MSc. Thesis
Geomatics for the Built Environment

INDOOR POSITIONING USING AUGMENTED REALITY

Laurens Oostwegel
July 2020

TU Delft

CGI

MSc thesis in Geomatics

Indoor Positioning using Augmented Reality

Laurens Oostwegel

July 2020

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of Master of Science in Geomatics

The work in this thesis was carried out in the:

3D geoinformation group
Department of Urbanism
Faculty of the Built Environment & Architecture
Delft University of Technology

in cooperation with:

CGI The Netherlands

Supervisors:	Ir. Stelios Vitalis
	Dr. Ken Arroyo Ohori
Co-reader:	Ir. Martijn Meijers
Company supervisor:	Ir. Robert Voûte

Abstract

Unlike outdoor environments, there is no wide-spread solution to positioning inside a building. Indoor solutions rely on pre-installation of infrastructure, such as Bluetooth beacons or ultra-wide band technology. Recently, there has been growing interest in the use of Augmented Reality (AR) for indoor positioning. AR devices use an algorithm known as Simultaneously Localisation and Mapping (SLAM) to scan an environment and find a position inside. This make it possible to estimate a position on-the-fly without any pre-installations. The Microsoft Hololens (MH) is a head-mounted display AR device that is able to perform SLAM. The use of SLAM for indoor positioning can be beneficial in the case of Emergency Response (ER). The place of impact in ER is unknown and there is no time to install any infrastructure beforehand.

Two problems arise with the use of SLAM for indoor positioning. (1) A position is a pin point in space that is defined by a reference frame. In the case of SLAM, the frame is the scanned object. In most situations, a map or floor plan is more appropriate as reference frame, in order to give a full context. (2) The SLAM algorithm suffers from drift, a growing error over time. This research tries to solve these problems using shape registration. The SLAM output can be aligned to a reference floor plan, by use of a spatial matching technique. This alignment is a continuous process to account for the drift errors of the SLAM algorithm.

Three spatial matching techniques are compared: Iterative Closest Points (ICP) iteratively tries to minimize the distance between two shapes using least squares; Instantaneous Kinematics (IK) is a variant on ICP that makes use of a velocity vector; and Hough Transform makes use of the Hough domain properties, where rotation is invariant to translation and scale. These algorithms are compared on their accuracy, computation time and robustness.

It is concluded that the Hough Transform algorithm gives the most accurate results and is fastest. In this research it was found that an average accuracy of $<$ 1m can be maintained over 80% of the experiments, with maximum errors up to 5 meter. In 20% of the experiments, the error can extremely diverge up to $>$100 meter. It is suggested that the quality of the scan and the existence of artefacts (e.g.furniture, people) are the cause of these errors. That makes the method unsuitable for indoor positioning in the case of Emergency Response, that needs extremely reliable systems. Research is needed to create a more robust method, that uses better outlier detection methods. However, the results are promising and do open the door for indoor navigation using Augmented Reality.

Acknowledgements

Before you lies the product of work done the past months. Work not only of one individual, but with many others involved. This research would never have been established, without the help of these:

First I would like to express my gratitude to Stelios Vitalis and Ken Arroyo Ohori, who guided me through the full process. Thank you for your feedback and support in our biweekly meetings. You helped me to achieve a more academic mindset. I would also like to thank Robert Voûte, who gave me the opportunity you to work in a professional environment. Your enthusiasm and dedication have truly helped to take this thesis to a higher level. Lastly, I would like to thank Martijn Meijers for co-reading the thesis.

I look back with delight on the many coffee moments, virtual meetings and activities together with the CGI trainees. Thank you for giving so much input on my project and the many great discussions. I would like to offer my special thanks to Floor and Bart-Peter for proofreading this thesis. Thank you Maruša for listening endlessly to not only my presentation, but also to my countless monologues about excitements, frustrations, breakthroughs and bugs that I encountered these months. On top of that, you made the amazing front page of this thesis, for which I am very thankful.

Last but not least I would like to give a special thanks to my parents. During the writing process I stayed with you for a while. You helped me focus on the thesis when I most needed to. Thank you for your encouragement and support through all these months.

...

Laurens Oostwegel

June 2020

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

AoA Angle of Arrival . . . 7
AR Augmented Reality . . . 1
BIM Building Information Modeling . . . 2
DoA Direction of Arrival . . . 7
ER Emergency Response . . . 1
GPS Global Positioning System . . . 1
HT Hough Transform . . . 3
ICP Iterative Closest Points . . . 3
IK Local Quadratic Approximation and Instanteneous Kinematics . . . 3
IMU Inertial Measurement Unit . . . 10
INS Inertial Navigation System . . . 8
LQAIK Local Quadratic Approximation and Instanteneous Kinematics . . . 27
MH Microsoft Hololens . . . 1
MOI Mobiel Operationeel Informatiesysteem . . . 6
MRTK Mixed Reality ToolKit . . . 38
MSE Mean Square Error . . . 32
PCA Principal Component Analysis . . . 12
PTAM Parallel Tracking and Mapping . . . 11
RFID Radio Frequency Identification . . . 8
RMSE Root mean Square Error . . . 47
RSSI Received Signal Strength Indicator . . . 7
SLAM Simultaneous Localization and Mapping . . . 1
SVD Singular Value Decomposition . . . 12
TDoA Time Difference of Arrival . . . 7
ToA Time of Arrival . . . 7
ToF Time of Flight . . . 9
UAVs Unmanned Airborne Vehicles . . . 1
UWB Ultra-wideband . . . 1
VR Virtual Reality . . . 9
WLAN Wireless Local Area Network . . . 8

1 Introduction

It is difficult to imagine the world nowadays without the Global Positioning System (GPS). Navigation applications are omnipresent and every mobile phone has the ability to tell you how to go from point A to B. However, this concerns outdoor navigation. Reliable indoor positioning and localization is still a challenging task. There is no large-scale solution: the signals coming from satellites are too low for GPS to function in an indoor environment. Most positioning systems inside buildings rely on pre-installed infrastructure, such as Ultra-wideband (UWB) or Bluetooth beacons, or use a priori information, such as Wi-Fi fingerprinting to position or localize indoors (Yang and Shao, 2015; Rubino et al., 2013; Molina et al., 2018).

All-the-while, the interest in indoor positioning is growing, as can be found in numerous sectors: public spaces, such as airports and shopping malls see the need for indoor positioning to navigate, due to the increasing complexity and size of indoor environments. The use of multi-media in cultural environments increased the interest to use location-based services to present information about nearby objects to visitors. Visually impaired individuals can greatly benefit from indoor positioning services in combination with audio-based navigation services. Lastly, the demand for positioning indoors is often discussed in the case of Emergency Response (ER), that can benefit from increased situational awareness in the case of a disaster.

1.1 Using Augmented Reality for indoor positioning

In recent times, applications that enhance the real world with virtual objects in real-time have emerged. Augmented Reality (AR) allows you to interact with the environment in a new way. Hardware platforms using AR could be your every-day smartphone, that can achieve simple AR applications. The more complex AR systems involve an understanding of the environment and need more specialized devices. One such device is the Microsoft Hololens (MH). The Hololens is a head-mounted display that can visualize 3D models and is able to have these interact with the environment. It has its own processing unit and runs on a Windows 10 OS (Kim et al., 2017).

A technique that is used in AR has attracted the attention for indoor positioning: Simultaneous Localization and Mapping (SLAM) is a method that makes use of sensors, such as RGB cameras, depth cameras or laser range finders, to continuously map a space and find its own location inside the mapped space. In AR it is a necessary functionality, in order to walk around in a partly virtual and partly real world. To be able to walk around an object, the position of the object within the space, as well as the position of yourself needs to be known. SLAM is not limited to AR, but is a technique also found in robotics, driverless cars and Unmanned Airborne Vehicles (UAVs). Multiple researches have proposed to use a SLAM-based method for indoor positioning (Brito et al., 2019; Rantakokko et al., 2011; Li et al., 2019a). They argue it is a promising method, because it allows for indoor positioning without any existing infrastructure, as the SLAM technique is able to both map an environment and position someone simultaneously.

1.2 Problem statement

In some scenarios it is impossible to use an indoor positioning system that needs any pre-existing infrastructure or survey. In the case of firefighting, first responders have no more than a couple of minutes to get familiar with the building beforehand. Therefore, pre-installation of a positioning system is not possible. However, it is crucial to know the positions of first responders inside a building, not only to

improve the safety of the personnel, but also to improve the efficiency of the operation. A navigation aid could help firefighters not only by giving directions to the place of accident, but also by finding a way back if situations get too dangerous. Positioning could help firefighters with situational awareness: visited rooms could be marked as such in the case of evacuation, limiting the amount of double-searched rooms; hazardous materials, but also elements useful in operations, such as fire extinguishers could be marked on the spot. For these kind of situations, standard positioning systems using Bluetooth or WLAN are not fit, so alternatives should be explored.

AR and SLAM have the potential to be used as accurate indoor positioning method in the case of ER. However, the SLAM method has two limitations: (1) the method only retrieves a position relative to the space that it scanned. Therefore, it is impossible to navigate to new spaces using only SLAM. (2) Over time the method will suffer from drift (Hübner et al., 2019; Li et al., 2019a). Drift is an error that gets bigger over time, making the algorithm less reliable if it is not accounted for. It is dependent on the device that is performing the SLAM how high the drift error is. Fonnet et al. (2017) proposed an approach to solve these issues using existing Building Information Modeling (BIM) data. The virtual building is roughly positioned manually according to the real-life situation. Afterwards the BIM is fitted to the mesh more accurately using a registration algorithm. This solves both issues: navigation to non-visited areas can be achieved using the existing model. Drift errors are minimized by recalibrating the devices position through registration. To our knowledge, no one has implemented this method yet.

The use of 3D reference data for indoor positioning seems the logical option. The verticality of buildings is one of the defining elements that makes indoor positioning differ from outdoor positioning. Large buildings have many floors, sometimes with more complex structures such as entresols or ramps. These are difficult to convey in 2D data. However, in the case of ER, 3D BIMs are usually not at hand, but 2D alternatives are accessible (Zlatanova et al., 2015). Therefore, the reference data model that is used will be constrained to two dimensions.

1.3 Research objectives and research questions

The goal of this research is to develop a method to estimate the position of the MH in a building, without use of any pre-existing infrastructure. Because the availability of floor plans is higher than the availability of 3D models, it is chosen to work with a floor plan as a reference to register the model scanned with SLAM to. While ignoring a dimension means that some data is lost, it is justified by (1) the assumed orthogonal configuration of walls in general and (2) the correct orientation of the MH SLAM, that automatically aligns the ground plane to the horizontal plane (for more information, see section 2.3). The positioning method has the potential to be used in situations such as Emergency Response, when there is no time available for installing infrastructure and when there is no 3D BIM at hand. The method should communicate an estimation of the position accuracy at real-time. The developed approach can ultimately help to improve and speed up decision making in ER situations. While the MH is used in this thesis, the method should generalize to other devices that use SLAM for positioning. It should also generalize to the use of 3D models instead of floor plans.

1.3.1 Research questions

Based on the the objectives, a main research question has been developed. The main research question is:

How can the Microsoft Hololens improve indoor positioning, using the on-the-fly produced mesh and an existing floor plan?

The following sub-questions will accommodate the main research question:

1. What spatial matching techniques are feasible to match a 3D mesh acquired in real-time from the Microsoft Hololens with an existing 2D floor plan?

2. What is the most promising spatial matching technique in terms of accuracy and speed?
3. How can the indoor positioning of the Microsoft Hololens be improved by making use of the researched positioning method?
4. In which cases does the researched positioning method fail to estimate the position on a 2D floor plan?

1.4 Research methodology

The objective of this thesis is to estimate the position of the MH on an existing 2D floor plan. To do this, this research is divided into three parts: (1) the selection of spatial matching techniques; (2) the implementation of the positioning method; (3) the evaluation of the implemented method.

(1) A literature review is conducted to find appropriate spatial matching techniques. These will be selected and compared on various criteria, such as complexity, ability to perform partial registration and robustness to outliers. Three registration algorithms are selected for implementation:

- Iterative Closest Points (ICP)
- Local Quadratic Approximation and Instanteneous Kinematics (IK)
- Hough Transform (HT).

These will be elaborated on in detail.

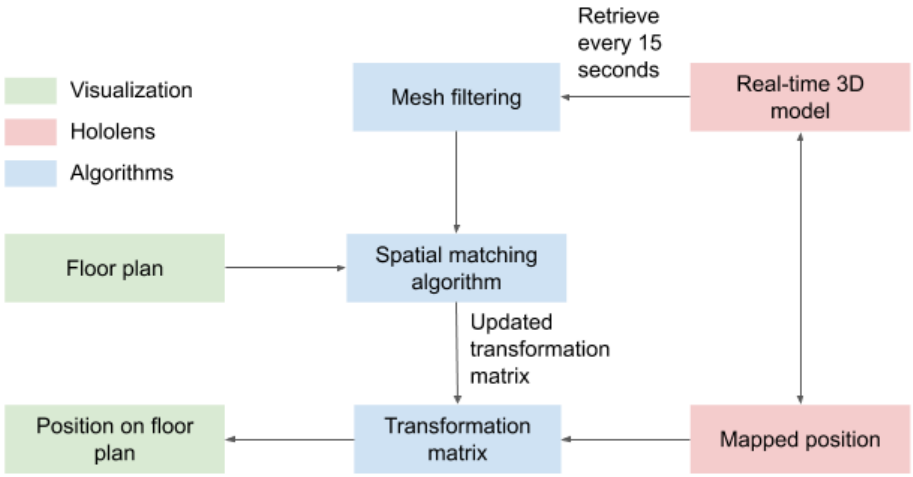


Figure 1.1: Workflow of positioning the Microsoft Hololens inside a 2D floor plan

(2) The method that is implemented in this thesis consists of a dual system. An illustration of the system can be found in fig. 1.1. On one side is the SLAM algorithm of the MH. This algorithm produces useable meshes in real-time, with the position of the device inside this 3D model. The other side of the system contains the floor plan and the position inside the floor plan. The aim of the method is to find the correct transformation matrix that transforms the position inside the 3D model to the position on the floor plan. To be able to do that, the meshes produced by the MH are retrieved every 15 seconds. These are filtered on vertical elements and transformed into a point cloud, in order to be fit for spatial matching. Using the spatial matching techniques that are found in the desk research, the point cloud is registered to the floor plan. The transformation matrix that is found to register the meshes to the floor plan is used to transform the position of the device inside the 3D model to the position on the floor plan. Every 15

seconds, the process is repeated to account for the drift errors.

(3) The Architecture building of the TU Delft will be used as a case study to evaluate the implemented algorithms. Ground truth positions will be measured in the building and all algorithms will be evaluated on three criteria:

- Computation time: the average time it takes to compute the algorithm.
- Accuracy: the accuracy of the computed position compared to the ground truth.
- Robustness: the capability to work with all sort of environments, included cases like outdated floor plans, big halls without walls or a bad initial registration.

These aspects will be explained in further detail in chapter 4.

1.5 Research scope

This thesis focusses on estimating the position of the MH on a 2D floor plan. A system is developed that works on the MH and shows its location on the 2D floor plan. While many 2D floor plans are in reality not to scale, in this thesis it is assumed that they are. This can be justified by the fact that the first responders have access to floor plans to scale. The method is evaluated on accuracy, speed and robustness. The method is tested on a general case, as well on particular cases that involve more complex forms, such as large rooms; incongruencies between floor plan and real world; and doors. While the proposed method should test in which of the particular cases the method fails, it is not meant to deal with all these cases successfully.

The processing power of the MH is mentioned as a limiting factor of the device (Brito et al., 2019). If the MH appears to have problems with the processing power, it suffices to establish a real-time connection with a computer and use that device for the computations. Although this is considered infeasible for real life situations, it does not detract from the method itself. This is justified by the fact that the first generation of MH is used in this thesis, that has less processing power than the already on the market second version of the MH.

1.6 Reading guide

The document will continue with the theoretical background in chapter 2. In chapter 3, the related work on the research questions will be discussed. Following that, the methodology is elaborated on in chapter 4. The case study is described in chapter 5. Afterwards the implementation is explained in chapter 6 and the results are shown in chapter 7. Lastly, the conclusion, discussion and future work is found in chapter 8.

2 Theoretical background

The following chapter will provide theoretical background on the topic of this thesis. The first topic that will be elaborated on is Emergency Response (ER) in section 2.1. Section 2.2 gives an overview of indoor positioning methods and their suitability to ER. In section 2.3, 2D and 3D solutions for positioning methods will be compared. The Microsoft Hololens and Augmented Reality in general will be introduced in the last section, 2.4.

2.1 Emergency response

Emergency response is part of disaster management (Zlatanova and Holweg, 2004). Disaster management is the *"understanding, managing and reducing of risks"* (Zlatanova and Fabbri, 2009, p.1) and is often used interchangeably with emergency management, risk management, crisis management or hazard management. Risk in this context means the *"probability for a negative, damaging outcome from an incident or by a natural event"* (Zlatanova and Fabbri, 2009, p.1). Disaster management can be divided into four phases: (1) mitigation; (2) preparation; (3) response and (4) recovery. A visualization of the emergency management cycle is shown in fig. 2.1. The first two phases find place before a disaster or hazard has happened. In the mitigation phase (sometimes mentioned together with prevention) measures are made to prevent hazards or disasters from happening, or to reduce the exposure to possible events. Dykes are an example of this, that stop rivers from flooding. Another example is the installation of fire alarms in buildings, that help catching fires in an early stage. In the preparation phase, active preparations are performed to get ready for a possible disaster. Rescue forces, such as the police, ambulance or fire brigade are trained during this phase. The response phase is directly after an emergency happened and can last from hours to days. Lastly, the recovery phase finds place after the acute emergency, where detriments are removed and long-lasting support is given. This fourth phase can last weeks up to years. Every phase of disaster management can take advantage of geo-information using diverse applications (Zlatanova and Fabbri, 2009; Zlatanova and Holweg, 2004). The type of data that is needed is dependent on the characteristics of each phase.

"Emergency response is the disaster management phase with the most extreme requirements" (Zlatanova, 2008, p.1631). In the ER phase the first hours after an event are the most critical. In these hours, the consequences of an event on people's lives and properties can be mitigated most. The dynamics of the phase are much higher than in other phases. Situations can change any moment. There are a lot of people involved, all with different responsibilities. Lastly, access to data and other sources might be obstructed (Zlatanova, 2008; Tashakkori et al., 2015).

In disaster management, spatial information is an important aid. Following the characteristics of ER, the type information that is needed to support rescue forces should be adapted. Data used for ER can be grouped into two categories: static and dynamic data. Static data already exists before a disaster has happened. It could be reference data (e.g. maps, photographs or 3D models) or more specific data, that is linked to an event, such as the location of fire hydrants or hazardous materials. The dynamic nature of disasters also holds the need for dynamic data and (near) real-time information. As situations can change any moment, continuous monitoring is needed (Dilo and Zlatanova, 2011; Zlatanova and Fabbri, 2009). A clear information-driven approach to ER is safer and more effective (van der Meer, 2018).

The time spent to reach the location of a disaster from outside the structure is often much shorter than the time spent navigating indoors (Kwan and Lee, 2005). Indoor navigation is generally slower: Rescue forces need to deal with blocked areas, inaccessible spaces and smoke. Therefore routes are continually adapted on the situation and travel distances are increased. On top of that, high rise buildings and

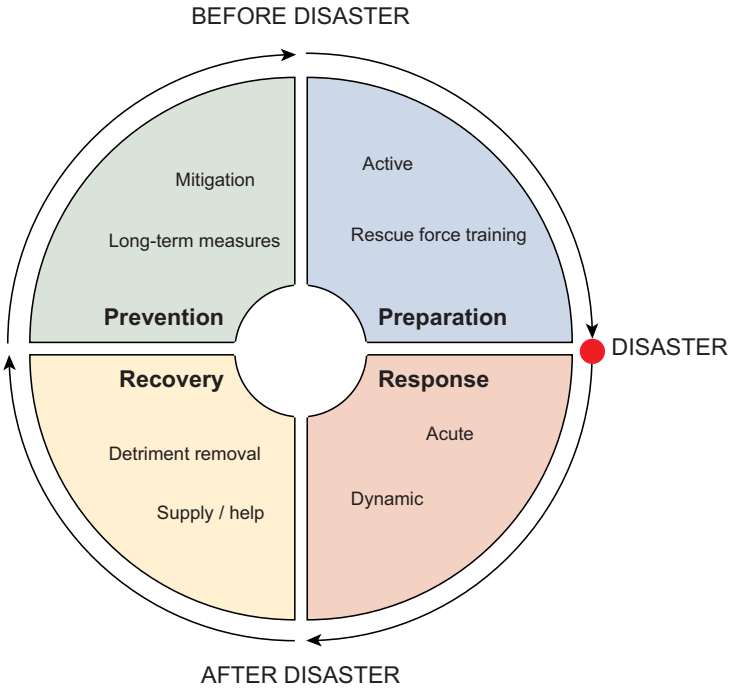


Figure 2.1: The disaster management cycle (adapted from Zlatanova and Fabbri, 2009)

underground structures are increasingly complex. This does not only make navigation harder, it also increases the vulnerability to disasters (Tashakkori et al., 2015).

Until a decade ago, fire fighters were using books instead of digital systems containing floor plans of the ground floor of large buildings in the area they are responsible for. van Capelleveen et al. (2008) predicted that the physical entities will be replaced for digital maps. Currently, not all physical maps are replaced by their digital counterparts (van der Meer, 2018). Indoor positioning datasets are available for public buildings or buildings with higher risks. These are integrated in indoor/outdoor digital environments, such as the Mobiel Operationeel Informatiesysteem (MOI). However, the indoor position on these maps is communicated via radio and speech. Therefore, it remains difficult to keep track of positions of all personnel as a supervisor (Nilsson et al., 2014; van der Meer, 2016).

