indoor space. For implementing this, we developed a classifier algorithm based on region growing. This algorithm classifies the different indoor surfaces based on the surfaces properties (height, area and orientation).

In addition, the doors traversed by the HoloLens users are detected. This more complex semantic will also add to the decision-making procedure. Indeed, as also stated in the interview that we implemented, doors position is crucial when they develop a deployment strategy. Doors are detected by implementing a door detection algorithm. This works on computing intersections between detected walls and the HoloLens users' track as those intersections actually correspond to door positions.

- "*How can we process raw data from **multiple sources** in **real-time**?*"

This question has two components, and should be answered by addressing these.

How can we process data from **multiple sources**? We linked them together using a singular spatial anchor common to all sources and wrote a communication system that, among other things, can recognise the sender, and can 'weave' data streams from multiple sources.

How can we do this in **Real time**? By setting up independent threads with asynchronous processes we can process the data "behind the scenes". The UI will remain running with no stutters. To make processed data appear on the screen, we had to communicate across different threads using queues.

- *"How can we acquire real-time tracking of the firefighter team(s) and **share** that **information** with the control room?"*

Real-time tracking was implemented in our product by making use of Microsoft HoloLenses. Those *SLAM* devices position themselves in the environment they are mapping, relative to where their mapping session started. This means we needed to implement a shared coordinate system for allowing the tracking of multiple HoloLens users. To do so, the *Azure Spatial Anchors* functionality of the HoloLenses is used. In that way, one HoloLens user can create a spatial anchor within his environment and the other users use that spatial anchor to position themselves in the shared coordinate reference system. When a HoloLens enters an area, it will look for that spatial anchor by using the *wifi fingerprinting* functionality of the HoloLens and by matching its environment with the spatial anchor. Once the spatial anchor has been found, the HoloLens can derive its position and orientation within the shared coordinate system. That whole procedure runs in real-time.

As for sharing that information with the control room, this is done by making use of the Azure Cloud. Once the HoloLens finds a spatial anchor, the Explorer starts sending the scanned environment in the form of meshes and the user's position and orientation to the cloud. The Navigator app can then retrieve this data from the cloud and display the 3D environment with the HoloLens users track to the app user. In that way, the control room and fire officers that are not staying inside the building during the intervention can visualise the HoloLens users track onto a 3D dynamic map showing the indoor space of the fire building.

7.2 Conclusions

The aim of this project was to answer the main following research question :

"*In what way can the use of indoor point clouds assist firefighter response teams in real-time tactical decision-making?*"

To answer this research question, we developed a 3D *information system* making use of *mixed reality (MR)* and indoor point clouds to support real-time tactical decision-making during fire emergency responses. More specifically, we implemented a 3D interactive environment displayed as a point cloud or mesh with additional information as the position and orientation of the firefighters within that environment. To develop this *information system*, some literature research was needed. Besides, we made a user requirements interview in order to take into account the specific needs of the user and make our *information system* display the needed information. A quality assessment was also performed to assess the capabilities and limitations of our product.

As a result, the output *information system* makes use of Microsoft HoloLens devices and consists of two related applications: the Explorer and the Navigator. The Explorer is the application used to run on the HoloLens devices. It creates a 3D mapping of the HoloLens user environment and provides those users an augmented vision of that environment. Through the use of spatial anchors, the Explorer makes it also possible to synchronise multiple HoloLens users and allows for users tracking. The environment and user tracking data are then stored through an Azure Cloud. These data can then be retrieved by the Navigator application. The Navigator is used for the visualisation and interaction with the retrieved data. It also allows some data processing which makes it possible to classify the 3D environment into different categories: walls, ground, ceiling and furniture. The classified environment of the HoloLens user is then displayed in 3D in the Navigator application along with the tracking information of the HoloLens users (thus, the explorers). In that way, people in the control room (thus, the navigators) can visualise the firefighters' environment and their positions. The implemented user interface enables the user to select the information he wants to display within the Navigator application. Besides, this 3D environment information is accompanied with a 2D representation as the user interface also allows for displaying a minimap of this environment. Finally, to enable communication between and within the response teams, the developed *information system* also has a ping functionality. The HoloLenses users can add pings (2D icons) in their environment and those icons will be visible for other users as well but also on the Navigator application in the control room. Reversely, the Navigator app user can add pings onto the 3D environment and those will be visible by HoloLens users.

The *information system* implemented thus has a lot of capabilities to assist firefighters in their work. However, it still has some limitations. Not all implemented functionalities are actually functional. For instance, a log system was developed to allow for sending messages from the control room to the HoloLens users, but it is not functional. The data processing for

adding semantics to the 3D environment causes lags in the system when scanning complex or large rooms. The system only makes use of one spatial anchor to create the shared coordinate system for multiple HoloLenses, which causes small to large deviations as the HoloLens go further away from the spatial anchor. Our product thus has some limitations and future improvements could be made to solve for that, as it will be addressed in the following section.

7.3 Future work and recommendations

In the following sections, future work and other recommendations will be addressed from three different perspectives; general future work, future work for the Navigator application and future work for the Explorer application.

7.3.1 General

This project opened a window for future research. In the first place, new researches could focus on testing our implemented product on the field with firefighters in order to make real tests. This could also help to do some usability assessment of the product with the intended users and know whether the product is easily usable for unskilled users or not.
Furthermore, as it was highlighted in the related work section, previous researches aiming at developing 3D *information systems* for fire emergency responses mainly focussed on implementing helmets equipped with thermal imaging cameras for providing firefighters with an augmented thermal vision. There is a comparative study to be done between those thermal data-based products and our product also focusing on providing an augmented vision to support firefighters in fire emergency responses.

7.3.2 Navigator

Indoor navigation
This was not in the scope of our project as it was part of our “won’t have” in the *MoSCoW prioritisation* but it would be interesting to add some indoor wayfinding to the implemented product. In that way, the data acquired by the HoloLenses could actually be used and processed in order to derive an indoor navigation graph. This functionality could even directly be implemented by making use of the scene classification and door detection functionalities of the product.

Semantics enrichment of meshes
Regarding the semantics enrichment of the meshes we achieved our initial goal, that is to add basic semantics and offer the possibility to the user to make sense of an indoor scene. A useful functionality that could be added is the detection of even more complex objects such as staircases and elevators. Deep learning algorithms could then be used to achieve such goals. This would help users to have a more detailed view of the indoor space and exploit even more features of the model. Moreover, as explained in a previous chapter the detection of doors was not tested a lot and stays thus experimental. The implementation of more tests and the optimisations of parts that might not work in all cases is needed in the future.

UI
As part of the user interface, a log system was implemented to display messages and information about the 3D model and about the actions the user makes. For now it is not yet functional, but it could be adapted to allow transferring messages between the Navigator application and a HoloLens in the Explorer application.

7.3.3 Explorer

In this section we discuss the future works that could be done to improve the Explorer application. Some of these future works are currently possible but were just not implemented due to time constraints, while others depend on additional hardware which was not available in this project.