The need for indoor positioning in ER is high. Many system architectures have been proposed for indoor navigation; indoor/outdoor environments; 3D and 2D indoor visualization in ER situations (See for example Dilo and Zlatanova, 2011; Snoeren et al., 2007; Tashakkori et al., 2015; Zlatanova and Fabbri, 2009; van der Meer, 2018). However, literature on useful indoor positioning techniques in ER are limited. The next section will elaborate on this.

2.2 Indoor positioning

Positioning and localization are often used intertwined. Other notions, such as place, area and pose diffuse this field of research even further. It is necessary to develop an understanding of position in the perspective of indoor environments. The framework of Sithole and Zlatanova (2016) will be used on this matter (the following citations will al be from this research). An illustration of the difference between *position*, *location*, *place*, *area* and *pose* is shown in fig. 2.2. A *position* defines a pin point in space that is set within a reference frame (Sithole and Zlatanova, 2016, p.91). The *position* is absolute, with a specificity that depends on the device that gives the position. A *location* is defined as *"the smallest physically defined space in a building"* (p. 92). *"A place refers to a particular object and the uncertain (functional) space around it"* (p. 92). The *location* is certain, defined by physical borders (e.g. someone is in the study room). The *location* is always with a context (someone is inside of some place that has a meaning). Lastly, an *area*

bounds multiple locations. A *location* is physically defined by boundaries such as walls, while an *area* is unbound by these. A floor of a building is an *area*, with the rooms as distinct *locations*. While Sithole and Zlatanova (2016) do not mention a *pose*, it is still a concept worth mentioning. A *pose* is the combination of a position and an orientation or direction. Indoor positioning is the acquisition of a *position* indoors, while indoor localization is the acquisition of a *location* indoors.

(a) Position: *I am at x,y: (5.2, 6.0)*

(b) Location: *I am in the study room*

(c) Place: *I am at the desk*

(d) Area: *I am on the first floor*

(e) Pose: *I am at x,y (5.2, 6.2), looking in the direction of x,y (0.9, -0.2)*

Figure 2.2: Differences between location; area; place; position; and pose.

Rantakokko et al. (2010) define sixteen user requirements that a positioning system should implement in ER. Only the requirements considered in this research will be discussed here. A positioning system in ER requires:

1. An accuracy in the horizontal plane of <1 meter, so a commander can determine the specific room a person occupies.
2. Constant availability of the positioning data.
3. Estimation of localization errors (uncertainty).
4. Direct deployment; no pre-installation of the positioning system should be needed.

Other requirements, such as an accuracy in the vertical plane of <2m to know which floor is occupied are also necessary, but not in the scope of this thesis. The developed positioning system will be weighted against these four requirements.

There are a few approaches to active indoor positioning. Yang and Shao (2015) categorize these into four groups: Time of Arrival (ToA); Angle of Arrival (AoA); combined ToA/AoA; and Received Signal Strength Indicator (RSSI). ToA, sometimes Time Difference of Arrival (TDoA), relies on a time between a receiver and a transmitter and works very similar to GPS. With at least three anchors, 2D positioning and with at least four anchors, 3D positioning can be performed. AoA, or Direction of Arrival (DoA) approaches use the incoming direction from a transmitter. The direction is computed using phase differences between antennas. Only two anchors are needed, but the algorithm relies on direct line of sight and therefore is vulnerable to multipath errors. Hybrid AoA/ToA use both systems to reduce the number of anchors

nearby and increase accuracy. Lastly RSSI uses the strength of transmitters to identify unique spaces. RSSI is simple and robust compared to the other techniques (Deak et al., 2012). However, it requires a fingerprinting survey in advance and the accuracy will degrade fast in places with a moving crowd. Deak et al. (2012) argue that computer vision techniques are passive positioning techniques, because the person does not carry a device him/herself. However, I argue that this is also an active positioning technique, where a person can have a mobile device that uses these techniques for positioning. Such a system can use Simultaneous Localization and Mapping (SLAM) methods to position in an environment. This will be elaborated on in section 3.1.

Deak et al. (2012) identify five possible positioning technologies: Wireless Local Area Network (WLAN), Ultra-wideband (UWB), field strength systems, Radio Frequency Identification (RFID) and next-generation indoor positioning systems. WLAN can be combined with RSSI and ToA. The advantage of WLAN with RSSI is that the infrastructure of WLAN is usually already in place. Therefore, the installation costs are low. WLAN and ToA can be used together for large indoor spaces, but need to be combined with another technique to retrieve accurate results: the error for a 10 MHz band can rise up to 30 meters. ToA is therefore more effective in combination with UWB, where it can give a resolution up to 30 centimetre (Yang and Shao, 2015). It is affected less by multipath errors. Unlike WLAN systems, UWB has high installation costs. Also RFID systems are usually not present in environments and need to be installed. These systems rely on an electromagnetic communication between readers and tags. The tags have a range of around ten meters. Usually they are combined with a signal strength technique (RSSI).

Next-generation positioning methods make use of a variety of sensors. A system that is proposed by Rantakokko et al. (2011); Nilsson et al. (2014) is based on dual foot-mounted Inertial Navigation System (INS) and inter-agent ranging using UWB. The base of the positioning system are the two INS that are placed on both feet of a person. Using two instead of one INS reduces the drift error that is accumulated. A UWB transceiver is placed on every agent and is used to find the relative position of agents in respect to one another, as opposed to finding a fixed position using static UWB. This data is used to correct for the accumulated drift errors of the agents (Nilsson et al., 2014; Rantakokko et al., 2011). Also AR devices could be placed in the last strand of technologies: next-generation indoor positioning systems. These use a variety of sensors and the SLAM technique for positioning.

System	Technique	Accuracy	Pre-installation
Microsoft RADAR (Bahl and Padmanabhan, 2000)	WLAN + RSSI	2-4m	Fingerprinting survey
Aeroscout (Deak et al., 2012)	WLAN + ToA	1-5m	Tags installation
Ubisense (Deak et al., 2012)	UWB + ToA	15cm-0.3m	UWB installation
BLIP (Deak et al., 2012)	RFID + RSSI	9cm	Beacons installation
Active Bats (Hightower and Borriello, 2001)	RFID + ToA / AoA	10cm-10m	Beacons installation
Inter-agent ranging (Rantakokko et al., 2011)	Mobile UWB + INS	1.4-3.5m	None
SLAM	-	not known	None

Table 2.1: Examples of positioning technologies and techniques.

Examples of various positioning technologies and techniques can be found in table 2.1. It can be seen that the most accurate results can be achieved using UWB or beacons. However, both methods need pre-installation of the method and are therefore not suitable in the case of AR. The solution using mobile UWB and INS is the only solution meeting this requirement, but is not able to achieve an accuracy < 1 meter. To date, positioning using SLAM, without use of any markers that are used as reference points has to our knowledge not yet been researched.

2.3 2D and 3D solutions in Emergency Response

The need for spatial data in ER is clear. Using 3D information is in many ways better than its 2D variant. The shape of buildings is better visible and therefore a better understanding of complex environments is accomplished (van der Meer et al., 2018). Some objects, such as ventilation shafts, are better perceived in 3D. However, the fact is that the work field uses mostly 2D data models. Many buildings lack a 3D model and the existing models usually are the BIM design, that show buildings *as planned* and not *as built* (Zlatanova et al., 2015). Therefore inconsistencies due to renovations, such as lower ceilings or higher floors, occur. Next to that, in a 3D model, some areas block the view on other areas. An advantage of using 2D data is the good understanding and detail of separate building layers (van der Meer et al., 2018). The best practice is to combine a 2D and a 3D visualization, if available. Such a system is found in van der Meer et al. (2018).

2.4 Augmented Reality

AR devices extend reality with placement of virtual objects, or holograms. It is the *"set of set of technologies and devices able to enhance and improve human perception, thus bridging the gap between real and virtual space"* (Manuri and Sanna, 2016, p.18). Augmented Reality (AR) is not a stand-alone concept, but rather exists on a spectrum of mixed reality concepts, with the 'real reality' on one side and Virtual Reality (VR) on the other side (Chatzopoulos et al., 2017). Where VR does not have a physical element, Augmented Reality (AR) blends the virtual and the real. Extended reality is seen as the overarching concept of virtual and real environments. AR is defined by three requirements: (1) it needs to combine virtual and real content; (2) it is interactive in real-time and (3) it is registered in 3D (Manuri and Sanna, 2016, p.18).

Two paradigms exists within AR devices: (1) see-through, head-mounted systems that show the real world as is and overlay that with virtual content. The MH falls under this category, but is not the only device. The Google Glass has received some attention before. Currently, also Magic Leap and Nreal Light are state-of-the-art see-through AR devices (Magic Leap, 2020; nreal, 2020). (2) Display, hand-held systems that show the real world through a video stream in real-time by cameras. Examples are the Google Tango project or apps on a standard smartphone (such as Pokémon GO). The benefit of head-mounted displays is that both hands can be kept free of equipment, unlike hand-held systems. In particular in ER applications, the hands should be kept free from any devices that could also be carried in any other way.

Environmental understanding is inherent to AR applications, simply because it needs to interact with the environment. Recently there has been interest in mapping and positioning as an outcome, instead of a necessary technology for AR to function. Most of the research into this field makes use of the MH, although research is also done with the Kinect technology (such as Klein and Murray, 2007).

2.4.1 Microsoft Hololens

The Microsoft Hololens is a mixed reality head-mounted display system with a variety of sensors. There are two versions on the market, the Hololens 1 and 2. The second version has a better processing unit and unlike the first version it uses neural networks to come to a better spatial awareness system: scanned objects are segmented into walls, ceiling, floors and platforms. In this thesis, the first MH is used, as this was the available device. It has a 32-bit architecture of Intel, with four cores (Microsoft, 2019). A picture of the device can be found in fig. 2.3.

The sensors of the MH include an RGB-camera, a Time of Flight (ToF) depth sensing camera and four greyscale tracking cameras. The ToF camera has a 'long throw' and a 'short throw' mode. The short throw (distances 0 to 0.8m) mode is mainly used for hand gestures, while the long throw (distances

Figure 2.3: Microsoft Hololens (Ramadhanakbr, 2016)

0.8m to about 3.5m) is used for mapping (Hübner et al., 2020). Fig. 2.4 shows the configuration of these cameras. The device also has an Inertial Measurement Unit (IMU).

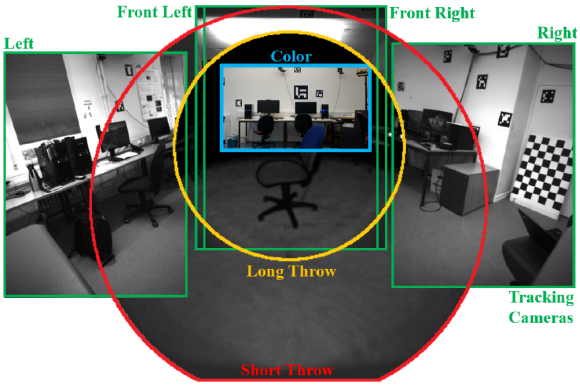


Figure 2.4: Hololens cameras used for tracking and mapping (Hübner et al., 2020)

The MH has been used for marker-based localization by Hübner et al. (2018). By referencing the pose of a marker on a 3D model and in the scan of the device, the position inside the 3D model can be determined. While the process of placing one or multiple markers in the building and referencing these on a floor plan or BIM significantly takes less time than installing any beacons or other devices, it requires a lot of manual work. Therefore is not a suitable method with regard to the use case in this thesis: Emergency Response. Nonetheless, errors between virtual and real world were as little as 3-4 centimeter in a 40m^2 room.

3 Related work

The following chapter goes into detail on the SLAM technology and on various registration techniques that can reference the acquired model to existing data. Various registration techniques will be discussed in section 3.2 and three are elaborated on in sections 3.2.1, 3.2.2 and 3.2.3.

3.1 SLAM

The Simultaneous Localization and Mapping algorithm uses RGB cameras, depth cameras or laser range finders, sometimes enhanced with INS to map a space and position itself inside. Devices that apply SLAM are found in AR, driverless vehicles and robotics. While recent years have seen an uprise in interest into the SLAM algorithms, researchers have been looking into mapping and localization problems by applying estimation-theoretic methods for many years. The problem stems from robotics and automation and has been introduced as early as 1986. However, SLAM as a joint problem of mapping and localization as a single estimation problem was presented at the 1995 International Symposium on Robotics Research (Durrant-Whyte and Bailey, 2006).

A typical system consists of four modules: a front end, back end optimization, loop closure detection and mapping (Li et al., 2019a, p.117). This framework is shown in fig. 3.1. The front end uses the sensor data to provide an initial camera motion value for the back end. The loop closure detection uses image similarity to determine whether the device has already been at that place before. The back end obtains a global optimal estimate based on the input of the front end and the loop closure detection. The output is the mapping.

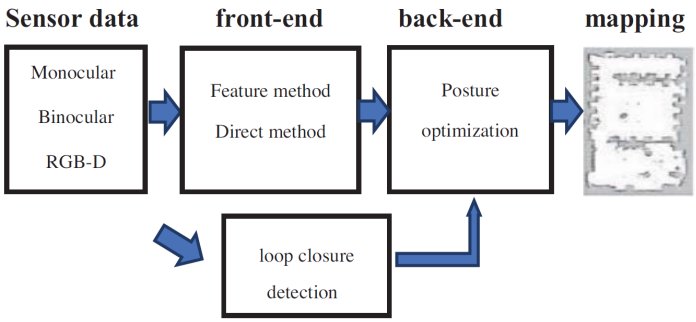


Figure 3.1: SLAM framework (Li et al., 2019a)

The front end can be divided into a direct and an indirect method. The latter is also called the feature point method and is currently most in use. Feature points are extracted from an image, that are matched to obtain pairs of corresponding points. These are used to solve for the camera pose, using motion estimation. The direct method on the other hand, does not extract features, but solves the pose of the camera using (greyscale) pixel values. It solves the pose by minimizing the photometric error. It uses all the information in an image and has a better robustness when features are missing.

3.1.1 SLAM of the Microsoft Hololens

Many SLAM variants exist. As the MH is owned by Microsoft, the exact algorithm could only be guessed. However, localization patents of Microsoft indicate a use of Parallel Tracking and Mapping (PTAM), with

use of key frames (Newcombe et al., 2014). A parallel algorithm divides the tracking and mapping over two separately-scheduled threads. There are some reasons to use parallel system. A robust tracking method can now be used, instead of tracking being a probabilistic output of the map-making procedure. Next to that, many of the frames processed by a regular SLAM algorithm are needed for tracking, but redundant for the mapping. With use of a smaller number of more useful key frames, instead of trying to filter every data frame, time can be saved.

The MH SLAM has been tested on accuracy with varying results. In general, the HoloLens produces a very accurate mesh if only one or a couple of small rooms are mapped. Khoshelham et al. (2019) find most points within 5cm distance of a 3D model. Hübner et al. (2020) find a similar results, with an average accuracy of 4 centimetres with a fixed scale. Landgraf (2018, in Hübner et al., 2019) has shown that drift effects visibly occur in large-scale buildings. The latest research to date found a relative drift error per second of around 1.3cm per second and a 1.5 rotation error per second. The total drift offset found over a course of 287 meters was 2.39 meter (Hübner et al., 2020).

3.2 Spatial Matching

Registration algorithms are used to match a data shape X to another data shape Y. They do so, using geometry features of both data shapes. Spatial matching techniques can be organized in broadly two directions: deterministic and iterative.

Principal Component Analysis (PCA) and Singular Value Decomposition (SVD) are two deterministic algorithms that are based on the covariance matrices of the shapes. PCA aligns the centres of both shapes to the origin. Afterwards, the orientation is corrected using the eigenvalues of the shape itself. Such a method is very easy to implement, but partial registrations are not possible. SVD is very similar, but instead of using the complete source and destination shape, the algorithm uses correspondences between points. The eigenvalues are obtained through a least-squares problem, using all corresponding point pairs (Bellekens et al., 2015).

Iterative spatial matching techniques try to improve on these deterministic algorithms. The most grounded is the ICP algorithm. It retrieves corresponding points in the two data shape and uses a method similar to SVD to minimize the error. This process is iterated until a certain threshold is met. As this method was first explained in 1992 by Besl and McKay (1992), many variants and optimizations exist. Pottmann et al. (2004) use an iterative method based on Local Quadratic Approximation and Instanteneous Kinematics (IK). Where the performance of ICP gets worse in the later iterations of the algorithm, IK does not.

Others use domain properties to find similarities between point sets, such as the Fourier Transform or the Hough Transform, described respectively by Makadia et al. (2006) and Ni et al. (2013). A method that uses one of these transforms tries to find a global transformation and therefore do not need an initial fit. While the implementation is different, both rely on finding the rotational component first using the transform and computing the translational parameter with correlation. The method using Fourier Transform uses ICP after finding the rotation and translation parameters, to come to a more precise transformation. The method using Hough Transform is distinctive from all others, as it uses a 2D map to register a 3D shape to, while other implementations are 3D/3D, with a possibility to reduce dimensions to 2D/2D.

There are other, more complex registration methods existing. Elbaz et al. (2017) constructed a neural network to register a small point cloud to a larger. The method find geometrical structures in the data sets, or 'super-points', that are matched using by correlating descriptors of the super points. The final fit is found by using ICP.

Three of these spatial matching algorithms will be selected as feasible options, to investigate further. A balance should be made between complexity of implementation, algorithm computation speed and the accuracy. A registration algorithm in this research is required to do partial registration, without knowledge of point correspondence pairs. SVD needs point correspondence pairs to be able to register data shapes. PCA is not capable of partial registration, as it takes the overall structure of the data shapes

into account. Both PCA and SVD are not fit for registration in this research. The neural network method of Elbaz et al. (2017) uses a point-to-point registration with similarities between groups of points. The spatial matching in this research should be able to perform point-to-line registration and cannot rely on super-point similarity. Therefore, also that method is not suitable. All the other methods do meet the requirements. ICP is chosen, because of its wide-spread implementations. Therefore, it is a relatively easy method to implement, as there are a lot of examples available. IK is chosen, because it is a method that theoretically should work faster than ICP, however does not have many implementations yet. HT is chosen over Fourier Transform. This has two reasons: firstly, the implementation of HT described by Ni et al. (2013) implements a method using a 3D source model and a 2D reference map. This is close to what is wanted to be achieved by this research. Secondly, because the implementation of the Fourier transform is only a rough transformation. It uses ICP for a closer fit to the data. This confuses the comparison of methods, where one of the methods uses the other. Lastly, implementing a neural network is a too complicated task, with respect to the time frame of this thesis and therefore is deemed unfit.

The selected algorithms ICP, IK and the Hough Transform (HT) are fully elaborated on in the next sections.

3.2.1 Iterative Closest Points

The following registration technique is the oldest and probably most used. It was first constructed by Besl and McKay (1992). Some variant of ICP is also used in the SLAM used in Microsoft AR devices (Newcombe et al., 2014). A rotation parameter R and a translation t are computed so that a data shape X is best aligned with model shape Y. This is achieved in a least-squares manner.

Each iteration, the closest points between X and Y are computed. Afterwards, a registration is computed that minimizes the distance between X and Y. The registration is applied to the model X and lastly, the change in mean-square error between the old and the new iteration is computed. If this change is below a threshold τ, the iteration is terminated.

If the distance between two data shapes can be described as $d(X, Y)$ and the displacement of a data shape can be described by a transformation q applied to a data model, the objective of each iteration can be described as in eq. 3.1.

$$\arg\min_{q} d(q(X), Y)^2 \tag{3.1}$$

First, for every point in data shape X, the closest point to model shape Y is computed. The distance of the closest point in the model shape Y from a point x in point set X can be described as in eq. 3.2.

$$d(x, Y) = \min_{y \in Y} \|y - x\| \tag{3.2}$$

The vector y describes the closest point to x. The transformation vector q can be split into a rotational component $q_R : \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}$ and a translational component $q_T : \begin{bmatrix} q_4, q_5, q_6 \end{bmatrix}$. A displaced point x' can be described as $x' = R(q_R)x - q_T$. The squared distance between y and the displaced point x' should be minimized. The least square function is described as such in eq. 3.3, solving for q.

$$\arg\min_{q} \frac{1}{N_X} \sum_{i=1}^{N_X} \|y_i - R(q_R)x_i - q_T\|^2 \tag{3.3}$$

When the optimal parameters for q are found, the registration is applied to data shape X: $q(X)$. The mean square error between altered data shape $q(X)$ and reference model Y is computed by $d_k = d(X, Y)$. If the improvement of iteration k over iteration k-1 is below threshold τ, the iteration is terminated. This can be seen in equation 3.4.

$$d_{k-1} - d_k < \tau \tag{3.4}$$

Variations of the ICP algorithm

Since Besl and McKay (1992) published the ICP algorithm, many variants were constructed. Rusinkiewicz and Levoy (2001) categorized these into five alteration typologies:

1. Methods to select a subset of points in the registration and/or reference shape
2. Alteration in the procedure selecting the corresponding points
3. The weighting of points
4. Methods to reject point pairs
5. Variations on the minimizing the error metric

A number of sampling methods can speed up the ICP computation. While the method of Besl and McKay (1992) uses all points, also subsets can be used. Some variations exist, such as uniform subsampling, random subsampling, but also more intelligent ways of creating a selection of points: choosing points such that the distribution of normals among the selected points is as large as possible (Rusinkiewicz and Levoy, 2001). Others use a multi-resolution, or pyramid system for a computation cost reduction. The resolution is increased when the fit of the algorithm gets better. The underlying idea is that the precision that can be reached with a full resolution is achieved in the last few iterations, while the iterations before are used to attain a coarse fit (Jost and Hugli, 2003).