Spatial anchors
Currently the application only uses a single *Azure Spatial Anchor.* In the future we would change it so that spatial anchors are dropped automatically as the user walks around. This prevents the drift in accuracy at distance from a spatial anchor. A possible implementation could consist of creating a graph of *ASA*s. If the explorer measures new positional or environment data, it looks for the nearest *ASA* in the graph. It then traverses the graph until a designated main anchor is found, applying a coordinate transformation on every edge.

Camera feed
It is possible to get direct access to the HoloLens camera feed. This could be sent directly to the Navigator where it can be viewed by the command centre. Getting access to the camera feed also opens up more options to improve environmental understanding. For example, computer vision methods could be used to detect faces to assist in the search for casualties (Viola and Jones, 2004). The camera feed could also be projected onto the scanned environment as a texture, thereby increasing its readability.

Audio
The HoloLens has the ability to use audio commands. In the future, we could use these to perform various functions we currently use hand gestures for. Audio could also be transferred to the Navigator, which could replace the current use of radio communication.

Sensors
Connecting the application to sensors outside of what the HoloLens has could massively increase its functionality. To start with, adding a GPS sensor would allow the explorer to integrate its data with the current VRR Veiligheids App. This would allow it to access data that was created beforehand, not just during the operating session.
Another sensor that would be very useful for firefighters is a thermal camera. This could be mounted on top of the HoloLens and overlaid on its display. This would allow the user to see the temperature of its environment. In the absence of a thermal camera a simple thermometer could also be used to show the current temperature on the HUD.
Furthermore, it could also be very useful to directly monitor how much oxygen the firefighter has left in his tank and display it on their HUD. If this is not possible, it could be done by

measuring how much time it usually takes to deplete the tank and then subtract the time that it has been used.
Finally, other researchers have successfully replaced the HoloLens depth sensor to improve its resolution and range (Garon et al, 2016). Reproducing this would allow us to create more detailed scans of larger areas. This would not just improve the readability of the scan but also allow for more sophisticated classification and more accurate hologram positioning. The downside of this is that it increases the computing power and Internet bandwidth necessary for the application to function.

Holographic displays
The field of view of the HoloLens is quite small. This is mostly due to a lack of processing power. Next generation holographic displays such as the HoloLens 2 improve this. Increasing the field of view would greatly improve the usability of the application because it allows more information to be displayed without cluttering the user's vision.
Next generation holographic displays are also capable of tracking the user's eyes. This means that instead of following their head gaze, users could place information just by looking. This could also improve performance by rendering objects that the user is not currently looking, at a lower level of detail.

Networking
The development of ever faster, low latency networks such as 5G opens up a world of possibilities for *augmented reality*. Currently the HoloLens' computing power is limited by its small form factor. If computation could be offloaded onto a remote computer, this could greatly reduce its size. This would not just improve user comfort, but also allow for more complex calculations to be performed. The increase in computing power and bandwidth would, for example, make it more feasible to use the high resolution depth sensors described above.

Robotics
The Explorer application depends on a human to create the first scan of the environment, which subsequent explorers can then use to their advantage. This role of first responder is consequently more dangerous than the other rules due to the lack of available information. Developments in robotics have led to robots that can handle a variety of environments and can respond dynamically to changes. An example of such a robot is Boston Dynamics' SPOT (see figure 31). If such robots could be used to fulfill the first responder role this would reduce the risk for human responders. Furthermore, such robots are equipped with high resolution environment scanners and are not limited by the processing power of the HoloLens, which means a detailed map could be created before humans ever step into the building.

Figure 31: A potential replacement for human first responders. Source: Boston Dynamics

8

Project Organisation

This Geomatics Synthesis Project lasts 10 weeks during which the focus was put on the product development but also on the project organisation. This chapter is dedicated to the project organisation part.

8.1 Contributions and responsibilities of team members

When we started working on the project we decided that we will make equal contributions in all aspects of the project. This means that everyone did research, wrote reports, attended meetings, wrote code, tested code and reviewed the work of other team members. In the first two weeks, we all took time to get used to the new environment that Unity supports and to C#. Then we reached a point where all of us were able to split coding among team members. The main technical contributions were as follows:

Semantic enrichment of meshes	→ Camille and Charalampos (Babis)
Navigator front end and user testing setup	→ Robin
Explorer front end & back end	→ Max
Navigator back end & Backbone	→ Jos

However, after the midterm presentation, we made an assessment of how our team work was at that point and we all agreed that we were proud of the fact that all of the team members have made a substantial contribution to the current codebase of the project. This was not an easy task since we had to work with an inherited codebase within a large application like Unity.

Although we were happy with the result that we had so far we also realised that the literature research and project part had been left behind. Based on that we decided that for the rest of the project, Max and Jos would focus more on the product management part while Camille and Charalampos (Babis) would take care of the literature research and the project management part (see Figure 32). Robin focussed on the UI and setup of user testing scenarios combining both. All of us still attended the meetings and worked together when needed.

Besides the contributions every team member made, each member also got a role assigned, for which they were responsible throughout the project. These roles were used to create structure within the organisation of this project. However, the division of roles changed after the midterm presentation. We swapped the roles as shown in the following table:

	Old	New
Charalampos (Babis)	*Communications (Internal & External)*	***Project** Manager*
Camille	*Editor*	*Editor*
Jos	*Project Manager*	***Product** Manager*
Max	*Software development*	*Technical Chief*
Robin	*Quality Control*	*Quality Control*

Table 3: Old and New roles of team members

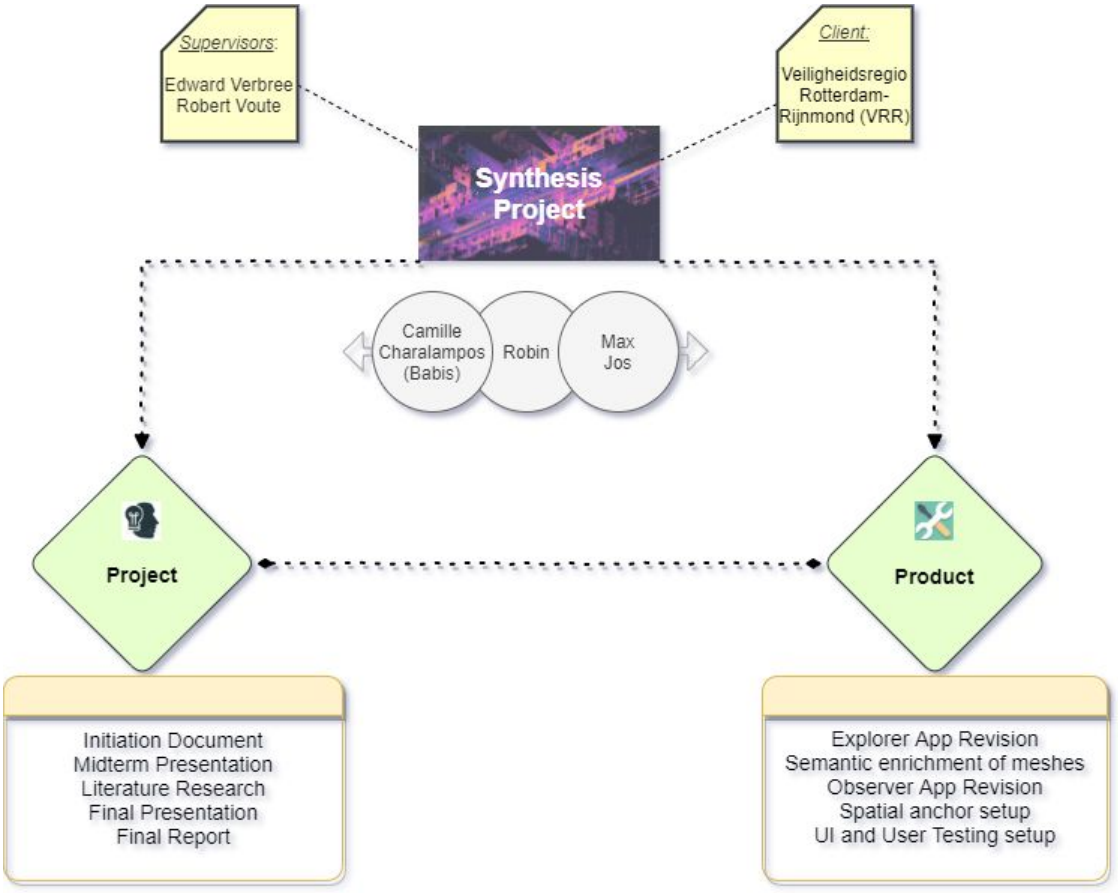


Figure 32: General Project Plan. Source: Author

8.2 Communication plan

Meetings

Days dedicated to the project were Mondays, Wednesdays and Fridays. Every Monday morning we met to discuss which tasks each of us was going to work on during the following week and on Fridays, we also used to make a meeting in order to discuss what each of us managed to achieve or not during the past week. In preparation for those meetings, an agenda was usually created with the subjects to be discussed during the voice call. Client and supervisor meetings were planned on Wednesdays. During those meetings, we demonstrated the current progress and talked about future developments. All comments and suggestions by the clients were documented during those sessions and distributed afterwards.

Communication tools

During the project we used the following tools to ensure proper communication :

- Discord server: we used Discord for our voice meetings (team and client meetings) and text-based communication spread over various channels, which also made it possible to exchange useful ideas, videos, files, etc.
- Git and GitHub: we used Git combined with GitHub as our version control system to allow easy sharing of our code and other data.
- Google Drive: we used Google Drive to share written documents like literature, reports, calendars and minutes.
- Google Calendar: we made a google calendar to keep track of the meetings and important deadlines.

8.3 Agile software development

During this project, we more or less followed the agile *software development* (see figure 33). In practice, this means that we had weekly sprints following the agile cycle as shown in the image. As we started the week with a meeting on Monday and ended it on Friday, we could easily incorporate weekly sprints. On Monday we had the planning phase and design phase, during the week we had the development and testing phase and on Friday we had the review phase. Once in a while throughout the weekly sprints we also deployed the new functionalities on the master merging them with the existing functionalities. The launching phase happened when we finished this project. So, we diverted a bit from the actual agile *software development* but most of the components were present throughout our project.

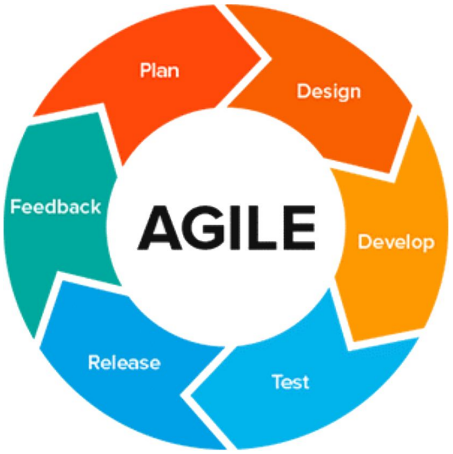


Figure 33: Agile software development. Source: vacationclubsoftware.com

8.4 Risk management

While we worked on a big team project and under special circumstances we had to be prepared for certain risks and uncertainties that may arise. When we started working on the project, we identified several risks. Some were linked to the current Coronavirus crisis while others had to do with other internal or external factors. Those risks are addressed in the coming section as well as how we managed them.

Work remotely
One of our main concerns was the fact that all of us were in different places, even in different countries, during the whole project realisation and we had to work remotely. Working remotely within a big team project was something new for all of us and it has been challenging. However, we quickly got used to that new way of working and thanks to well-organised meetings, it worked well for all of us. It still made it a bit more complicated to follow what each team member was doing and made the application testing more challenging as we were in separate places but we handled those challenges in the end.

Ineffective management of strengths and weaknesses of each member
Each team member has different strengths and weaknesses. Moreover, we all come from different backgrounds so the efficient management of our skills and weaknesses was essential. However, as everyone must learn from the project, we started working on the first weeks by making equal contributions in all aspects of the project. So everyone was doing research, coding, report writing, etc. After the midterm, we focussed on using more of each other's skills and weaknesses for finishing the final product and project. This combination of at first making equal contributions and then in a second plan taking advantage of each other's skills worked fine and brought all benefits of both approaches.

8.5 Plan and tasks repartition over the weeks

The various stages of the project were spread over 10 weeks. During weeks 1 and 2, our main focus was the initiation document, essentially to define the scope of the project and a methodology. During weeks 3 and 4, we developed that methodology, followed Unity and C# tutorials related to our project and created some separate Unity demos of functionalities that we wanted to implement in our system. For this, we also needed to get familiar with the code provided by Bart-Peter. At week 3, we also started *software development* and by the midterm presentation we had an implementation of our must-have functionalities and of some of the should-haves. Thus, at the midterm, we had already progressed a lot in the anchor point cloud synchronisation of two HoloLenses, in the UI development and in the data visualisation with added semantics. After the presentation we created a more advanced system with many of the should-have and could-have functionalities such as advanced semantics, minimaps, better visuals etc. and we started writing on the final report. The second-last week was mainly dedicated to product testing and final report writing. The final week will be dedicated to preparations for the final presentation (see figure 34).