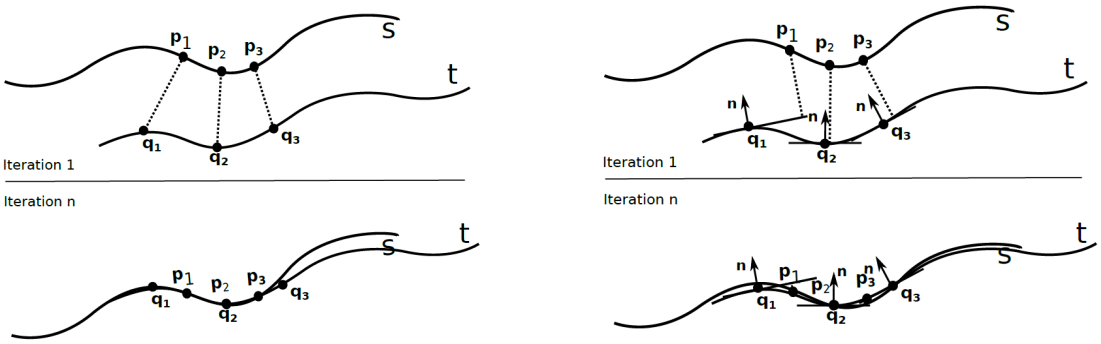
While the matching method is named iterative *closest* points, one could also choose for point pairs that are selected on other variables. Some point clouds contain colour, which can be selected as an extra constraint on the point matching. Matching can also be performed using the point normals, where the closest 'compatible' point is selected, with a normal within 45 degrees.

Points can be weighted with a constant weight, but some methods use different weights per point pair. One could weight on the compatibility of normals, or on the distance between two points. If points are close by, they should be weighted more than if the points are further away.

It is an option to reject pairs on certain properties. The rejection of point pairs can rely on a maximum distance threshold, percentage or standard deviation. Also inconsistencies with neighbouring points can be a ground to reject point pairs.

Besl and McKay (1992) uses a point-to-point error metric: the distance between the corresponding point pairs is used as an input measure for the least squares equation. However, there are variants that use a point-to-plane (or point-to-line) distance metric: instead of using the exact point pair, the distance to the plane containing the reference point is used as a least-square measure (Bellekens et al., 2015). The difference is shown in fig. 3.2. The minimization is be expressed in eq. 3.5.

$$\arg\min_{q} \frac{1}{N_X} \sum_{i=1}^{N_X} \|((R(q_R)x_i + q_T) - y_i)n_i\|^2 \tag{3.5}$$



(a) ICP alignment using point-to-point error metric (b) ICP alginment using point-to-plane error metric

Figure 3.2: A point-to-plane alignment metric could give a better fit than a point-to-point error metric. Figures from Bellekens et al. (2015).

3.2.2 Local quadratic approximants and instantaneous kinematics

The Local Quadratic Approximant and Instantaneous Kinematics method, IK in short, is constructed by Pottmann et al. (2004) because the ICP algorithm had problems. It is a variant on the previous algorithm, in the sense that it also uses the closest points to the point cloud that needs to be registered and it is iterative method. However, it uses a different objective function to minimize. ICP is computed using least squares and works well with points far away from the model that needs to be registered to. However, if the points are close to this model, the performance decreases. They show that using IK, the convergence is faster. The following paragraph paraphrases the method used by Pottmann et al. (2004).

Instantaneous Kinematics

Instantaneous Kinematics is the movement of an object at a particular moment in time. It is widely used in research fields that deal with complex displacements, such as robotics. The idea to use kinematics in point cloud registration, is to view the point cloud as a moving body towards a reference, or destination object. The bodies movement is defined by a velocity vector, that is composed out of a velocity vector at the beginning and an angular velocity. Generally speaking, the first defines a direction that the body moves to and a speed. The vector of angular velocity, also called the *Darboux vector* is the rotational component and therefore defines a rotational direction and power. The velocity vector v(x) can be presented by the following equation, where $\overline{\mathbf{c}}$ is the velocity vector of the origin and $\mathbf{c}$ the Darboux vector, as can be seen in eq. 3.6.

$$v(x_i) = \overline{\mathbf{c}} + \mathbf{c} \times x_i \tag{3.6}$$

The displacement of a point x_i to a location x_i' can be computed by simply adding the velocity vector to the point: $x' = x_i + v(x_i)$. The objective of the registration is to find a position, where point set X is as close to a reference model Y, by moving the point set by some velocity vector $v(x)$. In other words, some combination of a $(\overline{\mathbf{c}}, \mathbf{c})$ should be found to solve following optimization function, where F_i is a function that approximates the distance to a reference model Y. This is described in eq. 3.7.

$$\underset{v(x)}{\arg\min} \sum F_i(x_i + v(x_i)) \tag{3.7}$$

Local quadratic approximation

The distance between a point x_i and a surface or line ϕ can be approximated by a combination of nearest point y_i on the surface/line, a unit normal vector n_i at the location of the nearest point and a distance scalar d_i. It is described in following equation 3.8.

$$x_i = y_i + d_i \mathbf{n_i} \tag{3.8}$$

If a point is displaced by velocity vector $v(x)$, the movement in the direction of the surface can be extracted by taking the dot product of the velocity vector with normal n_i. By adding the previous distance d_i, the squared distance between x_i' and y_i is computed, as shown in eq. 3.9

$$F(x_i') = (d_i + \mathbf{n_i} \cdot (\overline{\mathbf{c}} + \mathbf{c} \times x_i))^2 \tag{3.9}$$

Now the distance functions is defined, minimization function 3.7 can be combined with the distance function. The total squared distance between displaced point set X' and reference model Y is shown in eq. 3.10.

$$F(C) := F(\overline{\mathbf{c}}, \mathbf{c}) = \sum_i (d_i + \mathbf{n_i} \cdot (\overline{\mathbf{c}} + \mathbf{c} \times x_i))^2 \tag{3.10}$$

An optimal $(\overline{\mathbf{c}}, \mathbf{c})$ needs to be found, that minimizes this function F(C). Where d_i is the distance between x_i and y_i and n_i is the normal vector from the tangent plane. Equation 3.11 shows the same function as a matrix function, with some rewriting (see Pottmann et al. (2004) for full explanation).

$$F(C) = D + 2B^T C + C^T A C \tag{3.11}$$

In this equation, D is just some scalar, B is a column vector with six entries, that is constructed by the sum of all $d_i \cdot (x_i \times n_i, n_i)$. A is a six by six matrix, constructed by the sum of all $(x_i \times n_i, n_i)^T (x_i \times n_i, n_i)$. A structure as found in equation 3.11 is part of the group of quadratic functions $ax^2 + 2bx + c$, where the minimum can be computed using the derivative shown in 3.12.

$$AC + B = 0 \tag{3.12}$$

This is a system of linear equations, that can be solved for $\overline{\mathbf{c}}$ and $\mathbf{c}$.

Using Plücker coördinates for a rigid motion

The motion $v(x)$ is a not a Euclidean rigid body motion, but an affine one. In point cloud registration, it is undesirable if the shape of the scanned object has changed. A rigidity constraint on the transformation should solve this problem. C can be approximated by a rotation α about an axis G in space and a translation or pitch ρ, that is parallel to this axis. The axis is constructed using Plücker coördinates $g, \overline{g}$. The rotation, axis and pitch can be retrieved by eq. 3.13

$$g = \frac{\mathbf{c}}{\|\mathbf{c}\|}, \quad \overline{g} = \frac{\overline{\mathbf{c}} - p\mathbf{c}}{\|\mathbf{c}\|}, \quad p = \frac{\mathbf{c} \cdot \overline{\mathbf{c}}}{\mathbf{c}^2}, \quad \alpha = \arctan \|\mathbf{c}\| \tag{3.13}$$

This results in a helical motion, as shown in fig. 3.3. The motion gives a better (quadratic) convergence behaviour than ICP for small residuals. Larger residuals give a convergence that is similar (linear) to ICP. The algorithm should account for small changes each time, therefore the algorithm is expected to work faster than ICP.

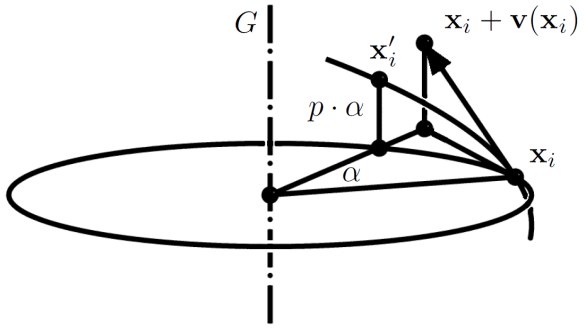


Figure 3.3: The transformation of the velocity vector $v(x)$ to a helical motion of x_i around axis G (Pottmann et al., 2004).

Performance of IK

The IK method is tested on accuracy and convergence behaviour using an industrial product. It is tested twice, once with and once without Gaussian noise. The object that is used in the research of Pottmann et al. (2004) research to register a point cloud has a lot of curvature. This is different from the object in the current study: the surfaces of buildings are usually planar. The tangent plane of planar surfaces is the plane of the surface itself. Therefore, the first step in the algorithm is arguably not performing better in the case of building point cloud registration and could possibly not yield the same results as found in Pottmann et al. (2004). The IK algorithm converges in 12-20 iterations, with an accuracy that is better than the accuracy of ICP after 100 iterations. This is mostly due to the fact that IK supports tangential movement, which helps converging over very small distances.

3.2.3 Hough Transform

The third spatial matching algorithm could be portrayed as the odd-one-out. It does not register a point cloud to another 3D model, but to a 2D map. That makes this algorithm suitable in this research. Moreover, it is not an iterative but a direct method.

The algorithm that is constructed by Ni et al. (2013). It relies on property of hough space that the rotation is invariant to the translation. The transformation is calculated in five steps:

1. Plane finding using Hough Space
2. Finding the up-vector (or correctly orienting the horizontal plane)
3. Finding the correct rotation
4. Finding the correct scale
5. Finding the correct translation

Plane finding in hough space and finding the up-vector

In hough space, the characterizing variables are the offset from the origin, ρ and the angle calculated from the origin, θ for lines and (θ, ω) for planes. The Hough Transform (HT) consequently, transforms lines or planes from euclidean into hough space. Points can be transformed into hough space as well, but as long as it is not known which line or plane they belong to, they will show up as sinusoids in hough space (see fig. 3.4).

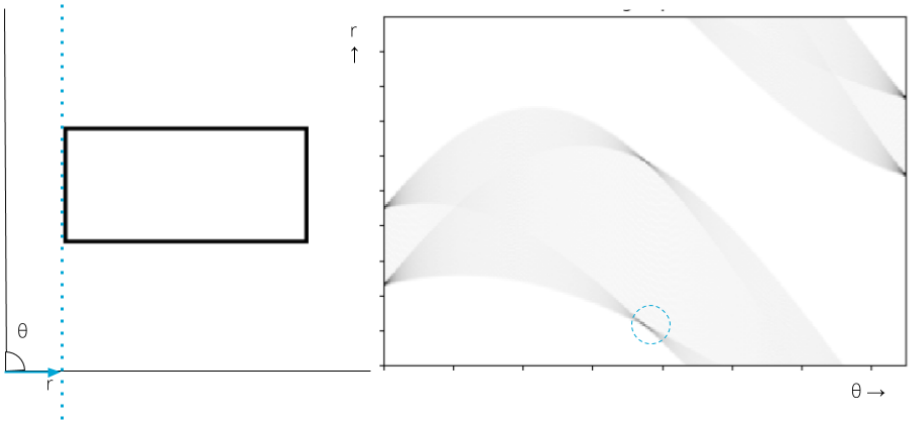


Figure 3.4: A rectangle becomes many sinusoids in Hough space. These sinusoids come together at four points. The x-axis becomes the θ, or rotation and has points at 0, $\frac{1}{2}\pi$ (or 90 degrees) and again at π (or 180 degrees). The r is different for every line. The dotted blue line corresponds to the point inside the dotted blue circle.

There are a number of ways to find out which hough point a euclidean point belongs to. In point clouds, often it is the case that the normals are available. Using the position of the point in euclidean space pt and its unit normal vector n, the ρ and θ can be computed as in eq. 3.14.

$$\theta = \arctan \frac{y_n}{x_n}, \qquad \rho = \mid x_{pt} \cos\theta + y_{pt} \sin\theta \mid \tag{3.14}$$

Plane finding can be done using a simple density kernel.

However, some point clouds do not have the normal vector available. Plane finding then relies on a point to plane distance metric. The point cloud in this thesis stems from a mesh, where the normals can be derived from the faces of the mesh. For further explanation on plane finding without the use of normals, one could look at the research of Ni et al. (2013).

The up-vector can be directly extracted from the found planes using SVD and eigenvectors. The up-vector will correspond to the eigenvector with the least amount of variance. While many walls exist in a building that use a distinctive plane, there are usually a limited amount of floor planes.

Finding the optimal transformation using Hough Space

One of the main reasons to use Hough Transform, is that rotation (the x-axis in hough space) is translation- and scale-invariant. Therefore, the rotation can be found separately from the scale and the translation. The optimal rotation is an optimization problem. Let $H^{(I)}$ be the hough transform of a model. Then $H^{(r)}$ is the hough transform of a reference model and $H^{(q)}$ the one of the model that should be rotated. $H^{(I)}(\theta, \rho)$ is the density of hough points at a certain θ and ρ. The optimization can be solved through sweeping γ by shifting the Hough spectrum of the planes / lines of the point cloud $H^{(q)}$ and observing when it aligns the Hough spectrum of planes of the reference floor plan ($H^{(r)}$. The γ where the result is

highest, is the best alignment and therefore the rotation difference between the source and destination model. The equation is shown in eq. 3.15.

$$\arg\max_{\gamma} \sum_{i} \sum_{\rho_1} H^{(r)}(\theta_i, \rho_1) \sum_{\rho_2} H^{(q)}(\theta_i + \gamma, \rho_2) \tag{3.15}$$

After computing the optimal rotation, the scale should be determined. This is deemed out-of-scope in this thesis, so an exact explanation on this scale-matching can be found in Ni et al. (2013). Lastly, the translation can be found by converting the rotation transformation back to euclidean space and using a conventional method to find the translational parameters. Ni et al. (2013) do this by MATLAB's xcorr, that correlates two discrete time-series.

Performance of Hough Transform

The results of the Hough Transform were compared to the ICP algorithm. The ICP took between 0.23 minutes and 1.36 minutes. The HT took 3.25 minutes to 6.57, more than 10 times slower. Also the accuracy is worse in the case of HT: the RMSE was between 1.4-3.9 meter, while the results of the ICP were 1-1.8 meter. There are multiple reasons why the method is still chosen: the scale factor computed a size that was 12% bigger than the optimal factor. As scale is not a weighing component in this thesis, it could give better results. Next to that, ICP was given several initializations, making the results look better than it actually might be.

3.3 Providing an initial transformation matrix

Most registration algorithms work best with a good initial value provided (Li et al., 2019b; Jost and Hugli, 2003). If the starting point of a registration algorithm is not sufficient, a local optimum could be reached that is not true to the real situation. On top of that, if there are large scale differences between the scanned and reference models, it is hard to find a correct transformation. Buildings typically contain many rooms of similar sizes and configurations. At the start of the process of scanning a building, when only one room, or of a part of a room is mapped, it is very unlikely to impossible that a registration algorithm could find this room without any initial value. This issue is already noted by Fonnet et al. (2017). Therefore, it is necessary to give some initial transformation.

There are multiple ways to reach an initial value. Fonnet et al. (2017) propose to manually put the model (a 3D BIM) by hand gestures. Microsoft provides some handles in the Mixed Reality Development Toolkit to achieve this. It is a relatively simple way, but time consuming way of estimating the initial value.

There is an existing method to detect doors in (voxelized) point clouds. The core principle behind the detection is the characteristic shape of a door and the trajectory of a SLAM device. There are three rules: a door center is in empty space; above the door center there are points and the trajectory goes through the empty space (Nikoohemat, 2016). The door can be identified on the floor plan beforehand, therefore automating the initial registration.

4 Methodology

The goal of this thesis is to construct a method to position indoors using the SLAM of the Microsoft Hololens and a floor plan. This chapter describes the methodology to reach this goal. The research takes a comparative design. It will compare the suitability of three registration algorithms for indoor positioning, ICP, IK and HT. Before the registration can take place, some preprocessing should be performed on the floor plan and the MH. This will be described in section 4.1. Section 4.1.3 explains the rough estimate of the transformation before the registration happens, to ensure a good initial fit. After the initial transformation of the spatial mesh of the MH, it will be converted to a point cloud, in order to be used for point-to-line registration, using the three registration algorithms described in the previous chapter. These will be explained in sections 4.1.1 and 4.1.2. Lastly, the evaluation is elaborated on in section 4.3. An illustration of the methodology can be seen in fig. 4.1.

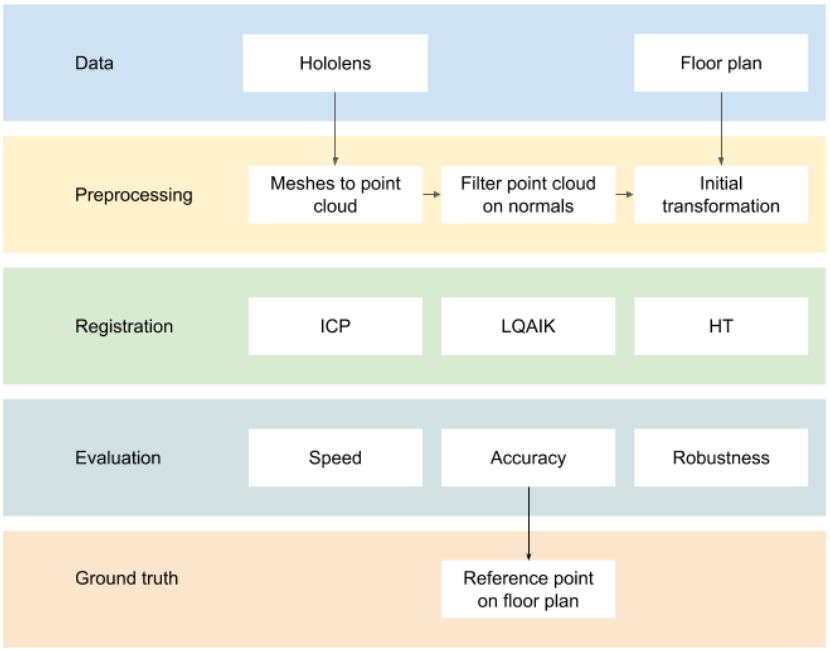


Figure 4.1: Proposed methodology to position in an indoor environment and evaluate the method.

The positioning method is a dual system that is made up of the Hololens data and the SLAM algorithm on the one side and the floor plan with the position on the floor plan on the other side (see fig. 4.2). The initial transformation is already performed at the time of mapping, therefore is omitted here. The walls are extracted from the mesh with use of surface normals (see also section 4.1.1). Afterwards, the transform parameters of the mesh to align with the 2D floor plan are estimated through the spatial matching algorithms. These transform parameters are not used to change the position the mesh or the floor plan itself, but to map the position of the SLAM algorithm to the position on the floor plan. This process repeats every fifteen seconds.

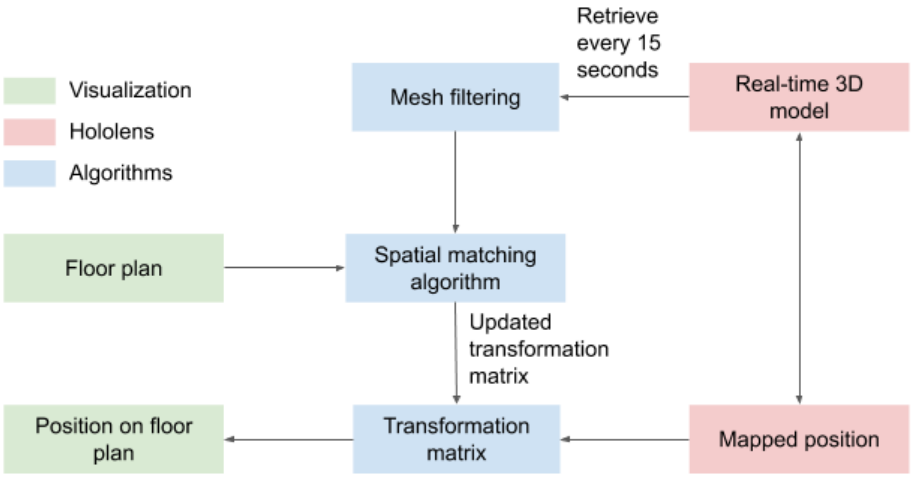


Figure 4.2: Continuous process of indoor positioning using the Hololens.

4.1 Data preprocessing

This section elaborates on the data that needs to be processed. In section 4.1.1, the filtering of the MH acquired mesh and the conversion to a point cloud will be explained. Section 4.1.2 concerns the preprocessing of the floor plan and 4.1.3 describes the initial transformation that is performed before the registration.

4.1.1 Hololens preprocessing

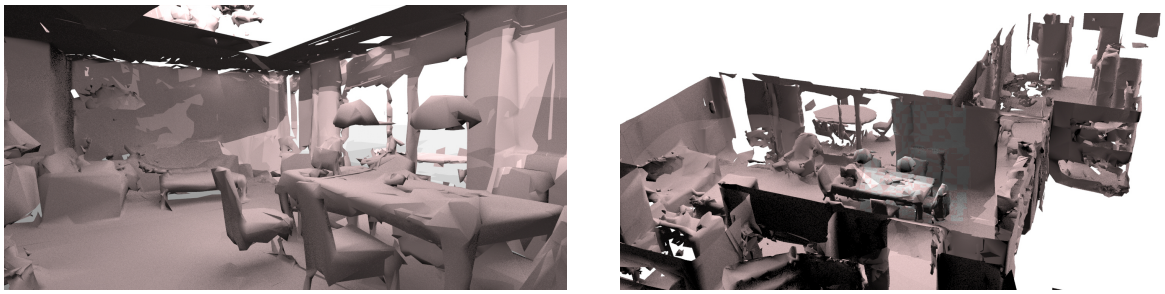


Figure 4.3: Quality of the Microsoft Hololens scanned mesh objects.