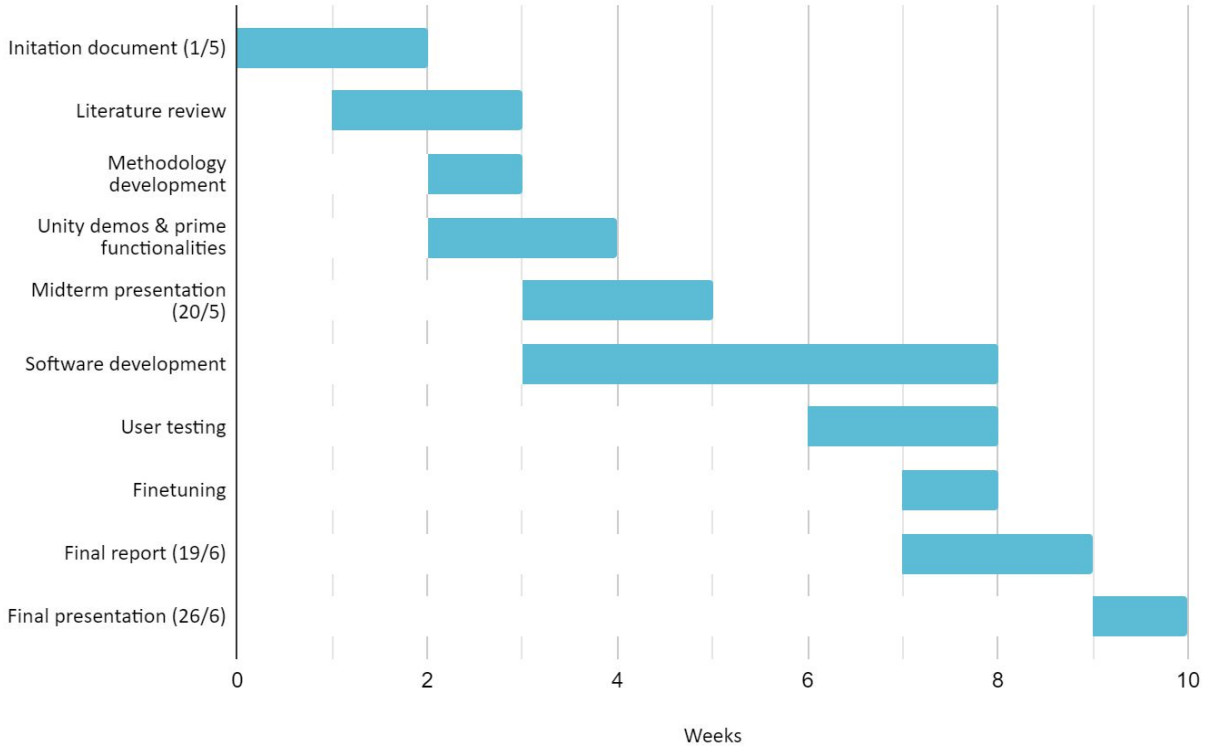


Figure 34: Project planning Gantt diagram. Source: author

8.6 Involvement of the clients

During this project, the clients have assisted us by providing us the hardware and the code from their current research, which aimed at using indoor point clouds to support tactical decision-making in fire emergencies. *CGI* also lent us a HoloLens for the full duration of the project and we also used their Azure Cloud account for every online test we made. The other HoloLens was provided by the VR Zone from TU Delft. On a less technical aspect, a user requirements interview was also done with the clients in the first week. Specific information regarding the correct direction that we must follow during the implementation of our product was given during this interview. Moreover, during the meetings with the client, insights about the deployment, the type of information that is most needed by the firefighters and guidelines about the interface of the final product were discussed as well.

Glossary

Terms defined in the glossary are written in italics within this report.

Augmented reality (**AR**) is a situation in which the understanding of the real-world environment is improved by overlaying that environment with virtual data (Klopfer and Squire, 2008).

Azure Spatial Anchor (ASA) is a proprietary Microsoft service that allows holograms to be linked to a point in space. These are used to establish a shared coordinate system between multiple HoloLenses. For more explanation see 4.5.1 - Shared Coordinate System.

CGI is a consulting firm founded in 1976 that provides IT consulting services, systems integration services, intellectual property solutions and services for business management.

Incident commander is the person that has the overall responsibility during the fire emergency operation. He is the one that defines the objectives and priorities based on the incident characteristics (Broder and Tucker , 2012).

Inertial measurement unit (IMU) is an electronic device composed of accelerometers, gyroscopes and magnetometers to measure force, angular rate and orientation (Mautz, 2012).

Information system is a formal, sociotechnical, organisational system designed to acquire, process, store and distribute information. It is composed of four components: task, people, structure and technology (O'Hara, 1999).

Mixed reality (MR) is a situation in which real-world information and digital information are mixed in a way that allows users to interact with virtual objects and virtual objects to interact with the surrounding environment. It is often used interchangeably with the term *augmented reality (AR)* but it actually defines a broader interpretation (Craig, 2013b).

MoSCoW method is a project management method that consists of separating the features of the envisioned product into four categories: must-, should-, could-, and won't haves.

Simultaneous localisation and mapping (**SLAM**) is the computational geometry problem of mapping an unknown environment while simultaneously locating the mapping device within it (Durrant-Whyte and Bailey, 2006).

Situational awareness refers to the perception and understanding of the elements of the environment within a time and space context and the projection of their status in the future (Gundersen, 2013).

Software development kit (SDK) is a collection of tools and programs used to develop applications for specific platforms or programming languages. This kit consists of various things including documentation, snippets of code, libraries and guidebooks.

TErrestrial Trunked RAdio (TETRA) is a narrowband land mobile radio used for low-speed and voice data communication. It is an ETSI standard. (Ulema, 2018).

Wifi fingerprinting is an indoor positioning technology that works on comparing live received signal strength with a database of pre-registered signal strength. The user is located in the room that has the registered signal strength the most similar to the live signal strength (Mautz, 2012).

Wireless ad hoc network (WANET) or **Mobile ad hoc network (MANET)** is a decentralised network as ad hoc means that it does not use an existing wireless infrastructure. Instead, the network consists of a collection of at least two devices equipped with wireless communications and networking capability (Han, 2004).

Wireless sensor network (**WSN**) refers to a collection of spatially dispersed sensors forming together a self-configured network for monitoring physical conditions of the environment (temperature, sound, pressure,...) and organizing the gathered data so that they can be observed at a central location (Matin, 2012).

References

Abulhassan, Y., DeMoulin, D. (2017). Locating Firefighters in Immediately Dangerous to Life or Health Environments. In: Proceedings of the Human Factors and Ergonomics Society Annual Meeting, vol. 61, nr. 1, pp. 1702–1705, doi: 10.1177/1541931213601914.

Agugiaro, G. (unpublished). Lecture notes of GEO1003 Geographical Information Systems (GIS) and Cartography, 2019.

Arif, S. M. U., Mazumdar, P., Battisti, F. (2019). A Comparative Study of Rendering Devices for Safety-Critical Applications in Operative Control Rooms, 11th International Symposium on Image and Signal Processing and Analysis (ISPA), Dubrovnik, Croatia, 2019, pp. 282-287, doi: 10.1109/ISPA.2019.8868791.

Broder, J. F., Tucker, E. (2012). 12 - Emergency Management – A Brief Introduction. In: Risk Analysis and the Security Survey, Butterworth-Heinemann, pp. 101-112, doi: 10.1016/B978-0-12-382233-8.00012-1.

Craig, A. B. (2013a). Augmented Reality Concept. In: Understanding Augmented Reality, Morgan Kaufmann, pp. 39–67, doi:10.1016/b978-0-240-82408-6.00002-3.

Craig, A. B. (2013b). What Is Augmented Reality? In: Understanding Augmented Reality, Morgan Kaufmann, pp. 1–37, doi:10.1016/b978-0-240-82408-6.00001-1.

de Leoni, M., Mecella, M., de Rosa, F., Marrella, A., Poggi, A., Krek, A., Manti, F. (2007). Emergency management: From user requirements to a flexible P2P architecture. In: Proceedings of the 4th International ISCRAM Conference, Delft, Netherlands, 2007, pp. 271–279.

Docs.microsoft.com. 2020. *Frequently Asked Questions - Azure Spatial Anchors*. [online] Available at: <https://docs.microsoft.com/en-us/azure/spatial-anchors/spatial-anchor-faq> [Accessed 17 June 2020].

Durrant-Whyte, H., Bailey, T. (2006). Simultaneous localization and mapping: part I, IEEE Robotics & Automation Magazine, vol. 13, nr. 2, pp. 99–110, doi:10.1109/mra.2006.1638022.

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 Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. 4, pp. 271-278, doi:10.5194/isprs-annals-IV-2-W5-271-2019I.

Gamma, E., Helm, R., Johnson, R., Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, USA.

Garon, M., Boulet, P. O., Doironz, J. P., Beaulieu, L., & Lalonde, J. F. (2016, September). Real-time high resolution 3D data on the HoloLens. In *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct)* (pp. 189-191). IEEE.

Gundersen, O.E. (2013). Situational Awareness in Context. In: Brézillon P., Blackburn P., Dapoigny R. (eds) Modeling and Using Context. CONTEXT 2013. Lecture Notes in Computer Science, vol 8175. Springer, Berlin, Heidelberg.

Haase, J., (2017). New Arrangement Opportunities and Safety Challenges for Offices of the Future. In: Eibl, M., Gaedke, M. (Hrsg.), INFORMATIK 2017. Gesellschaft für Informatik, Bonn, pp. 1447-1456, doi: 10.18420/in2017_144.

Han, L. (2004). Wireless Ad-hoc Networks.

Hannah, M. J. (1988). Digital stereo image matching techniques. *International Archives of Photogrammetry and Remote Sensing*, *27*(B3), 280-293.

Hardy P., Field K. (2011). Portrayal and Cartography. In: Kresse W., Danko D. (eds) Springer Handbook of Geographic Information. Springer Handbooks. Springer, Berlin, Heidelberg.

Held, B. Aljuneidi, S. Pham, V. T., Joseph, A. (2019). Helon 360: A smart firefighters' helmet Integrated Augmented Reality and 360° Thermal Image Data Streaming, doi: 10.13140/RG.2.2.28158.41287.

HoloLens 2 (n.d.). HoloLens 2 : Get to know the new features and technical specs, viewed 17 June 2020, <https://www.microsoft.com/en-us/hololens/hardware>.

Hosch, W. (n.d.). Augmented reality : Computer science, Encyclopedia Britannica, viewed 12 May 2020, <https://www.britannica.com/technology/augmented-reality>.

Huang, J., & You, S. (2012, June). Point cloud matching based on 3D self-similarity. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops* (pp. 41-48). IEEE.

Jiang, X., Chen Y. N., Hong, I. J., Wang, K., Takayama, A. L., Landay, A. J. (2004). Siren: Context-aware computing for firefighting. In: Proceedings of the 2nd International Conference on Pervasive Computing (Pervasive 2004), Vienna, Austria, 2004, pp. 87-105, doi: 10.1007/978-3-540-24646-6_6.

Kipper, G., Rampolla, J. (2013). What Is Augmented Reality? In: Augmented Reality, Elsevier, United States, pp. 1–27. doi:10.1016/b978-1-59-749733-6.00001-2.

Klann, M. (2008). Tactical navigation support for firefighters: The LifeNet Ad-Hoc SensorNetwork and Wearable System. In: Proceedings of the Second International

Workshop on Mobile Information Technology for Emergency Response (Mobile Response 2008), Bonn, Germany, pp. 41-56, doi: 10.1007/978-3-642-00440-7_5.

Klopfer, E., Squire, K. (2008). Environmental detectives: the development of an augmented reality platform for environmental simulations. Educational Technology Research and Development, vol. 56, nr. 2, pp. 203–228, doi: 10.1007/s11423-007-9037-6.

Ledoux, H., Ohori, K. A., Peters, R. (2019). Point cloud processing. In: Computational modelling of terrains, Delft.

Lemmens, M. (unpublished). Lecture notes of GEO1001 Sensing Technologies for the Built Environment, 2019.

Matin, M., Islam, Md. (2012). Overview of Wireless Sensor Network. doi: 10.5772/49376.

Mautz, R. (2012). Indoor Positioning Technologies, ETH Zurich, Department of Civil, Environmental and Geomatic Engineering, Institute of Geodesy and Photogrammetry, doi: 10.3929/ethz-a-007313554.

Nguyen, T. A. B., Englert, F., Farr, S., Gottron, C., Bohnstedt, D., Steinmetz, R. (2015). Hybrid communication architecture for emergency response—An implementation in firefighter's use case. In: Proceedings of the 2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops), Pune, India, 2015, pp. 524-529.

Nystrom, R. (2014). Game programming Patterns, Genever Benning.

O'Hara, Margaret & Watson, Richard & Kavan, C.. (1999). Managing the three Levels of Change. IS Management. 16. 63-70. 10.1201/1078/43197.16.3.19990601/31317.9.

Prasanna, R., Huggins, T.J., Yang, L. (2017). Information Systems Architecture for Fire Emergency Response, Journal of Enterprise Information Management, vol. 30.

Richmond, V. L., Rayson, M. P., Wilkinson, D. M., Carter, J. M., Blacker, S. D. (2008). Physical demands of firefighter search and rescue in ambient environmental conditions, Ergonomics, vol. 51, nr. 7, pp. 1023-1031, doi: 10.1080/00140130801939709.

Roh, T. H. (2015). Integrated command system for firefight safety in special disaster area, Fire Science and Engineering, vol. 29, nr. 6, pp. 98-108.

Rosenbauer (2017). HEROS-titan Firefighters Helmets, Rosenbauer, viewed 12 June 2020,
<https://www.rosenbauer.com/en/de/rosenbauer-world/products/equipment/firefighting-helmets/heros-titan>.

Salmon, P.M., Walker, G.H., Stanton, N.A. (2016). Pilot error versus sociotechnical systems failure: a distributed situation awareness analysis of Air France 447, Theoretical Issues in Ergonomics Science, vol. 17, nr. 1, pp. 64-79.

Sha, K., Shi, W., Watkins, O. (2006). Using wireless sensor networks for fire rescue applications: Requirements and challenges. In: Proceedings of the 2006 IEEE International Conference Electro/information Technology, East Lansing, MI, 2006 , pp. 239-244.

Shi, W., Ahmed, W., Li, N., Fan, W., Xiang, H., Wang, M. (2019). Semantic Geometric Modelling of Unstructured Indoor Point Cloud. *ISPRS Int. J. Geo-Inf.*, vol. 8, nr. 1, doi: 10.3390/ijgi8010009.