The Microsoft Hololens spatial mapping module provides an on-the-fly produced mesh of the environment (see fig. 4.3). All three registration algorithms need a 2D point cloud of only the walls as input. Therefore, the spatial mesh needs to be (1) converted to a point cloud and (2) be filtered on vertical surfaces and (3) reduced with one dimension. Ceilings; floors and other horizontal areas should be filtered as adequate as possible, to reduce noise in the registration. The conversion and filtering can be achieved in one algorithm, based on random points on the mesh faces and the normals of the faces (see algorithm 4.1). The input of the algorithm is the MH spatial mesh and a point density d. All triangle faces of the mesh are checked on their orientation. If the orientation is perpendicular to the 'up-vector' (the orientation of the Z-axis), the triangle is vertical. The area of all vertical triangles is computed and a number of points is calculated, using the point density. A random point inside the triangle is constructed, by (1) a random interpolation between two vertices of the triangle and (2) a random interpolation between this point on the edge and the third vertex. The dimension is reduced by simply omitting the Z-value.

Algorithm 4.1: Mesh to point cloud($\mathcal{M}$, d)

Input: Hololens Mesh $\mathcal{M}$, Density d
Output: pointcloud X with normals N

τ: triangle ;
$i : 0$;
$V_{up} : Vector(0, 0, 1)$;
for $\tau \leftarrow \mathcal{M}$ **do**
    $V \leftarrow \tau$.vertices ;
    $N_\tau \leftarrow (V[0] - V[1]) \times (V[1] - V[2])$; // N_τ: triangle normal
    if $\mid \hat{N}_\tau \cdot V_{up} \mid < 0.1$ **then**
        $A \leftarrow N_\tau \cdot 0.5$; // A: triangle area
        $n\_points \leftarrow A \cdot d$;
        for $j \leftarrow 0$ *to* n_points **do**
            pointonedge $\leftarrow$ RandomInterpolation$(V[0], V[1])$;
            $X_i \leftarrow$ RandomInterpolation(pointonedge, $V[2]$) ; // random point in triangle
            $N_i \leftarrow \hat{N}_\tau$;
            $i \leftarrow i + 1$;
return(X, N)

4.1.2 Floor plan preprocessing

Not only the spatial mesh needs preprocessing, also the floor plan needs to be in the correct format. The floor plan is used to register the point cloud output to, described in the previous section. All algorithms work with a closest-point-to-line mechanism. This mechanism finds the point that is closest to the set of lines of the floor plan, for every point in the point cloud. It is theoretically possible to work with various line types, such as arcs, splines, or other curved lines. However, most walls of buildings are straight, or have a curvature that is so low that it can be transformed into multiple straight lines with different orientations. As the focus point of this thesis is not to have as much support for various types of floor plans, but rather the matching method itself, it is chosen to work with simple lines. All lines used have no more than a start and an end point. There are no splines, arcs or other curves. Walls in a building do have depth, so every wall is represented by a line at at both sides of the wall. Data consists of walls and walls only, so preprocessing needs to be done to remove all other features, like doors; staircases; or windows. These are manually removed by use of CAD-programs. Preprocessing also needs to be done to change all complex lines to simple lines.

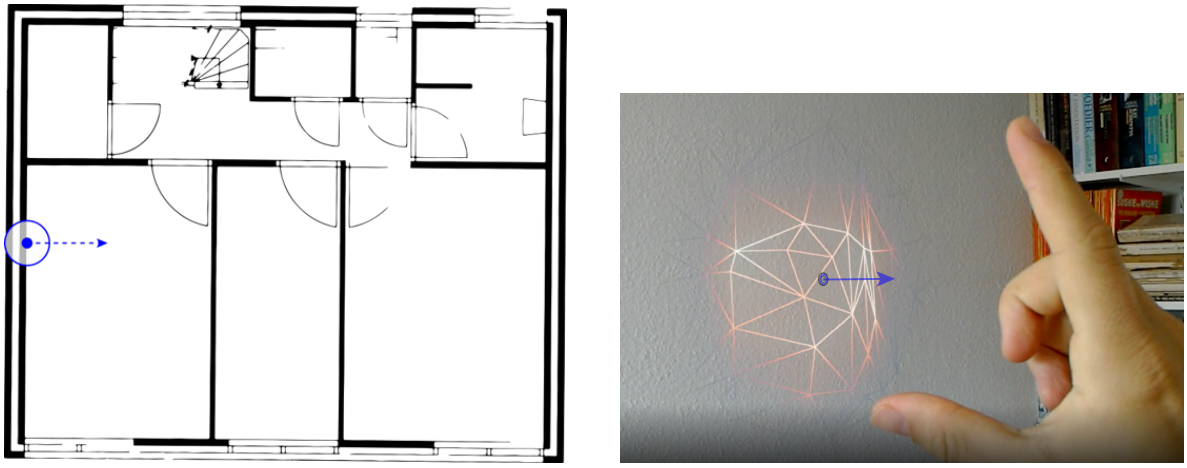
It could be argued that the manual processing of a floor plan contradicts the fact that ER needs to deal with limited time. A manual process would make the whole method unsuitable for the use case. On the other hand, it is only the closest-point-to-line algorithm that is affected by this choice. This particular algorithm could be changed in isolation by a more intelligent algorithm, that can deal with other line types as well.

4.1.3 Initial transformation

An initial transformation value should be provided before the registration process starts. Two methods are elaborated on in section 3.3. One is a manual registration, where someone can rotate and translate the floor plan to the correct place. This process can be very slow, thus is considered infeasible. A second method is proposed, using a door detection for providing a more automatic initial value. A considerable advantage is that someone does not need to be in the building for the manual part of this initial registration. However, there are also a disadvantages to this method: the detection mechanism could fail to detect a door as such, or could detect something else (such as columns in front of a building)

as a door incorrectly. Therefore a new method is constructed, which is faster than the first, but more robust than the second.

The proposed method is similar to the marker-based positioning method described in chapter 2. The method relates an object in the real world with one on the floor plan. Instead of using markers in both real and virtual world, a random position on a wall is used. A point on a wall is selected by a ray-cast. This returns a point on the wall in the coordinate space of the MH and the normal of the wall face. The wall is also selected on the floor plan. The rotation between the normal of the floor plan wall and the wall in MH coordinate space is equal to the initial rotation. The initial translation is equal to the difference between the (x,y) coordinates of the point in MH space and the point on the floor plan. This is illustrated in fig. 4.4



(a) Selected point and point normal on the floor plan

(b) Selected point and point normal on the scanned mesh

Figure 4.4: Using the position and orientation of a wall existing in the floor plan and the scanned object, the position on the floor plan can be estimated

4.2 Registration

Shape registration is a technique to align one shape to another. Three variables play a role: translation, rotation and scale. Scale is considered out-of-scope in this research, but the first two variables need to be computed in order to register a shape to another. The goal of a registration algorithm in this case is to find the best transformation matrix, consisting of a rotation matrix and a translation vector. The rotation is a 3x3 matrix that is defined by a rotation around the X-axis (α), one around the Y-axis (β) and one around the Z-axis (α). The translation is a three-dimensional vector. Both are shown in equation 4.1.

$$R = \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix}, T = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{4.1}$$

However, both the translation and rotation are used in shape registration in 3D space. As a 2D registration is performed instead of a 3D, both components can be reduced. The registration in 2D space only rotates around the Z-axis in 3D space. The translation does not have a z-component. The rotation matrix and translation vector used can be found in eq. 4.2. While the registration is 2D, the environment of the method will maintain a Z-component, as this will be convenient for visualization purposes and

projections on the MH.

$$R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, T = \begin{bmatrix} x \\ y \\ 0 \end{bmatrix} \tag{4.2}$$

The final product of registration is a transformation matrix. This is a 4x4 matrix with the rotation and the translation component combined. The last row in the transformation matrix is reserved for non-rigid transformations. An example of a non-rigid transformation is scale, but also shearing or changing perspectives are non-rigid transformations. These do not have to be dealt with and therefore the last row is a 4x1 matrix $\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$. The transformation matrix is shown in eq. 4.3.

$$M = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \tag{4.3}$$

4.2.1 2D registration

Shape registration mostly concerns 3D space. The ICP and IK algorithms both use 3D space. HT first correctly aligns the rotation in X/Y and subsequently uses a 2D technique to find the transformation parameters. The choice of using 2D registration in this research does not rely on the performance of 2D or 3D algorithms, but it relies on the availability of floor plans, as opposed to limited access to BIM. However, it has impact on the registration. A loss of dimension also means a loss of data.

I argue that the Z-component is not needed for a successful registration. First of all, the MH has the capability to correctly align its orientation to the ground plane. While in other use cases, such as the registration of mechanical parts in Pottmann et al. (2004), the point cloud needs to rotate around the X- and/or Y-axis, this is not needed with the spatial mesh of the MH. Secondly, (most) walls in buildings are perpendicular to the floor. Therefore, it is relatively easy to extract the walls from the 3D mesh. Finally, a floor plan is a section of a building at 1 meter above the floor. It shows the navigable space for a human being. A BIM shows more redundant information, especially considering the fact that the MH does not have a reach of more that 3 meters, so the high ceilings that large buildings often have are not mapped anyway.

There are some disadvantages to 2D registration. Most importantly, complex buildings with many non-vertical walls will be difficult or even impossible to register correctly. However, the methods used in this research all can be extended to 3D variants without changing the whole structure of the method. It is not in the scope of the thesis to do so.

In chapter 2 it is argued that a 3D visualization could enhance a 2D visualization in the case of ER. 2D/3D visualization is not possible if no BIM is present. However, it becomes possible with use of the scanned 3D model of the MH. This method makes a 2D/3D visualization possible without prior 3D knowledge of a building. Using a 2D registration method does not lose the 3D component of the MH mesh.

4.2.2 ICP

The foundation of the ICP algorithm used in this thesis comes from Besl and McKay (1992). However, in the last thirty years, many variants have been published that increase the speed or robustness of the method. These have been explained in chapter 3. Six types of variants have been identified, based on the stages it affects. These are: (1) selection of points, (2) matching of points, (3) weighting of points, (4) rejecting points, (5) assigning an error metric and (6) minimizing the error metric.

The implementation of Birdal (2020) is used as reference. It combines geometrically stable sampling (Gelfand et al., 2003), picky ICP (Zinsser et al., 2003), linearisation of the point-to-plane error metric

Algorithm 4.2: Iterative Closest Points

Input: point cloud X, floor plan lines L, maximum iterations $maxIter$, motion threshold τ, hierarchy levels h_max
Output: transformation matrix M

M : 4x4 identity matrix ;
$\hat{X} \leftarrow$normalize(X) ;
$\hat{L} \leftarrow$normalize(L) ;
$N \leftarrow$ length X ;
for $h \leftarrow h_max$ *to 0* **do**
  $N_h \leftarrow \frac{N}{2^h}$;
  $maxIter_h \leftarrow \frac{maxIter}{2^h}$;
  $\tau_h \leftarrow \tau * (h+1)^2$;
  $X_{moved} \leftarrow M \times \hat{X}$;
  `// Set motion parameters`
  $fval_{old} \leftarrow \infty$;
  for $i \leftarrow 0$ *to* $maxIter_h$ **do**
    $X_{subset} \subset X_{moved}$; `// random subset of N points`
    $Y, N_Y \leftarrow$ ClosestPoints$(X_N, \hat{L})$;
    $X_{subset}, Y_{subset}, N_{Ysubset} \leftarrow$ reduceDuplicates(Y, N_Y, X_{subset}) ;
    foreach X_i, Y_i, N_i *in* $X_{subset}, Y_{subset}, N_{Ysubset}$ **do**
      $B \supset N_i \cdot (Y_i - X_i)$;
      $A \supset \begin{bmatrix} X_i \times N_i & N_i \end{bmatrix}$;
    $\begin{bmatrix} angles & T \end{bmatrix} \leftarrow$ solve$(AC + B)$;
    $R \leftarrow$ Quaternion$(angles)$;
    $M_h \leftarrow$Matrix(R, T) ;
    $X_{moved} \leftarrow M_h \times \hat{X}$;
    `// Compute motion change`
    $fval \leftarrow \sum \|(X_{moved} - \hat{X})\|^2$;
    $fval = \sqrt{fval / N_h}$;
    $fval_{perc} \leftarrow fval / fval_{old}$;
    $fval_{old} \leftarrow fval$;
    if $\neg\,(fval_{perc} < (1 + \tau_h) \wedge fval > (1 - \tau_h))$ **then**
      break
  $M \leftarrow M_h \times M$
`// correct for normalization scale`
$T \leftarrow M$.Quaternion$*scale^{-1} + meanAvg - M$.Quaternion$*meanAvg$;
M.Translation$\leftarrow T$;
$returnM$;

(Low, 2004) and multi-resolution registration (Jost and Hugli, 2003). These are respectively in the stages 1 (selection), 4 (rejection), 5 (assigning error metric) and 6 (minimizing error metric).

The main objective of picky ICP is to improve robustness and speed. Corresponding point pairs are computed between the point cloud and the floor plan. Using the standard deviation of the distances, points can be rejected on a distance bigger than a multiple of this standard deviation. Points can also be rejected, if the destination point (the point on the floor plan) is presented more than once. All source (point cloud) points are rejected in this case, except for the point with the lowest distance. These two additions try to improve the robustness of the algorithm. It is a valuable addition to the standard ICP, because it tries to discriminate good pairs from outliers. A mapped building consists of a lot of noise, compared to the floor plan that is completely void of furniture, plants, humans or other objects. Lastly, to improve performance, the iterations are divided into hierarchy levels (h), where only the 2^h-th data point is used in the iteration. If a convergence is reached in one hierarchy level, it is continued to the next (h-1). This will speed up the computation time, especially for large data sets. A standard threshold of the registration error cannot be used as a metric to stop the registration. Instead, a change of motion parameter should be used per hierarchy level.

Low (2004) uses linear instead of non-linear optimization to solve problems more efficiently. As a requirement, the relative orientation or rotation between the input and reference object should be small. In this use case the condition is met, because an initial transformation is already given (see section 4.1). A linear optimization gives a faster convergence and therefore speeds up the process. It requires the normal of the reference line. It solves the linear least squares equation: $Ax + b = 0$. A is a nx6 Matrix, consisting of (1) the cross product of the source point and the normal vector of the line and (2) the normal vector of the line. b is a nx6 matrix of the dot product of the change vector between the source point and destination line and the normal vector of the line. This is shown in eq. 4.4.

$$A_i = \begin{bmatrix} x_i \times n_i & n_i \end{bmatrix}, \quad b_i = n_i \cdot (x_i - y_i) \tag{4.4}$$

4.2.3 IK

The Local Quadratic Approximation and Instanteneous Kinematics (LQAIK), or IK algorithm is in some way very similar to ICP. However, instead of directly using the distance of a source point to the closest point on a reference surface or line, it uses a local quadratic approximant F_i of the squared distance function to a surface or space curve. This could be the squared distance function of the tangent plane or line. This is relatively easy to compute for simple lines: it is the length of the unit normal vector of the line times the distance to the source point.

The output of the algorithm as explained by Pottmann et al. (2004) is not a translation and a rotation around the Z-axis, but a rotation around an axis somewhere in 3D space and a pitch (similar to a translation). Because the Z-value is omitted, this pitch will always be 0, so it can be excluded. The rotation axis however, is the Z-axis in some (x,y) point. Eq. 4.5 shows how to translate this into a transformation matrix.

$$M = \begin{bmatrix} R & -axisOrigin \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} i & axisOrigin \\ 0 & 1 \end{bmatrix} \tag{4.5}$$

The algorithm that is used to compute the velocity vector is shown in alg 4.4. The point cloud, closest points on the floor plan and normals of these closest points are the input of the algorithm. The equation AC + B needs to solved, where A is a 6x6 matrix and B is a 1x6 vector in the method described by Pottmann et al. (2004). C is also a 1x6 vector, with the first values being the Darboux vector and the last three the velocity vector at origin. However, a 2D registration is performed. Therefore, the darboux

Algorithm 4.3: Local Quadratic approximants and Instantaneous Kinematics

Input: point cloud X, floor plan lines L, maximum iterations $maxIter$, threshold τ
Output: transformation matrix M

M: 4x4 identity matrix ;
ϵ: Error between point cloud and floor plan
for $i \leftarrow 0$ *to* $maxIter$ **do**
    $Y, N_Y \leftarrow$ ClosestPoints(X, L) ;
    `// Compute the velocity vector`
    $c, \overline{c} \leftarrow VelocityVector(X, Y, N_Y)$;
    `// Solve constraint with plücker coordinates`
    $axis, \omega \leftarrow$ Plücker$(c, \overline{c})$
    `// construct transformation matrix`
    $rotation =$Quaternion(ω);
    $M_i \leftarrow [rotation - axis] \times [identityaxis]$;
    if $Betterfit(X \times M_i, Y)$ **then**
        $M \leftarrow M_i \times M$;
        $X \leftarrow M_i \times X$;
    if $\epsilon < \tau$ **then**
        Break;
return M;

vector is only 1 rotational value and the velocity vector of the origin are 2 (x and y). A can be a matrix of 3x3 and B can be a vector of 1x3.

Algorithm 4.4: Find velocity vector

$c, \overline{c} \leftarrow C$;
$A \leftarrow$ zeroMatrix(3x3);
$B \leftarrow$ zeroMatrix(1,3);
foreach $X_i \leftarrow X$ **do**
    $A_i \leftarrow X_i \times N_y i$;
    $A \leftarrow A + A_i^T A_i$;
    $B \leftarrow B + A_i \cdot \|X_i - Y_i\|$;
$C \leftarrow$solve$(AC + B = 0)$;
return X;

A velocity vector transforms a point cloud with a non-rigid motion. The shape of the point cloud should stay the same, therefore the transformation should be a rigid one. The rigidity constraint is solved using plücker coördinates $g, \overline{g}$. In algorithm 4.5, it is shown how to get these coördinates. Using a cross product, an axis G can be found. In the case of 3D point cloud registration, the axis can be in any direction and the pitch p that is parallel to this axis can be a value other than zero. However, since we are dealing with a 2D point cloud, the axis direction will always point in the direction orthogonal to the horizontal plane and pitch p will be zero. The transformation matrix can be constructed using a rotation of ω degrees around the axis G.

Algorithm 4.5: Find plücker coordinates

$\omega \leftarrow \|c\|$;
$p \leftarrow \frac{c \cdot \bar{c}}{\|c\|^2}$;
$g \leftarrow \frac{c}{\omega}$;
$\overline{g} \leftarrow \frac{\bar{c} - pc}{\omega}$;
$G \leftarrow g \times \overline{g}$;

4.2.4 Hough Transform

Hough Transform is the only one of the three algorithms that is in principle non-iterative. It is also already accustomed to the registration of a 3D point cloud to a 2D reference map. It computes its correct position in four steps: (1) finding the 'up-vector'; (2) rotation; (3) scale; (4) translation. Step one tries to correctly align the ground plane of the point cloud to the horizontal (x,y)-plane. However, this step can be omitted, since the MH already finds the correct orientation of the ground plane itself. Step three can be omitted as well, as the difference in scale is considered out of scope in this project. Therefore, only a rotation around the z-axis and a translation should be found.

The rotation is found by transforming the floor plan lines and the point cloud into hough space. Ni et al. (2013) proposes a density kernel to achieve the hough transform. All points of the point cloud can be converted to a sinus-line in Hough Space. Using a density kernel, basins of attraction could be found. These basins of attraction are points in hough space, but relate to lines in euclidean space. The point cloud that is used in this thesis comes from a mesh and therefore normals can be provided that are orthogonal to the mesh triangle a point stems from. A point with a normal could be transformed to a point in hough space directly, as it represents a plane (or line in 2D). This is achieved as following, where θ is on the x-axis and represents the angle and ρ is on the y-axis and represents the distance between the line and the origin. These are shown in eq. 4.6.

$$\theta = tan^{-1}\frac{n_{\cdot y}}{n_{\cdot x}}, \quad \rho = X_i \odot \begin{bmatrix}\cos\theta \\ \sin\theta\end{bmatrix} \tag{4.6}$$

Further computation of the rotation is identical to the method described in section 3.2.3.

Ni et al. (2013) does not elaborate on the translational value, except for the use of MATLAB's xcorr, that makes use of similarity between a vector and a shifted copy of that vector. A similar idea that makes use of histograms is used in this thesis and compared to a simple ICP algorithm that only accounts for translation. The first method divides both the lines and the points into n bands in X and n bands in Y. It tries to compute the best alignment separately for X and for Y, in a similar fashion that the optimal rotation is found by aligning θ. The translation-only ICP finds the closest points on the floor plan for the point cloud and simply computes the average x and y difference between source and destination points. It iterates this for a number of times, or stops if an error threshold is met. An illustration of the translational ICP can be found in algorithm 8. Both methods will be tried and compared.

4.3 Evaluation criteria

The question should be answered what a well-working indoor positioning method should do. To rely on a position indoors, a few requirements should be met. The registrations should be:

- Accurate up to <1m
- Computation time ¡15 sec and scaleable
- Robust

Algorithm 4.6: Hough Transform

Input: point cloud X, point cloud normals N_X, floor plan lines L, floor plan normals N_L
number of bands n_b, max rotation ω_{max}, max iteration $maxIter$, threshold τ
Output: transformation matrix M

$H_L \leftarrow$ toHoughSpace(L);
$H_X \leftarrow$ toHoughSpace(X, N) ;
$N \leftarrow$ length(X)

`// Compute the rotation`
bandwidth = $2\pi / n_b$;
for $\omega \leftarrow -\omega_{max}$ *to* ω_{max} *stepsize* ω_{max}/n_b **do**
  alignment $\leftarrow 0$;
  for $b \leftarrow 0$ *to* 2π *stepsize bandwidth* **do**
    $b_\omega \leftarrow \mod_{2\pi}(b+\omega)$;
    $H_b \subset H_X$ where $b \leq \theta_{point} \leq b+$bandwidth;
    $H_{reference} \subset H_X$ where $b \leq \theta_{line} \leq b+$bandwidth;
    $P_b \leftarrow countH_b$;
    $P_{reference} \leftarrow \sum$ line length $H_reference$;
    alignment $\leftarrow$ alignment $+P_b * P_{reference}$;
  if $alignment > alignment_{max}$ **then**
    $\omega_{max} \leftarrow \omega$;
    $alignment_{max} \leftarrow alignment$
$R \leftarrow$Quaternion(ω);
$\hat{X} \leftarrow X \times R$

`// Compute translation by histograms or by simple ICP`
$T \leftarrow$ histograms($X, L, nBins$) ;
$T \leftarrow$ ICPTranslation(X, L) ;
$M \leftarrow \begin{bmatrix} R & T \end{bmatrix}$;

Algorithm 4.7: Translational ICP

for $it \leftarrow 0$ *to* $maxIter$ **do**
  $Y \leftarrow$ ClosestPoints($\hat{X}, L$);
  $T_{it} \leftarrow \frac{\sum X_i - Y_i}{N}$;
  $\hat{X} \leftarrow \hat{X} + T_{it}$;
  $T \leftarrow T + T_{it}$;
  if $error(\hat{X}, L) < \tau$ **then**
    Break;
return T

The following sections will elaborate on why these requirements should be met and how they are evaluated. Next to that, the positioning method should give some measure of reliability. An estimation of the positioning error, or uncertainty should be found.