Speicher, M., Hall, B. D. Nebeling, M. (2019). What is Mixed Reality? In: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19). Association for Computing Machinery, New York, NY, USA, Paper 537, pp. 1–15, doi: 10.1145/3290605.3300767.

Thrun S. (2007). Simultaneous Localization and Mapping. In: Jefferies M.E., Yeap WK. (eds) Robotics and Cognitive Approaches to Spatial Mapping. Springer Tracts in Advanced Robotics, vol 38. Springer, Berlin, Heidelberg.

Ulema, M. (2018). Terrestrial Trunked Radio (Tetra). Pp. 87-106. Doi: 10.1002/9781119369554.ch6.

van der Meer, T.W.T., Verbree, E., van Oosterom, P.J.M. (2018). Effective cartographic methods for assisting tactics choice and indoor deployments during building fires - A case study the Dutch Fire Brigade, Remote Sensing and Spatial Information Sciences, vol. XLII-4, doi: 10.5194/isprs-archives-XLII-4-655-2018.

Verbree, E. (unpublished). Lecture notes of GEO1003 Positioning and Location Awareness, 2020.

Viola, P., & Jones, M. J. (2004). Robust real-time face detection. *International journal of computer vision*, *57*(2), 137-154.

Wilson J, Steingart, D., Romero, R., Reynolds, J., Mellers, E., Redfern, A., Lim, L., Watts, W., Patton, C., Baker, J., Wright, P. (2005), Design of monocular head-mounted displays for increased indoor firefighting safety and efficiency. In: Proceedings of the Conference of SPIE: Helmet- and Head-Mounted Displays X: Technologies and Applications, Orlando, FL, 2005, pp. 103-114, doi.org/10.1117/12.603590.

Wilson, J., Bhargava, V., Redfern, A., Wright, P. (2007). A wireless sensor network and incident command interface for urban firefighting. In: Proceedings of the 4th Annual International Conference on Mobile and Ubiquitous Systems (MobiQuitous 2007): Computing, Networking & Services, Philadelphia, PA, 2007, pp. 1–7, doi: 10.1109/MOBIQ.2007.4450980.

Wu, H.-K., Lee, S. W.-Y., Chang, H.-Y., & Liang, J.-C. (2013). Current status, opportunities and challenges of augmented reality in education. Computers & Education, vol. 62, pp. 41–49, doi:10.1016/j.compedu.2012.10.024.

Yuan, C., Yu, X., & Luo, Z. (2016, July). 3D point cloud matching based on principal component analysis and iterative closest point algorithm. In *2016 International Conference on Audio, Language and Image Processing (ICALIP)* (pp. 404-408). IEEE.

Qwake Tech, (2018). C-THRU, viewed 12 June 2020, <https://www.qwake.tech/>.

Appendix A Questionnaire

29-04-2020 Huib Fransen Questionnaire

Description:
We intend to answer the case described by the VRR by making a tool / app / software package. To get started in the right direction, we would like to know what you as a client require out of this product, and if you are willing to contribute anything to make sure we keep heading in the right direction (maybe let a couple of real firefighters test the product?)

Requirements

1) What is your function in the firefighters team when working on the field?

A: I work as a commanding officer

2) What are the most important parts of the buildings that the firefighters teams need to have information about?
A: first of all the location of the fire? Can they reach it? (what distances to cover?), Can they bring enough water there to extinguish the fire? How to ensure the safety of the firefighters? The proximity of people in the premises? The number of people involved? How to evacuate or rescue them?
the expansion of the fire to other floors/ Adjacent buildings or wings of the building
Are there dangerous goods (eg. Flammable liquids, stores, gas tanks, toxic , radiation, acids etc)
What are the building characteristics you need to know for a
safe deployment? (windows, number of stories,...)
A: selected (dedicated) fire entrances, straiways, elevators (suitable for fire brigade),

3) What type of information do you want from the information system?
A: fireman positions/tracking, where they are,
3b) Is the geometry of the building enough or would you like surroundings?)
A: yes please!

4) What are the advantages and what are the disadvantages of your current information system? Which aspects need further improvement?
A: adv.: it works! We are owner of the system can

5) What display device would be best for the firefighters teams to use on the field? (wrist mounted, hand-held device, HoloLens...)
A: I suggest some sort of 'heads-up display' mixed view so you keep your hands free

6) Which of 2D and 3D indoor representations are more suitable for safe development (or the combination)?
A: I suggest to go further with the concept of Togglemaps (thesis Tom vder Meer). It makes use of 2D and 3D simultaneously (a bit like Doom, 1th generation)

7) What are your requirements about the shared interface (an interface that allows the communication and share of data between teams and control room)?
A: I'm not sure if I can answer this. I know of other experiments eg. 'Red Suit' of safety region Twente (check it!)

8) Are there any other requirements that we might have missed, but are important for the product we are creating?
A: easy access, easy start-up,

9) What kind of GIS systems (hardware and software) are currently involved in deployment? (basically do they have GPS, INS etc. in their vans)
A: GPS (tom-tom) and tablet with MOI (Mobile Operational Information), thermal imaging camera (TIC), LEL-meter (explosive mix, eg. Detection of Gas leaks)

10) What kind of dangers do you encounter during deployment?
A: getting disorientated! You move in an unfamiliar building, filled with toxic smoke, no/ bad sight, danger of collapse, explosion, electrocution. And fireman wear breathing apparatus with worth of 15-20'airsupply, they have to monitor this closely

Contributions
1) Please describe the main technical and non-technical aspects that you could contribute to our project (code, data from the field,...)

A: different options:
1. Further enhance Togglemaps, positioning, mix 2D/3D

2. post PoC 3D-scans Wilhelminapier R'dam: scan with hand scanner is vulnerable. + Don't forget the organisational aspects as well: owner en user of the building might be different persons. Consider privacy aspects – not everybody likes to be 'scanned' as well as security issues (scanning entrance building etc.). New suggestion (next PoC) is to make use of existing 2D drawings of floors in the building and mix them with City GML and BIM (if available)

3. Make an operational scanning procedure for post – incident activities, eg. Evaluation, Fire investigation and or training purposes. Using the HoloLens

Appendix B Testing Scenario

Scenario fire at library TU Delft

Testing synced HoloLenses

Date: 15/06/2020

Aim:

Exercise and live testing of HoloLenses to orientate with 3D-mapping

Sub-targets:

Is it possible to move, orientate, communicate and show location on tablet

Needed:

- 2 HoloLenses, lap-top for operational officer , hard-copy drawings of building
- Permission to access and execute exercise in library

Timetable:

- 30'prepping location, hardware, briefing of teams
- 30' executing exercise
- 30'debrief

Scenario:

Location of exercise is library TU Delft

No smoke, fire involved. 2 students with HoloLens will move through the building trying to find a designated location.