4.3.1 Accuracy

Accuracy is one of the main bottlenecks in indoor positioning. Obstructions, such as walls and furniture, give multipath errors and make it impossible to maintain Line-of-Sight. Systems using existing WLAN infrastructure give results that have an accuracy of around 10-30 meters. Using more expensive variants, with UWB or infrared beacons a positioning of up to 9cm can be achieved (Deak et al., 2012). The more suitable system for ER, such as the combination of INS and inter-agent ranging can achieve an accuracy of around 1.5-3 meter on a path of almost 200 meter (Rantakokko et al., 2011; Nilsson et al., 2014). In this thesis it is tried to achieve a lasting accuracy of at least 1 meter, like proposed by Rantakokko et al. (2010). An accuracy of < 1 meter gives a reliable estimate of where someone is, as the chance that someone is positioned in the incorrect room is very low with such an accuracy.

The accuracy is tested on the basis of the ground truth markers explained in section 5.1. There are three variables that influence this accuracy: the accuracy of the MH SLAM itself, the fit of the initial transformation and the effectiveness of the spatial matching algorithm. All need to be tested separately. For every experiment, the spatial matching algorithm needs to be compared to (1) the best possible fit of the MH SLAM, computed after the experiment and (2) the fit of the MH SLAM when only doing the initial transformation, without spatial matching.

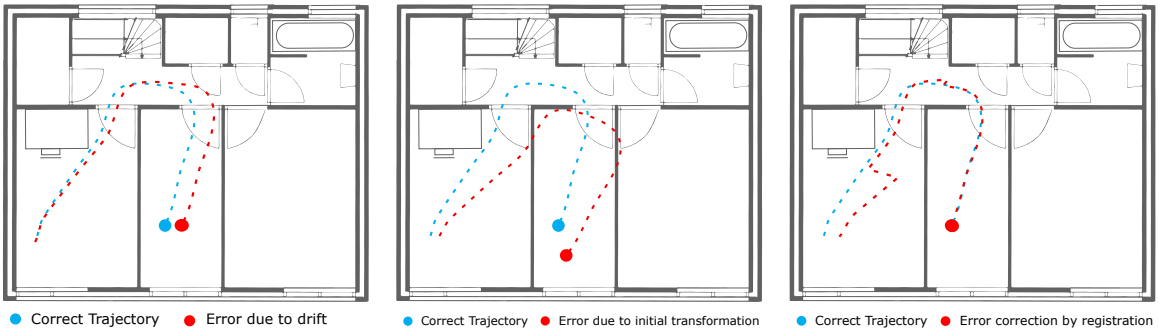


Figure 4.5: Three variables influencing indoor positioning accuracy that should be separated. (1) Errors due to drift, (2) errors due to an incorrect orientation or position at start of the trajectory and (3) the effectiveness of the registration to correct these errors. The errors in the illustrations are exaggerated.

4.3.2 Computation time

Time is a fuzzy concept and therefore the most difficult to evaluate. Time is critical in ER and therefore a 'fast' algorithm is needed. What exactly is fast? It depends on many variables. If the orientation and position of the initial transformation are near to perfect and the SLAM of the MH is not drifting at all, one might want to perform the correction four or five times during the whole operation. If, on the other hand, the orientation is off by a few degrees and the SLAM drifts a meter on every walked ten meters, a correction every five or six seconds is needed.

The other side of the equation is the time that the algorithms actually take. This is dependent on other variables too. Not only the efficiency of the algorithm, but also the power of the processors and the scaleability of the algorithm affect the speed. The MH 1 has less capabilities than the MH 2, which will increase the computation times. The scaleability is about the density of points and the total area that is used in the registration. The density of points will affect the closest point-to-line algorithms, but also the time that it takes to transform an array. The area will only affect the closest point-to-line algorithms, as there are more lines on the floor plan that should be checked. Some scalability tests will be performed to

find out how well the algorithms scale. The algorithms will be tested on 500 points, 2.000 points, 10.000 points and 50.000 points, with varying area sizes of the mesh and floor plan. Using a larger area on the floor plan and a higher point density will result in a computationally more difficult closest-point-to-line algorithm.

Consequently, it is difficult to say something about an absolute time that the method takes. To make an estimated guess, the algorithms should take no more than fifteen seconds. The algorithm is computed once every fifteen seconds and therefore no parallel processes have to run. A better evaluation between the algorithm can be achieved by using the relative time the algorithms take. If all other variables (density of points, processor etc.) are identical, the scaleability and general efficiency of the algorithms could be compared.

4.3.3 Robustness

The MH maps a building how it is in present time, not how it is designed. A floor plan shows a building design, not the present situation. The spatial mesh of the MH contains noise, such as other people, furniture, open doors and more. A positioning method should take all these variables into account and even then give a good estimate of where someone is. There are also degenerate cases, where the algorithms could fail or have difficulties. A list of degenerate cases to be researched is as follows:

- Doors
- Walls that do not exist in the floor plan
- Walls that exist in the floor plan, but not in the present situation
- Large spaces (>20x20 meter halls)

These cases are visually examined. While it is also interesting to research the behaviour on multiple floors, this case is excluded due to time limitations.

4.3.4 On-the-fly reliability

A reliable system not only should be accurate and robust, but should also give a probability that it's position estimate is correct. The system becomes more reliable when it can inform the carrier that it is not reliable. A Mean Square Error (MSE) of the point cloud to floor plan registration is proposed as a metric to evaluate the accuracy in real-time. A high MSE means the position is probably not correct, while a low MSE would mean it is probably correct. A high MSE could indicate a few situations:

- The registration did not happen successfully
- The spatial mesh of the MH contains a lot of noise
- The MH detects walls that are not present in the floor plan

In all situations the reliability of the positioning method will get worse. Therefore it is hypothesized that MSE could be a could metric to measure reliability. It is important to find out what the relation is between the MSE value and the probability to be in the right position. Half of the experiments could be used to train the value of the MSE and to show its correlation to the accuracy. In section 4.3.1, an accuracy of one meter or better is found appropriate. A logistic regression should determine the probability of the accuracy being better than one meter, with the dependent variable being the MSE. The other half of the experiments could be used to test if this works correctly.

5 Case study

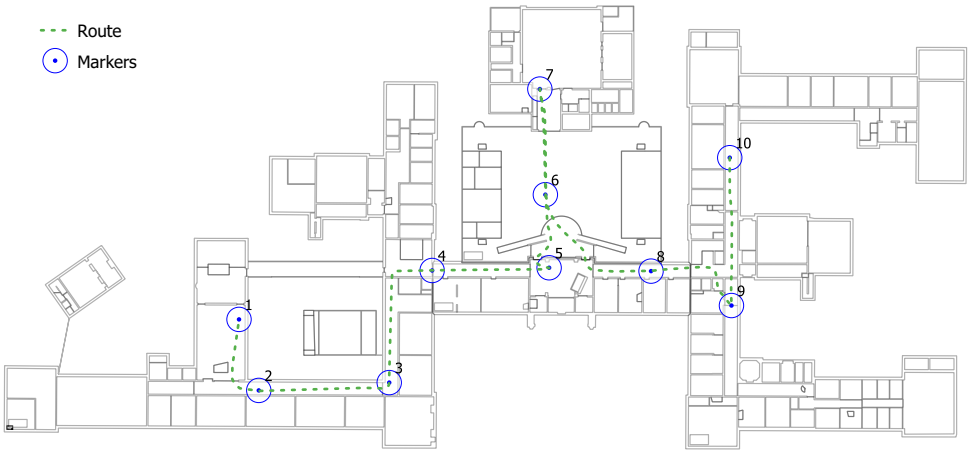


Figure 5.1: The trajectory of the walked route and the position of the ground truth markers on the floor plan of the Architecture building

The architecture building of Delft University of Technology, The Netherlands is used as a case study. The building is chosen because of its dimensions: it is more than 230 meters long and around 100 meters wide. A large building is needed, as Hübner et al. (2020) found a total drift of only 2.5 meter on a trajectory of 200 meters.

An AutoCAD file is provided by the technical information management team (see fig. 5.1). The file contains a lot of (redundant) information, such as supporting walls and columns; lightning; electricity; windows; doors. These are all separated in layers. As all walls were together in one layer, it was relatively easy to extract that information. In QGIS, these lines were converted to simple lines with only a start- and an endpoint and saved to a text-file. This was used as the input for the implementation.

Two buildings are selected as validation cases: the TU library building in Delft and the CGI office in Rotterdam. These buildings are vastly different in their building styles. The hallways in the Architecture building are wide and straight. The TU Delft library's ground floor mainly consists of one big open space, surrounded by glass walls. There is a long hallway on the east side (the lower part of fig. 5.2), that is narrow compared to the hallways in the Architecture building. Lastly, the office building is smaller in size than the former two buildings. There are no isolated hallways that function as circulation space, as could be found in the Architecture building. The movement through the building is the office space

itself. Therefore, when walking through the building one has to deal with more obstructions, such as office desks, counters and cabinets. The floor plans of both buildings are processed in a similar fashion to the Architecture building.

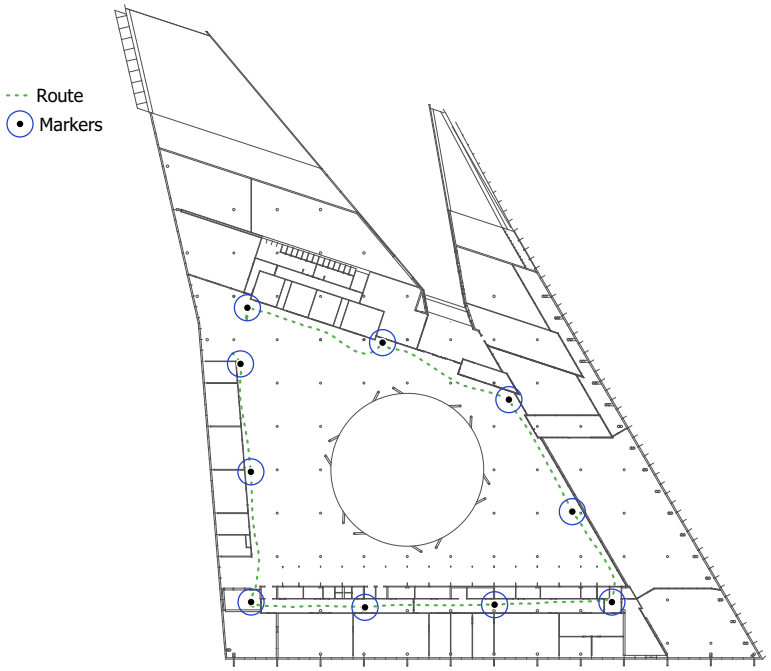


Figure 5.2: The trajectory of the walked route and the position of the ground truth markers on the floor plan of the library building

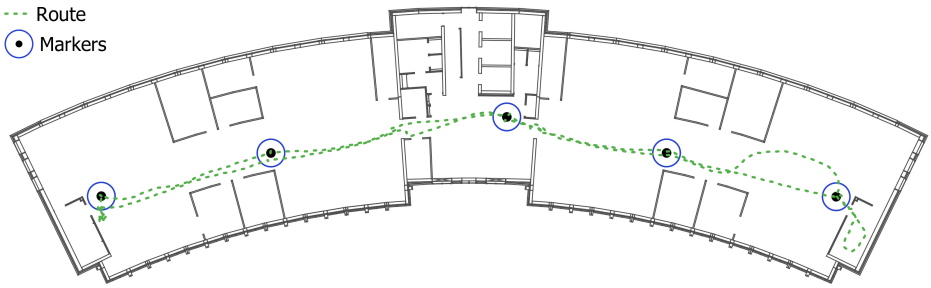


Figure 5.3: The trajectory of the walked route and the position of the ground truth markers on the floor plan of the CGI office building. The route goes back and forth on different floors.

5.1 Experiment design

A route will be set up through one floor of the building. At roughly equal distances, ten markers are placed on the floor. The positions of these markers are measured and placed on the floor plan.

The same route is walked five times, to ensure some generality in the research. At the start of mapping the route, all holograms (including the spatial mesh information) are removed from the MH to ensure a clean slate. An initial transformation is given in-situ, explained in section 4.1.3. At every marker, the position of the device at that moment is saved. While the registration is meant to be real-time, it is chosen to simulate the registration after acquiring the data. This is considering the validity of the research: if all registration will be computed in-situ, differences in quality of the acquired data will occur. Saving

all data and simulating the mapping later will make possible that all registration algorithms can be compared on exactly the same data. All data is saved with a time stamp, to make the playback possible. The spatial meshes are saved every time there is an update to these. The position of the MH is saved every 0.3 seconds and the position of the MH when standing on a marker is saved separately.

In fig. 5.1 the trajectory is shown. Masking tape is used to make crosses, that function as the markers (the blue circles) on the route. At every marker, the ground truth is measured by referencing the position on the floor plan. The center of the crosses is measured using defining elements on the floor plan, such as corners, or the edge of a door. Three experiments start at number 1 and ended at number 10, and two experiments are in opposite direction. The total trajectory of the walked route is 294 meters, although in reality the distance is longer. After the data acquisition, the markers are referenced using the AutoCAD file. The trajectory of the validation buildings are shown in figs 5.2 and 5.3

Some degenerate cases can be identified within the floor plan:

- There are multiple walls on the floor plan that are not present in reality, mostly because these walls are actually big doors used for fire protection. This is for example the case at marker 3 and 4. Between 8/9 the route also crosses some walls, that are not there.
- At marker 5, the main entrance, there is a round object or wall in the middle of the space. It is not present in the floor plan.
- Marker 6 is in a big open space without walls of approximately 30x30 meter. The hall is not void of any objects, it is filled with tables used for model making.

In order to make generalizations about the proposed method, a different building should be selected, where the method could be tested. Unfortunately, due to Covid-19, most large buildings are closed or only open for certain workers. Therefore, no other building could be used.

6 Implementation

This chapter describes the implementation of the method. An application is created with a main purpose of real-time indoor positioning. It can perform an initial transformation of the MH mesh to align with a floor plan, as described in section 4.1.3. This floor plan is visualized as a hologram somewhere in the space, with the position of the device carrier on the floor plan (see fig. 6.1). It can perform the registrations between the floor plan and the MH meshes, to compute a more accurate position. Next to this real-time system, the application is also able to simulate previous scans on the computer. In that case, the registrations are not computed in real-time during the scan, but afterwards in the simulation.

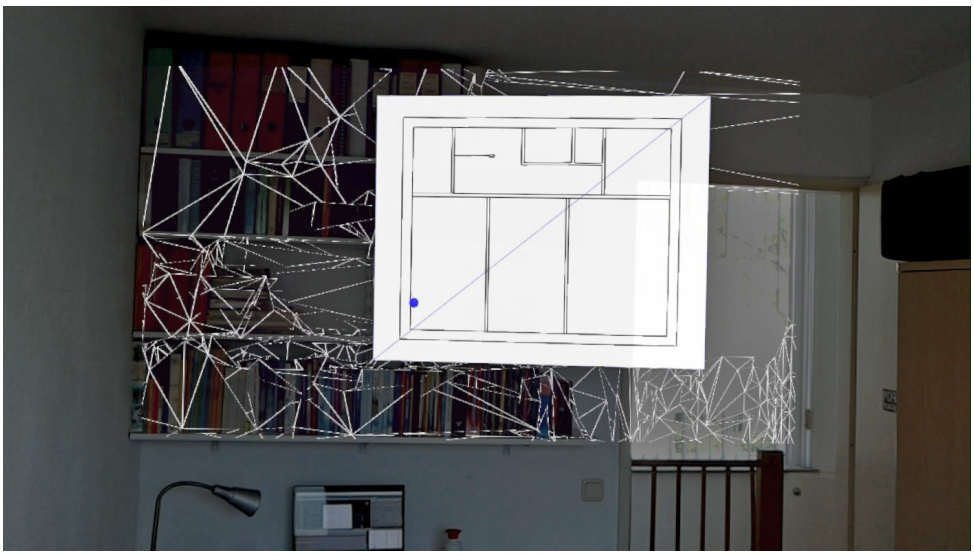


Figure 6.1: This is how the application looks like when run. The position (blue dot) is visible on a hologram of the floor plan placed somewhere in the room.

Section 6.1 describes all software and hardware that has been used to complete this thesis. Section 6.3 explains the application that is constructed to (1) map the environment and acquire all needed data to run the simulation and (2) have a real-time indoor positioning application. Section 6.4 elaborates on the implementation of the simulations that are used for the results.

6.1 Tools

This section describes all tools used in the thesis. First, the hardware is described, which is followed by the software, programming languages and modules used.

6.1.1 Hardware

The MH is the main device used for positioning. This device is already described in the theoretical background, section 2.4. The MH 1 is used, even though the MH 2 is already on the market. The second version however is in possession of neither the university or the company. The computer used to run the simulations is a HP ZBook Studio G5 mobile Workstation X360 running on Windows 10. It has an

Intel Core i7-8750H CPU, with a 64-bit Operating System. The hard disk is a SSD and the computer has 16GB RAM.

6.1.2 Software

Unity is the main software environment used, as it is has the ability to make applications for the MH. Other programs are used as well. A list of software, programming languages and modules is provided below.

Unity

Unity is a 3D software development program that its main use can be found in the development of games. This specialisation in gaming has the result that the platform is also very suitable for geo-related purposes. 3D games often make use of transformations, geometry operations. There are many advantages of using Unity instead of common GIS programs. The embedded 3D environment makes 3D visualization simple. Time is a native feature in the program: in GIS programs, someone has some dataset, that can be queried and transformed. In contrast, Unity has a built-in 'update'-method, that checks if something needs to be updated at every frame. This is needed for games, as (among other things) user input changes the state of the game, but also comes in handy when making applications on the Microsoft Hololens, where content needs to be changed every time a user looks into another direction or walks to some other area. The programming language used with Unity is `C#`. The combination of Unity and `C#` is used for developing the application on the MH and to run the simulations. The list of `C#` and Unity modules/packages used for the research is as follows:

- The Mixed Reality ToolKit (MRTK) 2.3 of Microsoft is used for support with the MH. This is a progressed version of the HoloToolkit.
- Accord is the mathematical library used. It can solve equations, such as least squares or systems of linear equations.
- NuGet for Unity is used to import all packages
- FBX exporter is used to export the spatial mesh of the MH to use in other programs for better visualization.

One disadvantage of using `C#` and Unity is the limited amount of geographical modules like Python's Shapely, that has more advanced and optimized tools for spatial data processing. While the Accord module has some geometry sub-module, this works with different vector formats than Unity's native vector formats. Therefore it is chosen to implement all spatial operations that are not provided by Unity by hand (closest points; intersections; bounding boxes etc.).

AutoCAD

As explained in chapter 5, AutoCAD is used for preprocessing of the floor plan data.

Python

Python is used to process the results and visualize the (non-spatial) data. Python is chosen over `C#` mainly because of its ease of use, ability to construct a program in a short amount of time and its extended plotting libraries. Modules that were used in python are:

- NumPy, a module that supports working with N-dimensional data
- Pandas, a module that makes use of dataframes possible and helps structuring the data.
- Matplotlib, a plotting library used to make the graphs in this research
- Seaborn, an extension of matplotlib, used for data visualization

QGIS
QGIS is a GIS program that is used for data preparation (see chapter 5) and for the visualization of spatial data.

6.2 Overview implementation

The application has two modes: a real-time registration mode and a simulation mode. Both modes run in the same application, but while the first uses the input of a real MH, the second imports all input from a previous scan. The first mode has some more features than the second, as it is run in-situ. These are explained in section 6.3. The registration will be elaborated on in section 6.4.

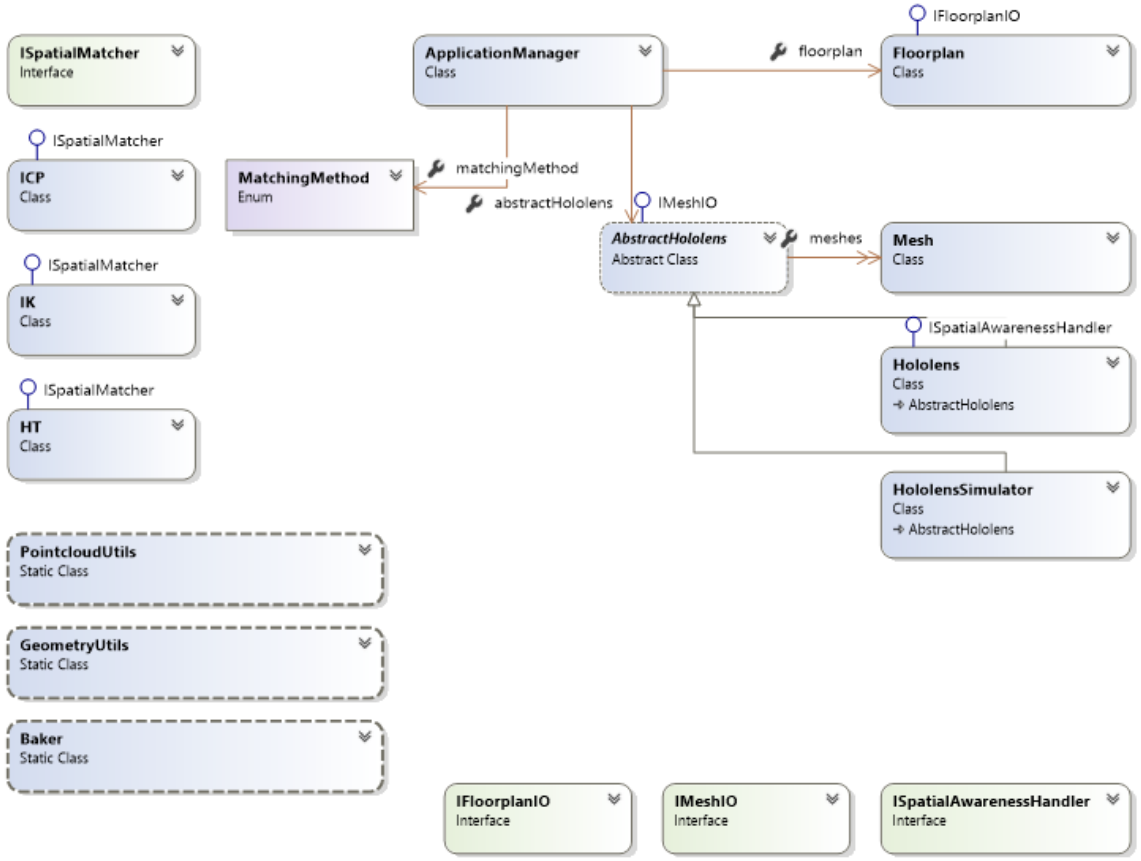


Figure 6.2: UML diagram of implemented application. The Application Manager manages both the simulated as the real-time variant. All components are shown in their expanded version in appendix C

.

Fig. 6.2 shows the structure of the application. It is managed by a program manager, a class that links all separate building blocks. It is the place where all properties are set and that holds all the data. On top of all properties, there are three main entities that included in this manager. It contains the floor plan. This class implements the IFLoorplanIO interface, that is needed for importing and exporting the floor plan data. The Application Manager also contains a Hololens, or a Hololens Simulator, depending on the settings. Both inherit from the abstract Hololens class, that mainly holds the mesh geometries and list of positions and implements the IMeshIO interface. This interface is used for importing or exporting the meshes. Depending on the mode (real-time / simulation), the meshes should be im- or exported. Lastly, the Application Manager also holds one item of the ISpatialMatcher Interface. All three registration methods inherit four functions from this Interface, with four base functions: the parameters

are set in the *Initialize()*-function. The *Compute()*-function starts the computation. The *Computing()*-function determines if the computation is finished. This enables to work with coroutines, that make it possible to visualize the position while the algorithm itself is busy computing. The last function is the *getTransformationMatrix*, that returns the current transformation matrix computed by the registration algorithm.

There are four static classes that provide utility functions that are needed for the program to operate. A point cloud utility function is used for all operations performed on point cloud entities (such as transformations, clipping on a buffer). A geometry utility function is used for all other geometrical operations (point-to-line distances, bounding boxes, intersections). The Hololens utility contains all operations that should be performed using the Hololens. The Baker is a utility function to visualize all geometries, such as meshes, positions, the lines of the floor plan.

6.3 Hololens-specific implementation of the application

The goal of the application is to show the real-time position of the MH on a floor plan hologram. The application needs to have a number of functionalities. It needs to:

- process the input of the Hololens, such as the mesh acquiring by the SLAM algorithm and user input (tapping)
- have a user interface to let users perform actions
- show the floor plan somewhere in the space
- find the position of the device at start-up with an initial transformation
- adjust the position of the device on the floor plan every time the carrier moves
- correct for errors in the position using registration algorithms

The last item in this list is possible in the application. It is not performed in practice, because the registrations are performed in the simulations that are run afterwards, to ensure that all circumstances are the same between the different algorithms.

6.3.1 Processing input

The Mixed Reality ToolKit is used to set up a project with Unity. The toolkit has a number of services that interact with the MH. There are four systems active in the case of MH, excluding possible extensions:

- The camera system can be used with the regular RGB camera to capture screenshots / videos.
- The input system handles all possible actions, such as a raycast-hit or a hand gesture
- The spatial awareness system is responsible for all input coming from the SLAM of the MH.
- The diagnostics system is a tool to analyse if the device has any application issues.

The spatial awareness system communicates via a system handler. An observer can subscribe to certain events and are notified whenever there is a change. In this case, the event is an addition, removal or update of a mesh. Every mesh has its own coordinate system, that is defined by a position and orientation in the 'world space', defined by a *spatial anchor*. To transform the meshes to the world space, they need to be deconstructed into their vertices. These need to be multiplied by the transformation matrix given with the spatial anchor and constructed again. All events have a unique event ID. Therefore, the meshes can be saved on their ID, time of event (measured in seconds after the start of the application) and the type of change (addition, update or removal). On top of that, the meshes can be loaded and visualized in the application, which makes the meshes visible aligned of the real world as holograms (see fig. 6.3).

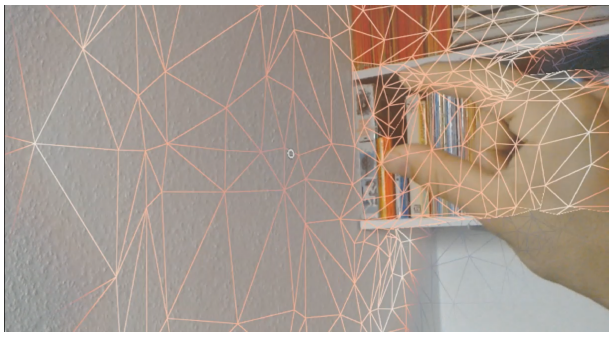


Figure 6.3: Hololens spatial mesh overlaid on the real world

The interaction with users comes from the input system. There are two features of the input system used: the gaze provider and the interaction manager. The gaze provider can be queried to give a direction that the carrier is currently looking into. It also provides a position and normal vector of the first object that is hit when looking into that direction. The gaze provider useful when selecting holograms or objects in the real world (such as walls). The interaction manager deals with the hand gestures. A script can subscribe to an event, similar to the system handler. When a 'tap gesture' (see fig. 6.4) is performed, this event can trigger functions inside of a program. Tapping on the Hololens is equivalent to a mouse click on the computer.

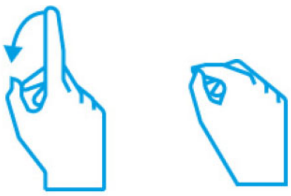


Figure 6.4: Tap gesture on the Microsoft Hololens (adapted from Tang et al. (2018))

6.3.2 GUI

The MRTK provides a standard 3x3 menu that can be personalized for a particular application. It floats in space in the view sight of the person, and can be hidden by tapping the left upper arrow (see fig. 6.5). When a button is triggered, it can push an event to a custom defined function. In this case, six of the nine buttons are in use. The buttons are explained in the order left to right; top to bottom. When the first button is triggered, a floor plan of a building could be chosen. Currently, this is set to architecture, but also other buildings could be shown, depending on which resources are available. The second button lets a person choose a point on a wall that can be referenced with the floor plan for an initial fit. The third button lets the person place the floor plan somewhere in the room and select the wall on the floor plan to reference the wall in the real world to. If both the wall in the real world and on the floor plan is set, the initial transformation is automatically performed. In the second row, only two buttons are active. If the initial transformation is correct, one could press the first button in that row. It will start a registration automatically every fifteen seconds. It could also happen that the initial transformation is rotated with 180 degrees. This can occur because the direction of the normal on the floor plan is not known, since the start- and endpoint are arbitrary. Therefore, it could happen that the position of a person is on the wrong side of the wall that is selected. The middle button in the second row fixes this problem. The last button in the last row stops the program.

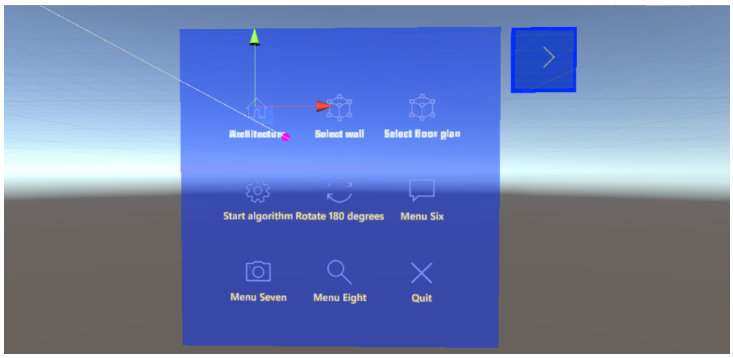


Figure 6.5: Menu hologram that is used for real-time positioning mode.

6.3.3 Selecting a wall

The selection of walls is as follows. The spatial awareness system loads all spatial meshes in a separate layer in Unity. The gaze provider of the input system provides a hit position and hit normal of the first obstruction in this layer. A standard rotation of 90 degrees over the X-axis is performed, because the Hololens (and Unity) have the Y-axis as vertical axis. The floor plan is in 2D, so it only has a X and a Y component. Therefore, to map anything from the Hololens onto the floor plan, the Y and Z axis need to be swapped. This rotation will be described as transformation matrix T^{Y-axis}_{Z-axis}. Both the hit point and the hit normal need to be transformed by this matrix.

6.3.4 Selecting a wall on the floor plan

The floor plan is placed on a canvas that has a maximum of 1.6 by 1.6 meter. This is an arbitrary size, but is found to be a good trade-off between having enough detail in the floor plan and not having a floor plan that is too big. To place the floor plan onto the canvas, the floor plan needs to be translated to the centre (0,0) and scaled. This transformation is encaptured by a 4x4 matrix. The transformation of the floor plan to the canvas is described as $T^{Floorplan}_{Canvas}$. A bounding box is constructed from the floor plan lines and the centre of the box is placed at position (0,0). The scale is defined by dividing 1.6 by the maximum of the width and length of the bounding box.

When the 'Add floor plan'-button is tapped, the floor plan is placed somewhere in the room. The gaze provider is used to find the position and orientation of the MH carrier in the room. The canvas is placed two meter in front of the person, by multiplying the orientation vector by two and adding the position vector. If there is an obstruction found by gaze provider that is closer by than two meter, the floor plan canvas is placed on that surface. Consequently, the transformation between the world-space and the canvas is described by the 4x4 matrix $T^{Canvas}_{WorldFrame}$.

The position that has been selected on the wall, can be selected on the floor plan hologram. A ray cast finds the hit on the canvas where the floor plan is on. This point is transformed back into the coordinate space of the floor plan, using equation 6.1. Using a closest line algorithm, the wall that is closest to the hit point can be found. The closest point on the line and its corresponding normal is found.

$$x_{floorplan} = T^{Floorplan}_{Canvas} \times T^{Canvas}_{WorldFrame} \times x_{WorldFrame} \tag{6.1}$$

Position on floor plan

The next objective is to find a initial transformation $T^{Floorplan}_{MH}$ that projects the position of the Hololens on the floor plan. This is achieved using the reference point and normal on the floor plan and the measured point and normal found in the space of the Hololens. The rotation, decomposed in a rotation

θ and an is computed using both normals. The translation is found by subtracting the rotated measured position from the reference position, as is shown in eq. 6.2.

$$\begin{aligned} \theta &= \arccos \frac{n_{ref} \cdot n_{measure}}{\|n_{ref}\| \cdot \|n_{measure}\|} \\ axis &= n_{ref} \times n_{measure} \\ R &= angleAxis(\theta, axis) \\ T &= x_{ref} - R \times x_{measure} \end{aligned} \tag{6.2}$$

$T_{MH}^{Floorplan}$ is the combination of the rotation and translation found in the previous equation. After that, the position is directly and in real-time updated every frame. This is achieved using the transformation shown in eq 6.3, where $x_{WorldFrame}$ is the current position as mapped by the MH and $x_{floorplanHologram}$ is the current position as visualized on the canvas in world space:

$$x_{floorplanHologram} = T_{Canvas}^{WorldFrame} \times T_{Floorplan}^{Canvas} \times T_{Z-axis}^{Y-axis} \times x_{WorldFrame} \tag{6.3}$$

After the transformation, the position is visible on the floor plan hologram (the result is shown in the start of the chapter in fig. 6.1). What remains, is to correct the errors in the initial transformation and correct drift errors by doing the registration. This will be elaborated on in the next section.

6.4 Registration

This section deals with the registration that can be computed either in a simulation run, or in the Hololens app. Such a program needs to have certain functionalities, that are listed as follows:

- A GUI needs to be set for changing the parameters of the positioning method. This includes the parameters specific for every registration algorithm (thresholds, maximum iterations etc.), but also general parameters such as which building is going to be used for positioning, what should be visualised, or how often a registration needs to be computed.
- The general data, such as the floor plan and the meshes need to be imported at start of the program.
- Each registration needs to be done as follows:

  (1) The data needs to be configured

  (2) The registration needs compute a transformation matrix

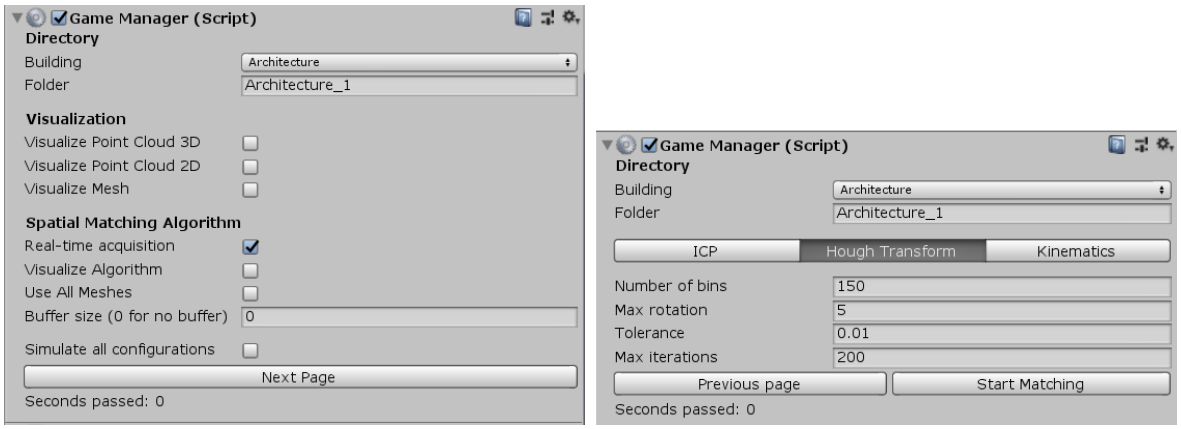
  (3) A function should check if the new transformation matrix is better than the old

These functionalities are needed for real-time positioning, but also using a simulation. Next sections will elaborate on these functionalities.

6.4.1 GUI program

An interface is made using the UnityEditor, a `C#` module for making a GUI inside the Unity environments. This interface is not only used to set the parameters for the simulation, but also for the real-time positioning. The interface is made up of two tabs: the first tab (fig. 6.6a) is used to set all general parameters. In this tab one can decide which building floor plan should be loaded and which folder should be used to import (in the case of simulation) or save (in the case of real-time positioning) all data. Also the visualization of the mapping are set, a 3D/2D point cloud and/or the mesh itself. Lastly, the parameters of the positioning are set on this page. These include (1) using real-time acquisition or a simulation; (2)

use all meshes or only a part of the meshes (see sec 6.4.3); and (3) which size buffer should be used (see sec 6.4.3 as well). On the second page, a registration method is selected and the parameters of that technique is set (see fig. 6.6b).



(a) Menu general property settings

(b) Menu registration settings

Figure 6.6: Menu used for property selection. The first page is used to set the general properties and the second is used to set the properties of the algorithm that is used.

6.4.2 Importing data

At the program start-up, some data have to be loaded. In the case of real-time positioning, only the floor plan of the building in question should be loaded. In the case of simulation, it also includes the meshes that have been mapped before, the positions that the MH has moved to and the initial transformation matrix that has been computed. The floor plan is saved in a text file, with the first row reserved for the geometry count and the remaining rows for the x/y positions of the start/end points of all lines.

Each mesh is saved in a separate file. One file, the *mesh manager*, has kept track of all meshes stored, the time they are added and the type of change (update/addition/destroyed). Using this manager, all meshes are loaded into the program consecutively. The meshes themselves are saved in a text file that is similar to a .PLY format. The first two lines are reserved for the amount of vertices and the amount of triangle faces in the mesh respectively. After that, all vertex positions are stored. Lastly, each face is stored by three vertex IDs. The name of the mesh is the same as its ID, therefore it is easy to import the correct mesh and store them in a dictionary.

Roughly every 0.3 seconds, the positions of the MH are stored in another text file. Each row contains the exact time that it is added and a x/y/z position. The initial transform matrix is also stored in a text file. It contains four lines with four entries, where each entry corresponds to the same row/column value in the 4x4 transformation matrix.

6.4.3 Running the program

When positioning in real-time with the MH, the registration starts right after the initial transformation matrix is computed and will not stop until the program is quitted. In the simulation, the registration starts when the button *start matching* (see fig. 6.6) is pressed. When that button is pressed, the parameters set in the menu are fixed and used until the program has finished or is manually quit.

Every 15 seconds a new registration is started, if the previous registration is finished. If the registration before did not finish computing yet, the new again starts 15 seconds later. All meshes that are new since the last registration are loaded and converted into a point cloud. If the option 'use all meshes' is selected, all points that have been loaded before are added to this point cloud. The point cloud is rotated by the

current transformation matrix (the initial transformation matrix in the case of the first registration). If a buffer is selected as an option, the transformed point cloud is clipped on a buffer of the selected size around the floor plan walls.

The Mean Square Error (MSE) is used as a metric used to evaluate if the registration succeeded or not. This metric is computed on the unclipped point cloud. This is the 'old' MSE that the new MSE after the registration should be weighted against to see if it has a better fit. The spatial matching algorithm is then initialized. When the initialization of the spatial matching is performed, not only the parameters are set for the registration, but also the floor plan is clipped. The lines are clipped on the bounding box of the point cloud plus two meter extra. This speeds up the computation. Next to that, the 3D point cloud is transformed to a 2D, simply by removing the z-component. Moreover, a simple spatial index is set up to find the closest points on the line faster. After the initialization is done, the new registration matrix is computed.

A new MSE is computed by using the new transformation matrix and the point cloud before buffering, after the registration is computed. If the new MSE is better than the old, the new transformation matrix is multiplied with the old transformation matrix, making this the new best fit.

7 Results

The following chapter shows the results of applying the three registration methods. In section 7.1, the performance of the MH without registration, but after the initial transformation is shown. This is as reference to the performance of the algorithms. In section 7.2, the accuracy assessment of the MH is described.

The following sections portray the results of the algorithms obtained with the experiments in the Architecture building. Section 7.3 shows the results of registrations with the ICP algorithm. This is followed by the section on IK (section 7.4) and on HT (section 7.5).

Every algorithm is compared on two aspects: The use the meshes from the start up to the starting time of the registration (*global* meshes / registration) and two use only the meshes that are new since last registration (*local* meshes / registration). The other comparison is made between the use of a fifty centimeter buffer around the walls of the floor plan in order to reduce the artefacts (such as furniture or persons) in the 3D model. The positioning experiment has been carried out five times, with ten measured points each. Therefore, there are fifty data points per registration configurations (and therefore a total of six hundred data points across all registrations). The overall effects of the configurations is shown in section 7.6.

The best performing configuration of each algorithm is tested on the experiments in the TU Library and CGI building, that are used as validation cases. The robustness of the method is described in section 7.7. In this section, the degenerate cases are discussed. Section 7.9 identifies the correlation between Root mean Square Error (RMSE) and the position accuracy and shows if the RMSE is able to be used an estimate for accuracy.

All algorithms have been tested on their scaleability. They are tested using multiple point densities and three different fragments of a scan of the Architecture building. The first benchmark will only contain a small area (fig. 7.1a). The second is a significant part of the trajectory (fig. 7.1b) and the third is the full trajectory (fig. 7.1c).

(a) Small mesh size (b) Medium mesh size (c) Full mesh size

Figure 7.1: The benchmarks are performed on the architecture building, using three mesh sizes and a varying amount of points.