Alarm message:

At ..:.. hr (time) Fire detection system alerts smoke in room 21.00.00840

Operational officer gives orders – stays outside of building provides teams with tasks – stays connected with teams with Wifi/ laptop

Task #1 team 1: “Go to the designated area and make a reconnaissance. Report for fire, smoke and remove persons involved”

[see attack-route on map 1 +2]

Team 2: on hold

Task #2:

Team 1: Go investigate adjacent area (horizontal planar) in room 21.00.00.880

Team 2: Go investigate level above fire, first floor, check rooms 21.00.0182 and 21.001780 (see route on map)

Meanwhile keep communications with each other: check conditions, location and progress of fire and safety

Task #3:

Team 1 +2 meet each other near staircase on x,y vector K/L- 15, brief each other and compare notes

EoEx:

- Debriefing, after action review:
    - What was the plan?
    - What happened?
    - Why did it happen?
    - What can we learn from this?

The paths of the two teams are shown in the figures below.

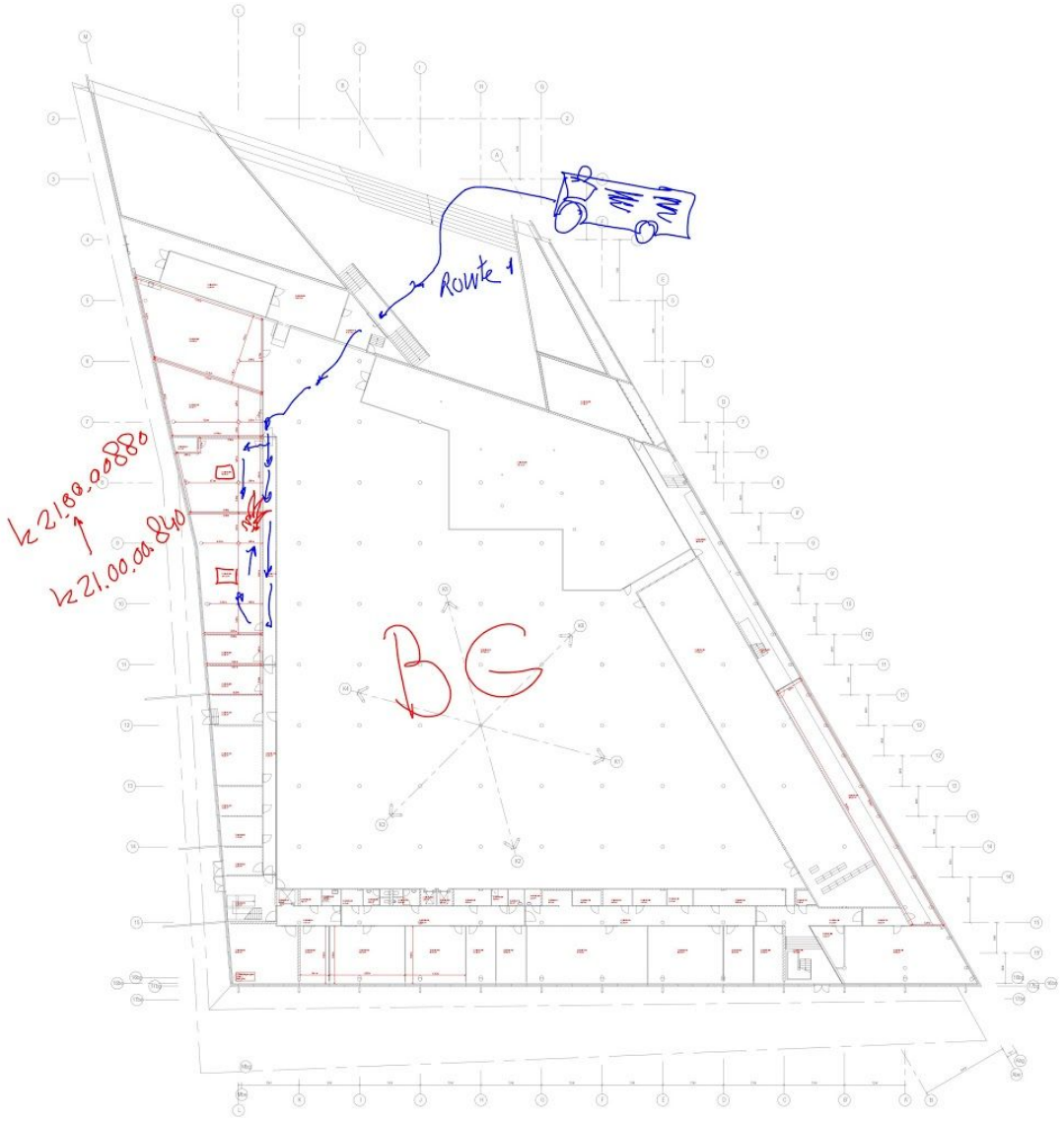


Figure 34 : path of team 1

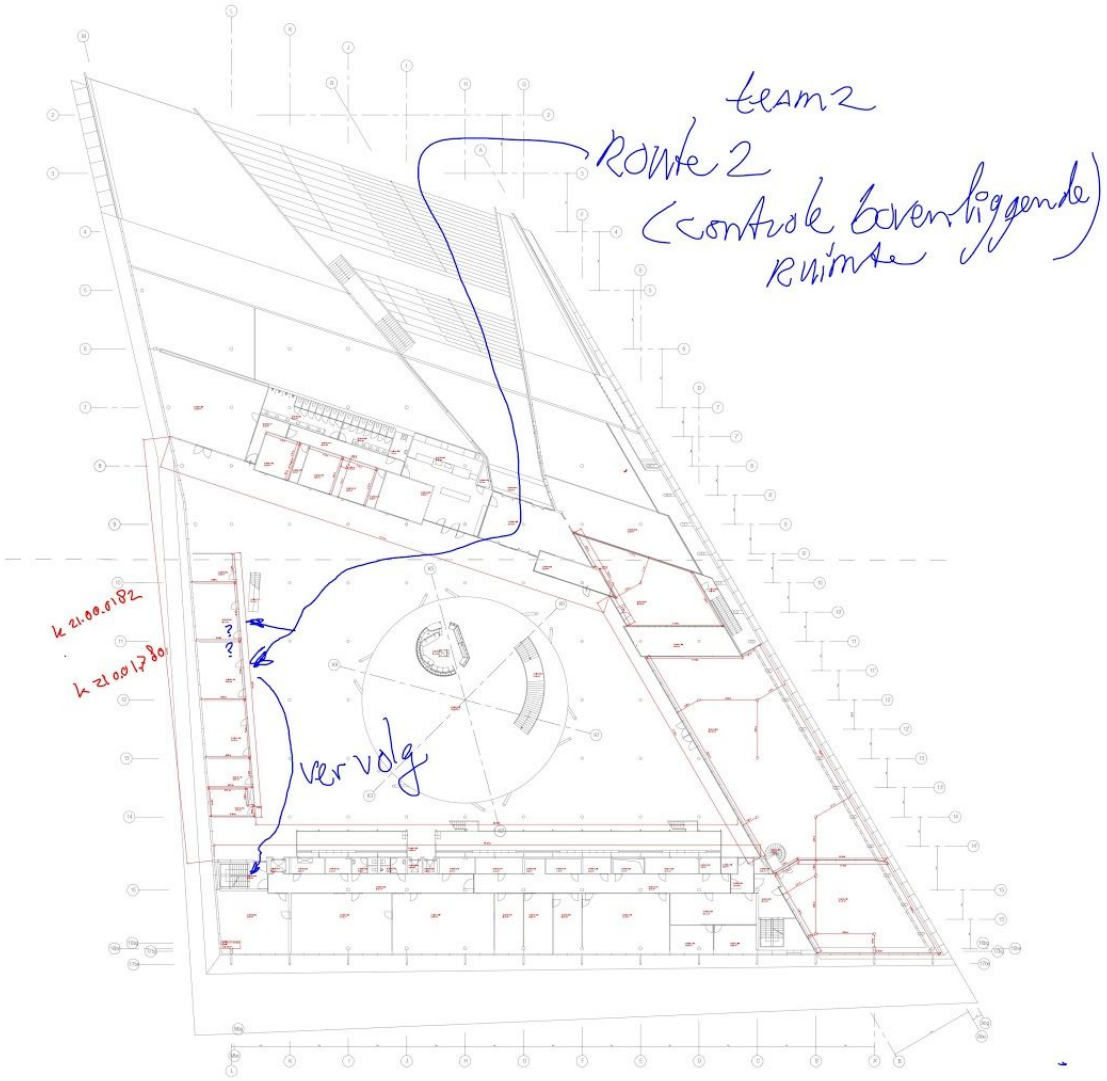
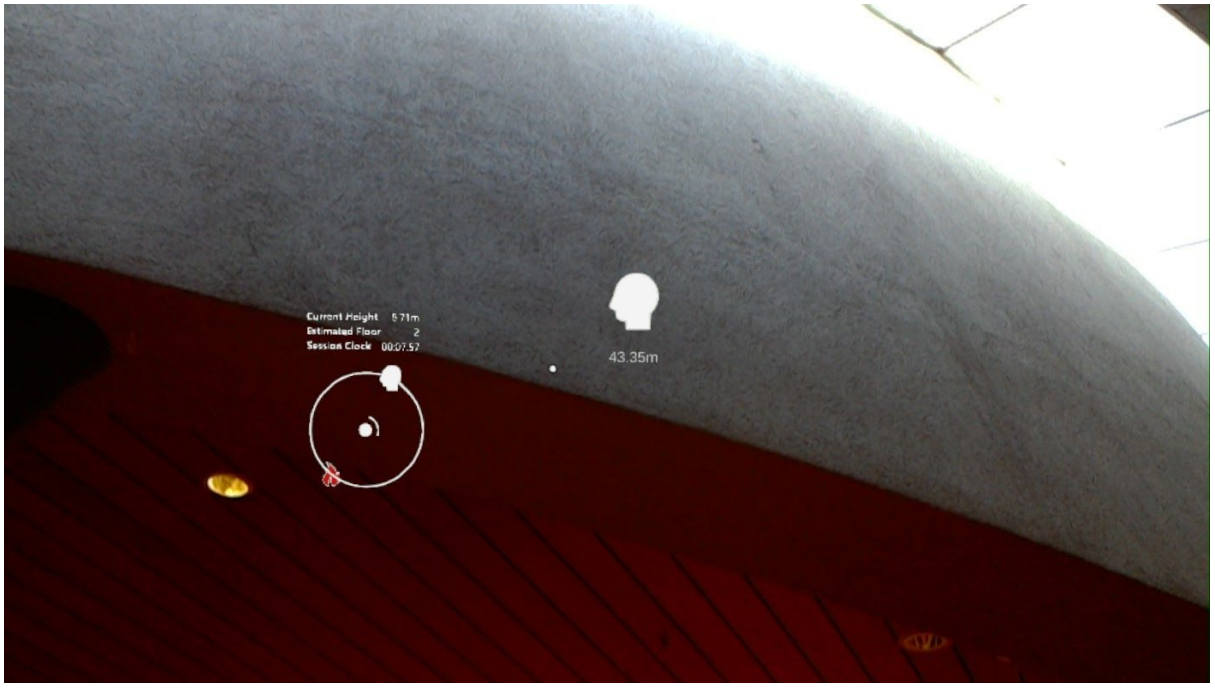
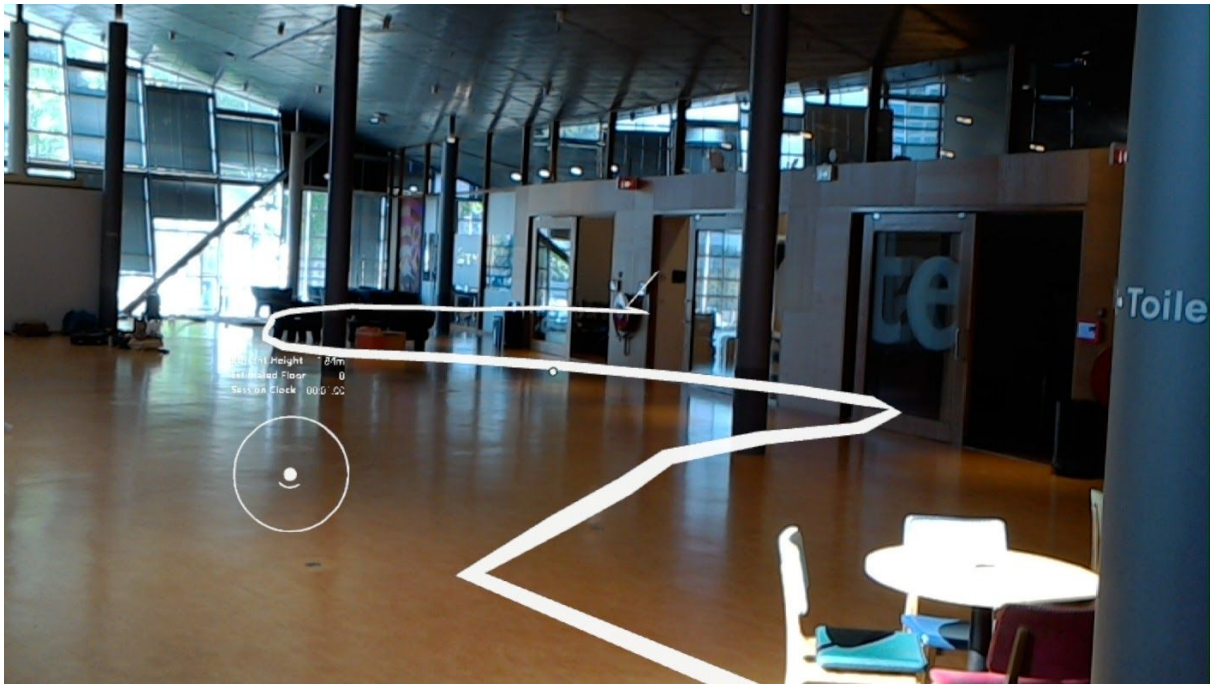


Figure 35 : path of team 2

Appendix C Explorer Screen Captures


Current Height 5.71m
Estimated Floor 2
Session Clock 00:07.57
43.35m


Toile