7.1 Initial transformation

The experiments start with an initial transformation, that is estimated by the position and orientation given at the start of the scan. To evaluate the accuracy if the initial transformation, no registration has been performed afterwards. If the positioning is highly accurate after this initial transformation, the need for registration afterwards is low. If, however, the initial transformation is oriented slightly, the

Table 7.1: Errors between measured and ground truth points without use of a registration algorithm, after the initial transformation

Building	Mean error (standard deviation)	Max error
1	10.04m (5.77)	18.16m
2	2.26m (1.52)	5.27m
3	3.6m (2.06)	6.64m
4	1.93m (1.45)	5.0m
5	2.68m (1.37)	5.0m
all	4.11m (4.17)	7.93 (mean)

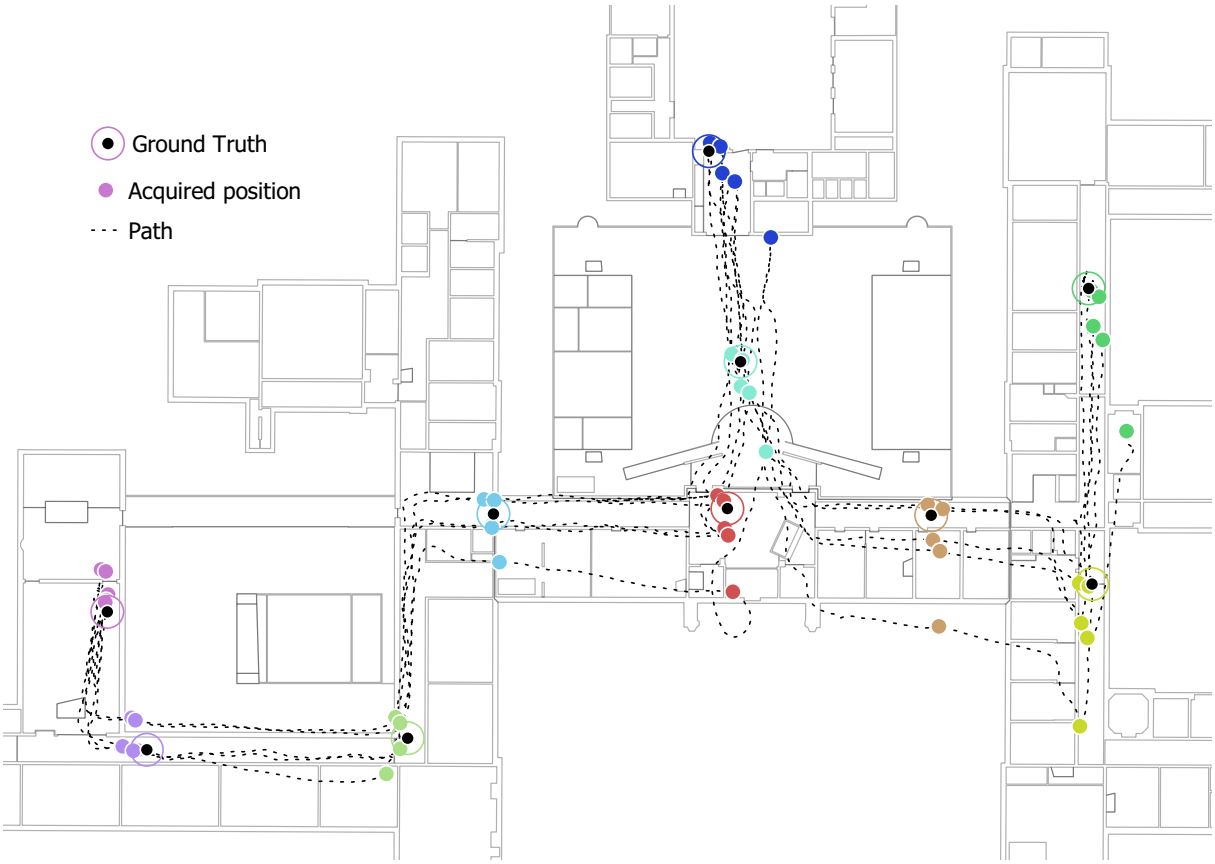


Figure 7.2: The trajectories of the five experiments after the initial transformation are shown in this figure. Every colour corresponds to one ground truth point. The coloured circles with the black dot is the ground truth, while the coloured dots are the measured points from one of the five experiments. The black line is the estimated trajectory. A similar figure of the TU Library and the CGI building can be found in B

small errors over time can accumulate towards large errors at the end of the trajectory. The results are shown in figs 7.2 and 7.3. The length of the trajectory ranges from 315 to 343 in the longest run. The initial error starts at between 0-2.5 meter and goes up to maximum 5-6 meters for the majority of the experiments. There are two cases where the error becomes larger and goes up to more than 18 meter (see table 7.1). This error has two reasons: First of all, the MH drift can take a role in the accumulating error, but also an incorrect rotation at start can have an impact. To disassemble these, the points should be fitted to their best position. The next section will deal with that.

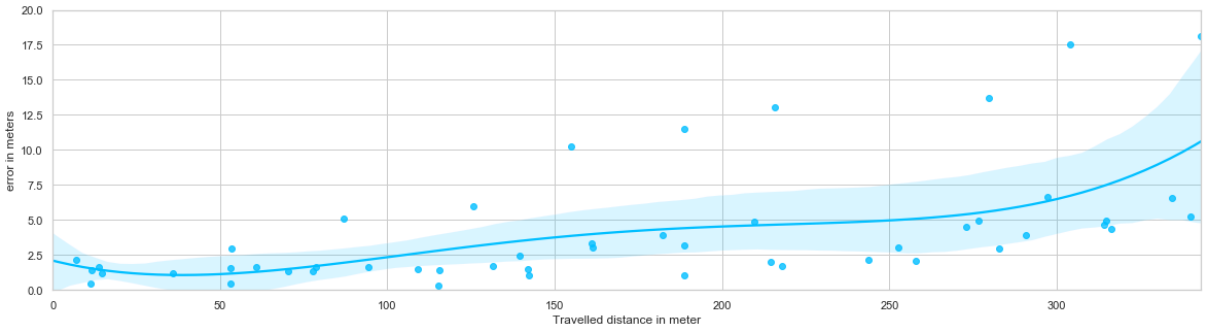


Figure 7.3: The accumulation of errors between measured points and ground truth from start to the end of the routes. The errors start around 1 meter and accumulate up to 18 meter. Results are separated between the three buildings.

7.2 Hololens drift error

The drift error of the Hololens is computed by translating the measured points so that the first measured point is exactly on its ground truth. Afterwards, the points are rotated to their optimal fit between the measured points and the ground truth using Single Value Decomposition. It is important to note that this 'best fit' of the measured points can only be computed after the mapping of the building and measuring specific positions that are also referenced on the floor plan beforehand. Therefore, it is not a possible method for positioning. However, it is a way to measure the drift of the MH. The result can be found in fig. 7.4 and table 7.2. The MH appears to have an excellent accuracy, where all mean errors are are around 70cm. This result is better Hübner et al. (2020) found. They found a drift error of around 2.4 meter in a trajectory of 287 meter. This is more than the drift found in this research, especially since all registrations have a total distance of around 340 meter.

Table 7.2: Errors between measured and ground truth points after applying SVD. This is the equivalent to the drift errors of the MH

Building	Mean error (standard deviation)	Max error
1	0.63m (0.34)	1.18m
2	0.68m (0.36)	1.78m
3	0.49m (0.34)	1.08m
4	0.71m (0.41)	1.38m
5	0.87m (0.4)	1.34m
All	0.79m (0.46)	1.78m

The achieved accuracy has a few implications. Firstly, the errors found after the initial transformation, described in the previous section, are mostly caused by orientation errors. While the four experiments that have up to 5 meters of error after the initial transformation have a rotational error of around two degrees, the first experiment has a rotational error of more than 9 degrees. Therefore, the large error found in experiment one (see table 7.1) is mostly due to this rotational error.

The low drift error leads to the computation time becoming less important than having an accurate result in the registrations. If a good result is found in the first registration, it remains a good result until the end of the experiment, since there is not much drift. Next to that, the use of meshes that are in proximity is proposed, because the drift error might confuse the registration algorithms. If a scanned object does not have the same shape as a reference object, it is hard to find an accurate registration. However, the drift errors are very small compared to the length of the trajectory. Therefore, the use of all meshes for the registration will likely be at least as accurate as the use of meshes in vicinity.

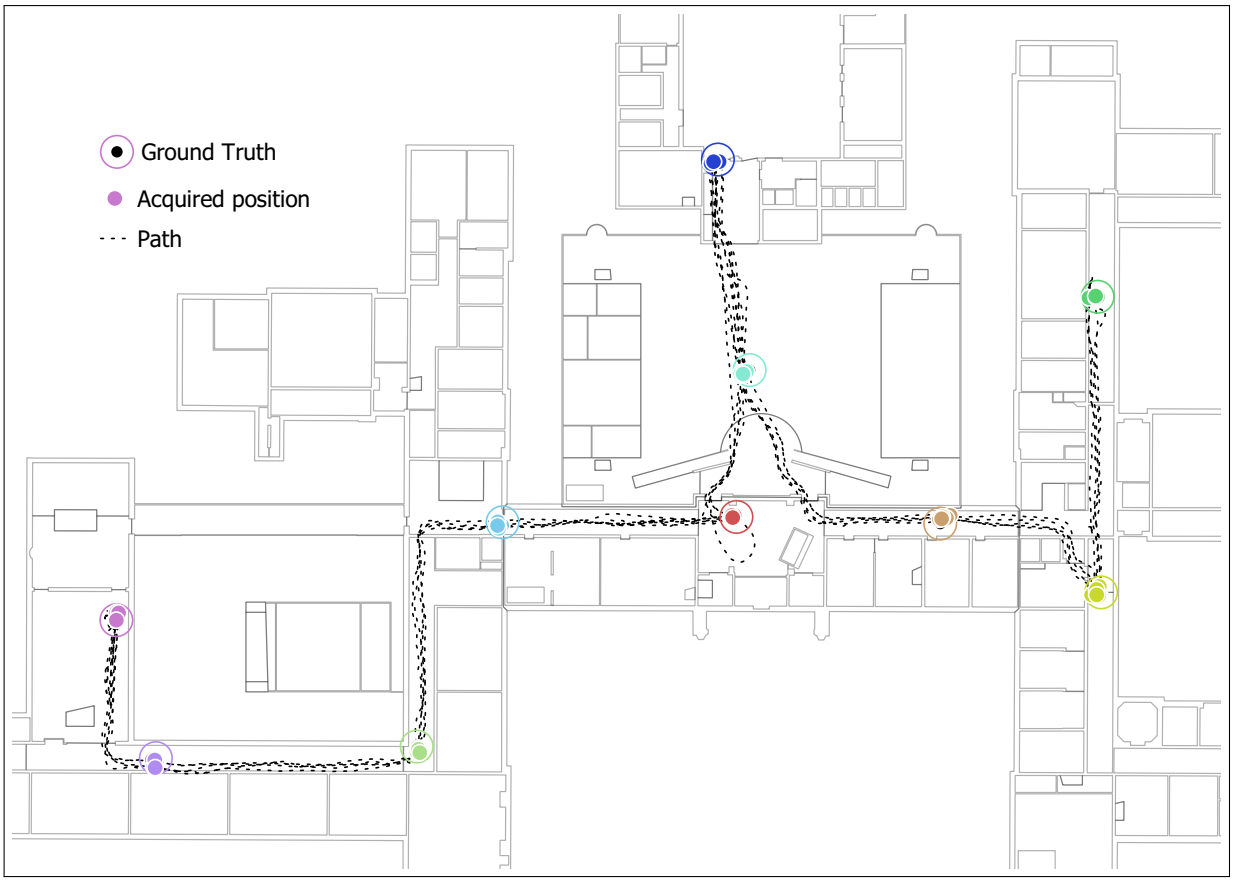


Figure 7.4: The trajectories of the five experiments after the applying SVD are shown in this figure. Every colour corresponds to one ground truth point. The coloured circles with the black dot is the ground truth, while the coloured dots are the measured points from one of the five experiments. The black line is the estimated trajectory. A similar figure of the TU Library and the CGI building can be found in B

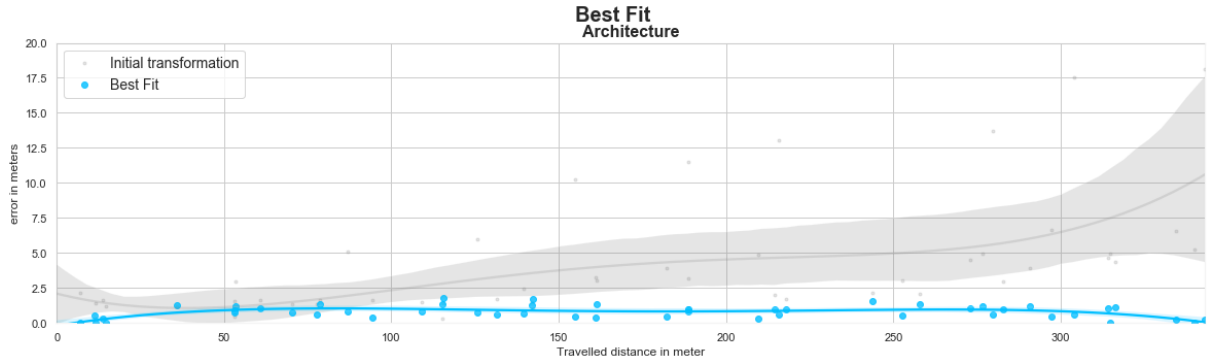


Figure 7.5: The accumulation of errors using SVD between the measured points and the ground truth. **note**: the gray accumulating area is the error after initial transformation, while the blue line is the transformation found using SVD.

7.3 ICP results

The ICP algorithm has some settings that need to be adjusted, described in section 6. In this case, the *standard* settings provided by Birdal (2020) are used. A maximum amount of 200 iterations is chosen.

The threshold is set to 0.001. This is not a mean error threshold, but percentage that the point cloud changes in an iteration, compared to the previous iteration. The number of levels is set to eight.

Table 7.3: Errors between measured and ground truth points with use of ICP, after the initial transformation

Meshes	Buffer	mean error (sd)	max error	Registrations	Correct registrations (%)	Computation time (sd)
global	50cm buffer	1.67m (1.6)	6.95m	23.8	8.6 (36.1%)	18.31 (18.86)
global	no buffer	2.59m (3.07)	10.69m	23	13.2 (57.4%)	26.31 (24.11)
local	50cm buffer	60.33m (48.02)	134.06m	18.8	9.8 (52.1%)	9.59 (28.14)
local	no buffer	4.98m (5.47)	27.27m	24	18.4 (76.7%)	10.9 (18.23)
No algorithm		4.10m (4.17)	18.16m		-	-

The four configurations all give varying results (see table 7.3). The algorithms using local meshes is clearly worse than the algorithms using all. While the most accurate algorithm has a 50cm buffer, the least accurate does as well. The latter is extremely off: a maximum error of 134 meter. One algorithm performs best and will be looked at in further detail. The configuration using all meshes, with a buffer has a mean error of 1.67 meter and with a maximum drift of 7 meter, compared to 18 meter using no algorithm.

The computation time of ICP is okay in the configuration that is accurate (mean computation time of 18.31 seconds. However, there with a standard deviation of 18.86 seconds, these algorithms varied a lot in computation times. There are around twenty-four registrations per experiment, of which a 36% has a better fit than the transformation before. While in other configurations the computation time is shorter (local meshes configurations are ¿10 seconds), the algorithms are not accurate at all.

One of the trajectories of ICP can be found in figure 7.7. It uses the configuration that is giving the best results in the case of ICP: all meshes are used for the registration, without a buffer. At start, a registration comes to a local optimum that is not congruent with the real position at that time (see fig. 7.7a). The correct position is not found until around 1/3 of the trajectory, where the position is corrected (see fig. 7.7b). This correct transformation is kept until the end of the experiment (see fig. 7.7c). If the configuration would be different, e.g. using a 0.5m buffer, it would be harder to find the correct position again. The position is off for a few meters and it is likely that a lot of the correct points would be filtered away due to the buffer.

Benchmark

The ICP algorithm is tested on a varying amount of points and size of mesh (see fig. 7.1). The results are found in table 7.4. The algorithm scales linearly in every case.

Table 7.4: Scaleability of the ICP algorithm, comparing number of points and size of mesh in seconds of computing time

N Points	Case 1: small area	Case 2: part of trajectory	Case 3: complete mesh
100	1.24	1.3	9.79
1,000	2.611	4.8	9.63
10,000	14.96	17.25	36.75
100,000	126.33	214.5	408.46

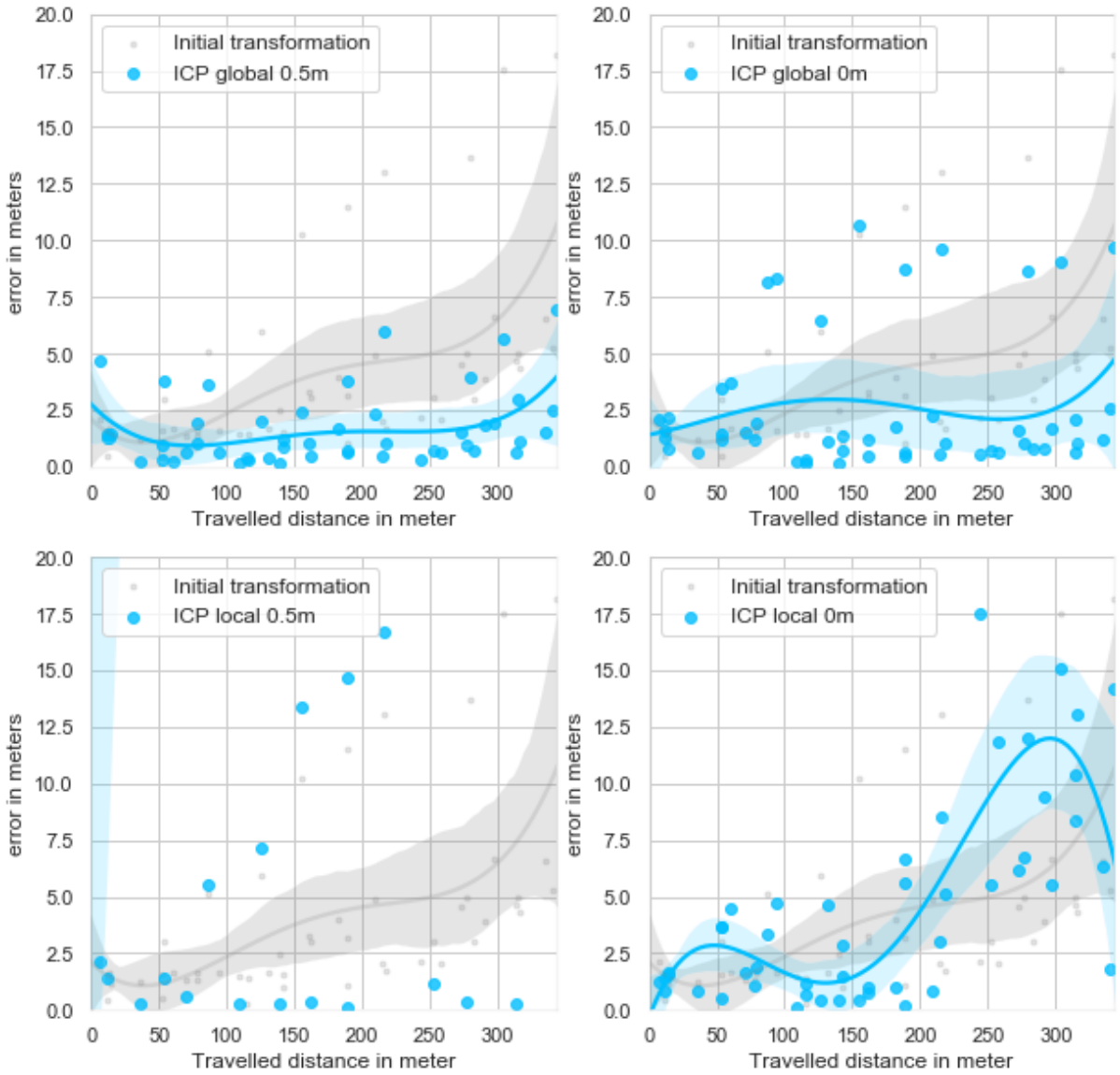
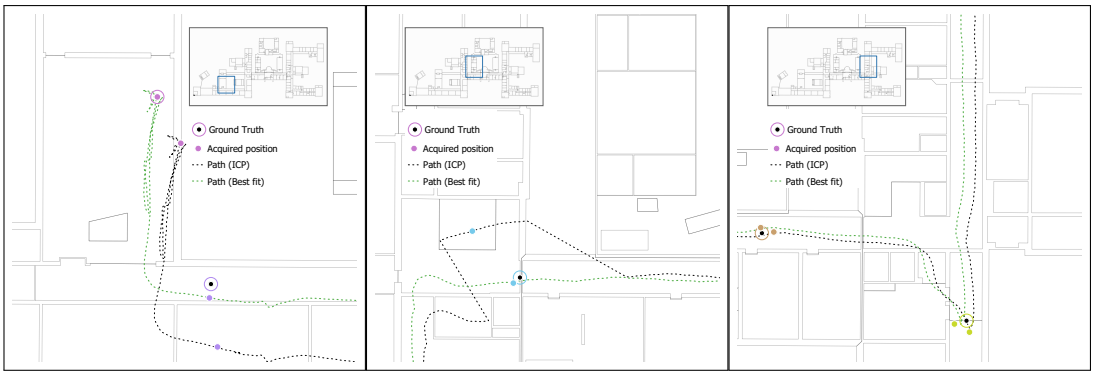


Figure 7.6: Accumulating errors in position using ICP

7.4 IK results

Also the IK algorithm has some parameters set. The maximum amount of iterations is set on 20, instead of 200 for the ICP algorithm. Pottmann et al. (2004) came to the conclusion that IK converges much faster than ICP, as they reach their optimum in about 12-17 iterations. Using less iterations makes the algorithm a lot faster. The threshold has been set to a mean error of 0.01m. This threshold has not been used in practice, since the mean error remains higher than the threshold at all times due to artefacts, even with the best fits.

All configurations have an average accuracy below four meter. However, the configurations using all meshes are again better than the configurations using only the meshes in proximity to the estimated position. Using a buffer of 50 centimeter is beneficial as well, but only in the case of using all meshes. Therefore, the configuration that uses all meshes and a 50 centimeter buffer has the best results: an accuracy of 1.61 centimeter, with a maximum error (or lowest accuracy) of 7.34 meter. 75 percent of the measured points was within 2.5 meter of the ground truth. The configurations can be compared in detail in appendix E.



(a) The start of trajectory (b) The middle of the trajectory (c) The end of the trajectory

Figure 7.7: Positioning ICP using all meshes and no buffer: registration at start is poor, but somewhere in the middle the correction position is found and held until the end

Table 7.5: Errors between measured and ground truth points with use of IK, after the initial transformation

Meshes	Buffer	mean error (sd)	max error	Registrations	Correct registrations (%)	Computation time (sd)
global	50cm buffer	2.31m (1.73)	7.68m	4.4	2.4 (54.5%)	28.34 (12.12)
global	no buffer	1.61m (1.8)	7.34m	3.4	1.2 (35.3%)	30.91 (12.56)
local	50cm buffer	3.83m (3.22)	11.39m	4.4	2.2 (50%)	25.38 (14.5)
local	no buffer	2.66m (2.29)	8.85m	3.4	2.4 (70.6%)	19.17 (12.73)
No algorithm		4.10m (4.17)	18.16m	-	-	-

The algorithm configurations took on average between 20-30 seconds, quite a bit longer than the proposed 15 seconds. On average there were 12-15 registrations. Something interesting happens when taking a look at the share of correct registrations among the total amount of registrations: the configuration that behaves best (global/no buffer) only has 35% registrations correct of the total of 4 per experiment. Therefore, each iteration, only once or twice the transformation parameter gets updated. Yet, the accuracy of this configuration is very good compared to the other configurations. It is likely that a good fit is found at start.

7.4.1 IK edge case concerning point-to-line distance

There is an edge case where IK is not able to find a solution to the registration problem. IK tries to minimize the point-to-line distance between the floor plan and the scanned model. Lines in this case are not vector geometries, with a start- and an endpoint, but infinite representations of lines. When all closest points are on lines that have the same orientation, this becomes a problem. While the registration distance orthogonally to these lines can be found correctly, the distances parallel to these lines cannot be computed. This is illustrated in fig. 7.9. The first registration shown (figures 7.9a to 7.9c) tries to register points to lines that are oriented mainly in two directions. The velocity vector gives a sensible output: if the center of the scanned area would be translated by this vector, it would be displaced by 0.2 meter in x and 0.3 in y. The second registration (figures 7.9d to 7.9f) is a section of a hallway that is mapped. All points are registered to lines that have the same orientation. Consequently, the displacement of the center by the velocity vector in x is -0.4 meter, a reasonable outcome. The displacement in y however, is -135 meter. Therefore this registration would give a incorrect output. The consequences are that no registration can occur before someone scans walls that are oriented in two directions. It is also likely that the registrations using meshes in proximity behaved worse, because if this edge case.

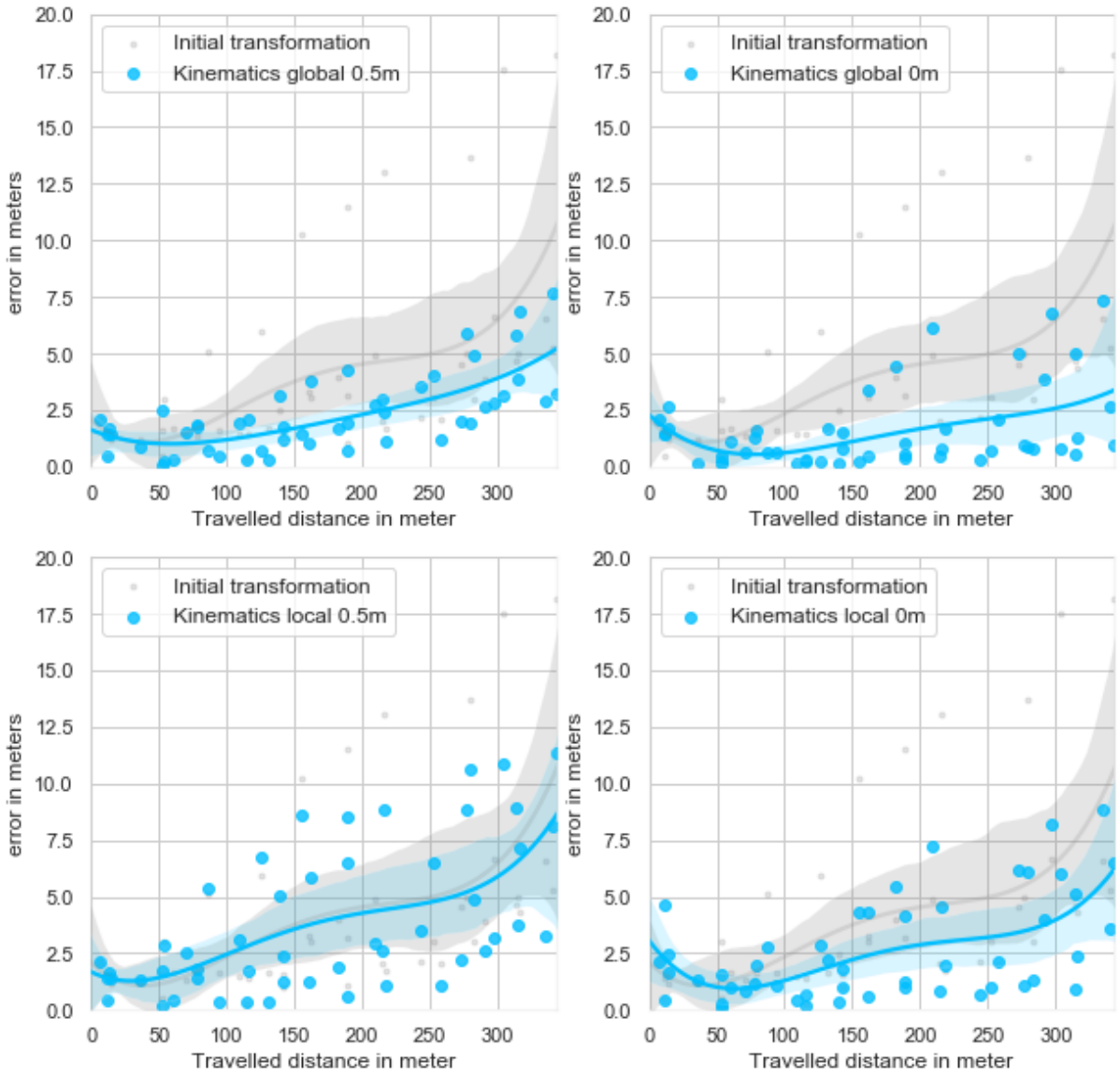


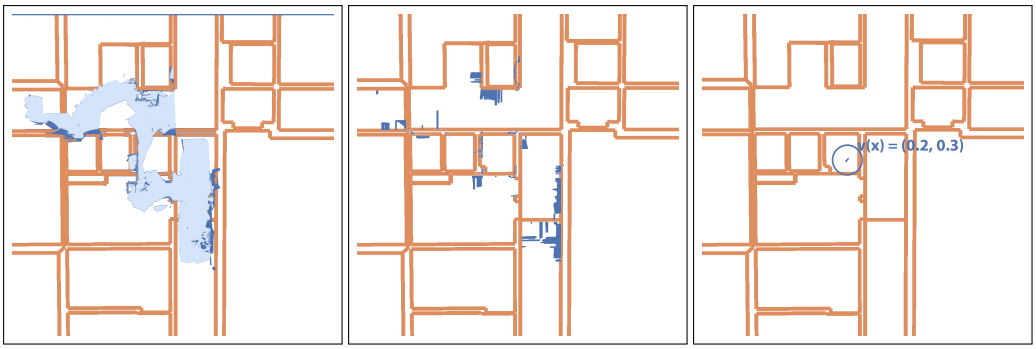
Figure 7.8: Positioning errors using Instantaneous Kinematics

Benchmark

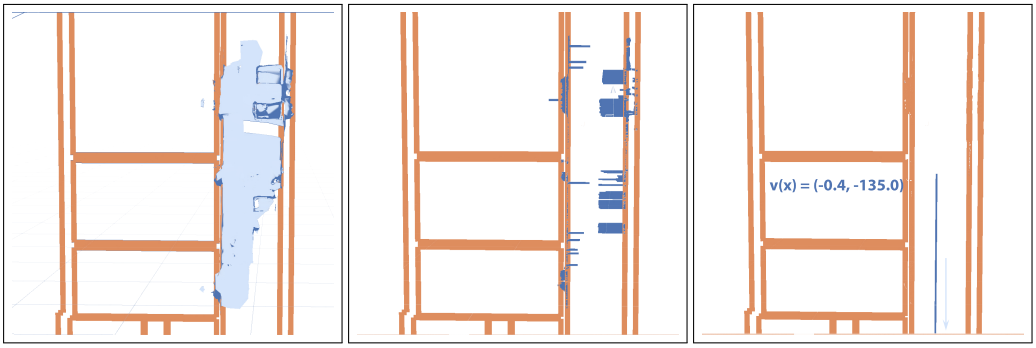
The IK algorithm is tested on 100, 1,000, 10,000 and 100,000 points. The algorithm scales linearly with more points. The computation time when using a larger part of the mesh is lower than the ICP algorithm.

Table 7.6: Scaleability of the IK algorithm, comparing number of points and size of mesh in seconds of computing time

N Points	Case 1: small area	Case 2: part of the trajectory	Case 3: complete mesh
100	4.3	2.8	2.26
1,000	3.55	3.9	4.81
10,000	16.55	11.9	22.47
100,000	171.72	166.28	202.63



(a) Scanned mesh on the floor plan (b) Closest points of mesh walls to the floor plan (c) Displacement of center by the velocity vector v(x) of (0.2, 0.3).



(d) Scanned mesh on the floor plan (e) Closest points of mesh walls to the floor plan (f) Displacement of center by the velocity vector v(x) of (-0.4, -135)

Figure 7.9: Figures a-c show the process of obtaining the velocity vector v(x), when there is more than one direction of displacement. This results in a displacement of the center of the scanned mesh of 20cm in x and 30cm in y. Figures d-f shows the same process, but all displacements are in x-direction. This results in an arbitrary displacement of the center of the scanned mesh in y, in this case -135 meter.

7.5 HT results

Two methods for the translational component are tried: using a similar method to Ni et al. (2013) that correlates the histograms of source and destination models, or using an ICP variant that only computes the translation. The result is discussed in the next section. After that, the general results of HT will be described.

7.5.1 Using correlation with a histogram or ICP as translational component in HT

In the algorithm using Hough Transform, the rotational component is computed separately from the translational. Ni et al. (2013) use MATLAB's xcorr, which is similar to the method using a density histogram for x/y in this research. This is compared to using an ICP algorithm, that only computes a translation. The method that tries to correlate the floor plan and the meshes with a histogram in x- and y-direction behaves poorly when doing partial registrations. If the input data of the Hololens does not fully overlap with the reference data of the floor plan - something that will happen often, as registrations are computed while scanning the building - it is impossible to find a correlation between the two datasets. This is illustrated in fig. 7.10. The object on the right side inside the black square is in reality a big staircase. This staircase is weighted in the histogram in the case of the floor plan, but not in the case of the scanned mesh. It will likely give an incorrect translation. ICP picks the closest points to the reference data, therefore has less problems while doing a partial registration. As a simple

translational ICP did not give the same problems as the correlation method, this was chosen as the translational component of the Hough Transform algorithm.

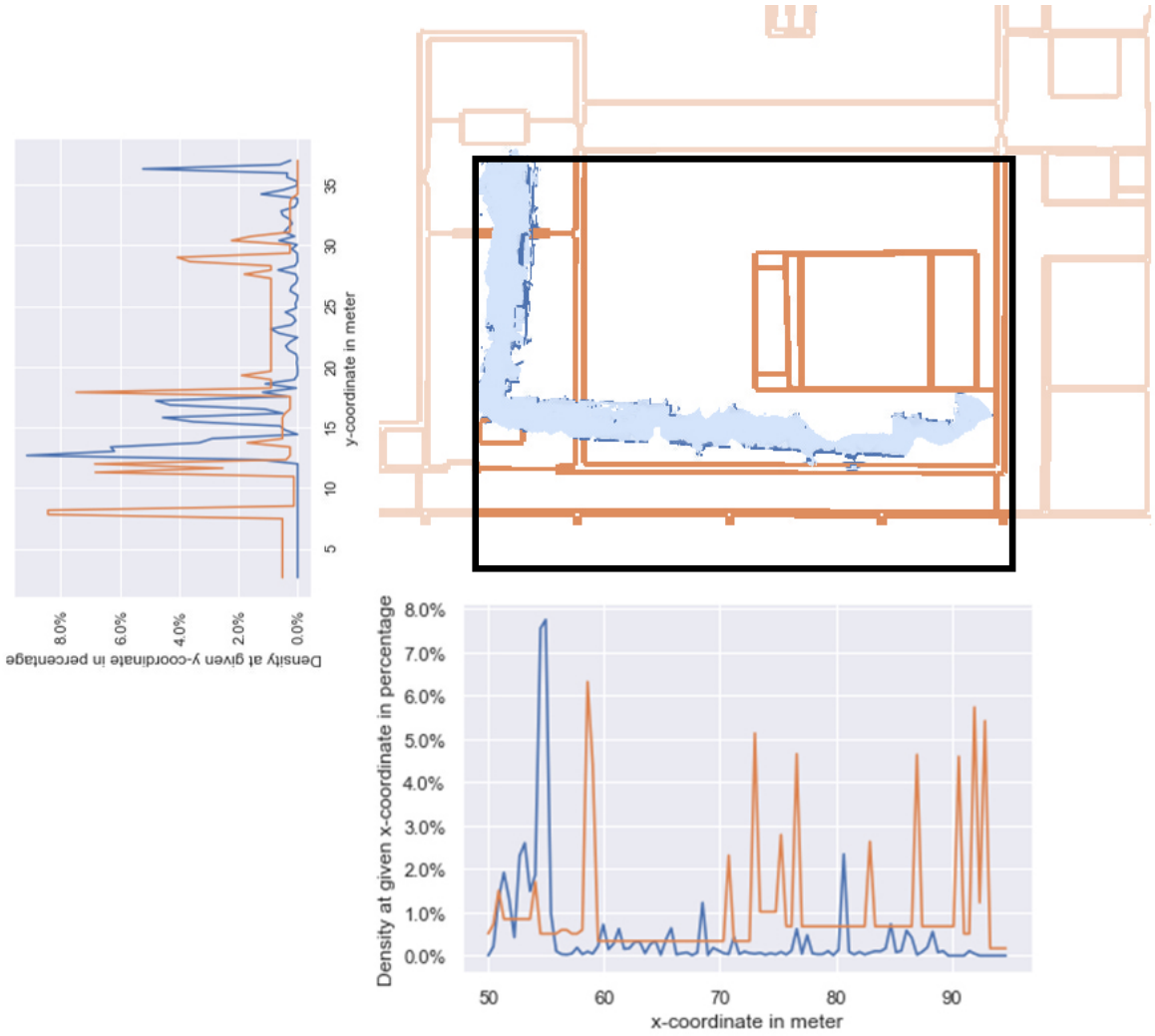


Figure 7.10: This figure shows a part of the scanned Architecture building (in blue) and its floor plan (in orange) after rotational alignment with Hough Transform. The graphs show the density of points in X- and Y-direction of the scanned object and the lines of the floor plan in the same colours. The graphs are aligned in respectively the X- and the Y-direction with the image. A good alignment is difficult to find, as there are many spikes found in the floor plan, that are not scanned by the MH.

There are a few parameters that need to be adjusted when using HT. The rotation of the point cloud is found with use of hough lines of the points and the hough lines of the floor plan lines. These are aligned using a histogram. Therefore, a number of bins needs to be set. This is set to 100 bins. As the initial transformation already has an estimate for the rotation, a maximum rotation can be set, to speed up the algorithm. This is set to 5 degrees. The translation is done with a simple translational ICP algorithm. A maximum iterations and a threshold needs to be set for these. The threshold is set to 0.01, and the maximum iterations to 100. Less iterations are needed than with the full ICP algorithm, since only the translational component needs to be computed.

Compared to the other two algorithms, HT has a good accuracy overall. The mean error of all configurations is somewhere between 80 centimeter and 2.75 meter. The error at the last measured point is somewhere between 2 and 14 meter. Using the meshes in proximity / a 0.5 meter buffer gives best results. This is remarkable, as this configuration is least accurate for both other algorithms. A maximum error of only 2.13 meter is found, with an average error of only 0.79 meter.

Table 7.7: Errors between measured and ground truth points with use of HT, after the initial transformation

Meshes	Buffer	mean error (sd)	max error	Registrations	Correct registrations (%)	Computation time (sd)
global	50cm buffer	1.52m (1.05)	5.69m	23.8	12.8 (53.8%)	17.94 (7.88)
global	no buffer	1.2m (0.89)	5.5m	24.2	15.6 (64.5%)	22.06 (10.23)
local	50cm buffer	0.79m (0.51)	2.13m	23.6	15 (63.6%)	6.74 (3.81)
local	no buffer	2.35m (2.62)	13.68m	23.2	21.2 (91.4%)	8.42 (5.37)
No algorithm		7.93m (5.76)	18.15m	-	-	-

As expected, using only meshes around a device speeds up the computation, as it is around twice as fast. All configurations have on average 23/24 registrations, but it varies how many are better than the fit before. The most accurate result is also the fastest, with a computation time of only around 7 seconds, which meets the requirements of an algorithm faster than 15 seconds.

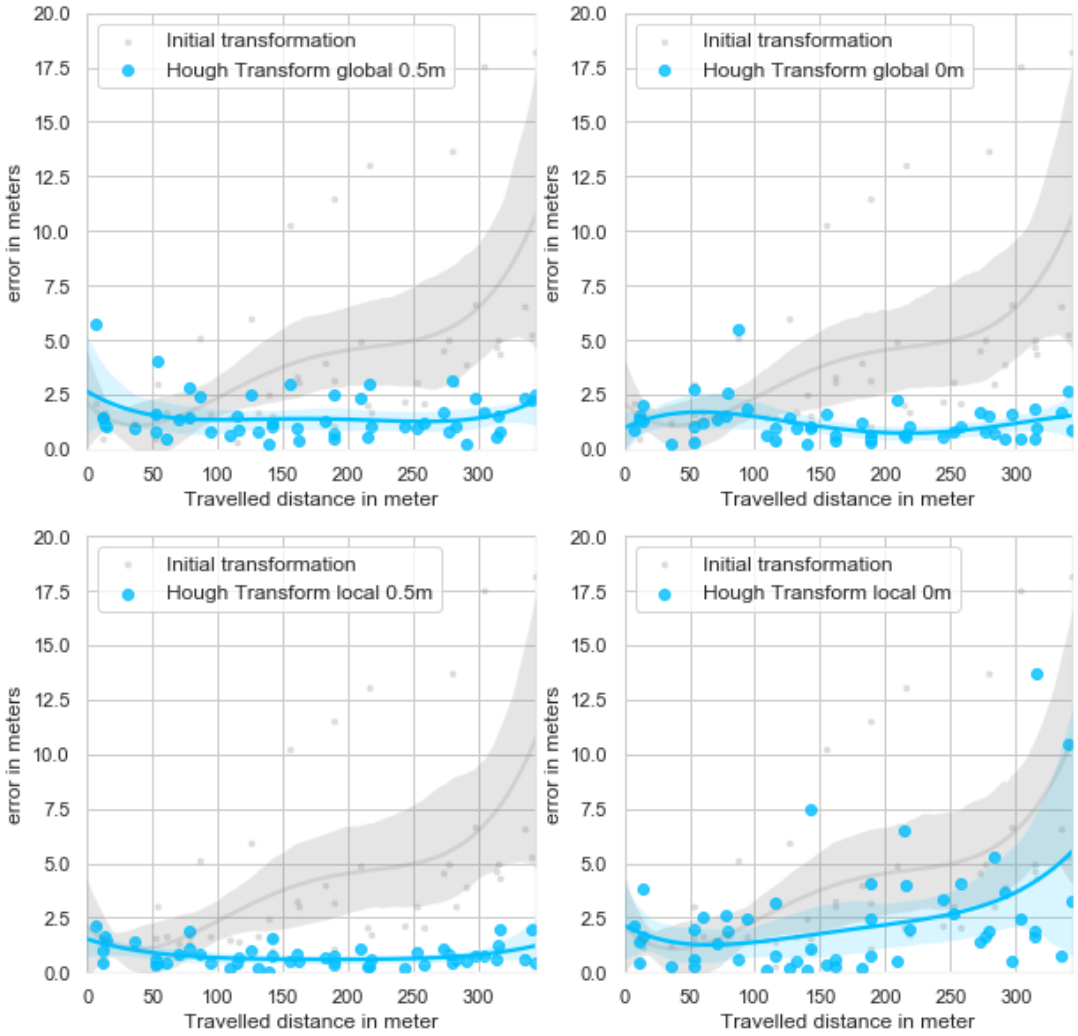


Figure 7.11: Positioning using Hough Transform

Benchmark

The HT algorithm scales linearly, similar to IK and ICP. The size of the mesh seems to have little impact on the computation time, therefore making it more suitable to larger meshes.

Table 7.8: Scaleability of the HT algorithm, comparing number of points and size of mesh in seconds of computing time

N Points	Case 1: small area	Case 2: part of the trajectory	Case 3: complete mesh
100	0.51	1.3	3.37
1,000	3.17	4.6	5.54
10,000	20.80	26.15	29.33
100,000	244.1	264.0	281.7

7.6 Configurations

For ICP and IK it can be seen that using all meshes for the registration is better than using only the meshes in proximity. For Hough Transform the configurations using local meshes with a 50cm buffer come to the most accurate results, while the use of local meshes without a buffer is the least accurate. In section 7.2 it is described that the MH only drifts off for around 1.5 meter. Therefore the biggest disadvantage of using all meshes is not present: if the MH would have a lot of drift, the registration would be performed with two input datasets that do not conform to each other. On the other hand, using all meshes is more robust than using the meshes in proximity. This can be seen in practice: if one of the registrations is not correct, but finds a local optimum, it is hard to find the correct position back, only using a part of the meshes. This could be illustrated with an example,shown in fig. 7.12: if two rooms have the same shape and the position before the registration is somewhat off, it is hard to find the correct position using a part of the meshes. However, when both rooms are mapped and the meshes of both rooms are used in the registration, an algorithm can find its correct position easier.

The results of using a buffer are also varied. Using a buffer on a registration with an initial fit that is already okay, can reduce noise and remove artefacts from the meshes. However, a problem occurs if the initial fit is not so good and in some areas the point cloud are more off than the buffer zone. Using a buffer can in that case remove good input data (see fig. 7.13). The ICP algorithm, that is sensitive to outliers, shows varying results in use of a buffer zone. In combination with meshes in proximity, it is worst with an average error of >60m, while it works best when using all meshes. IK works best without a buffer, while HT works best with a 50cm buffer. For both algorithms, the difference between the two configurations is not as big as for ICP.

7.7 Robustness

In section 7.2 it is shown that the SLAM algorithm of the MH gives an accurate position. Therefore, cases where the indoor positioning method fails, are due to the registration algorithms. Some edge cases are defined beforehand to be considered as possible causes of the registration to fail. These are:

1. Doors
2. Walls that do not exist in the floor plan
3. Walls that exist in the floor plan, but not in the present situation
4. Large spaces (>20x20 meter halls)

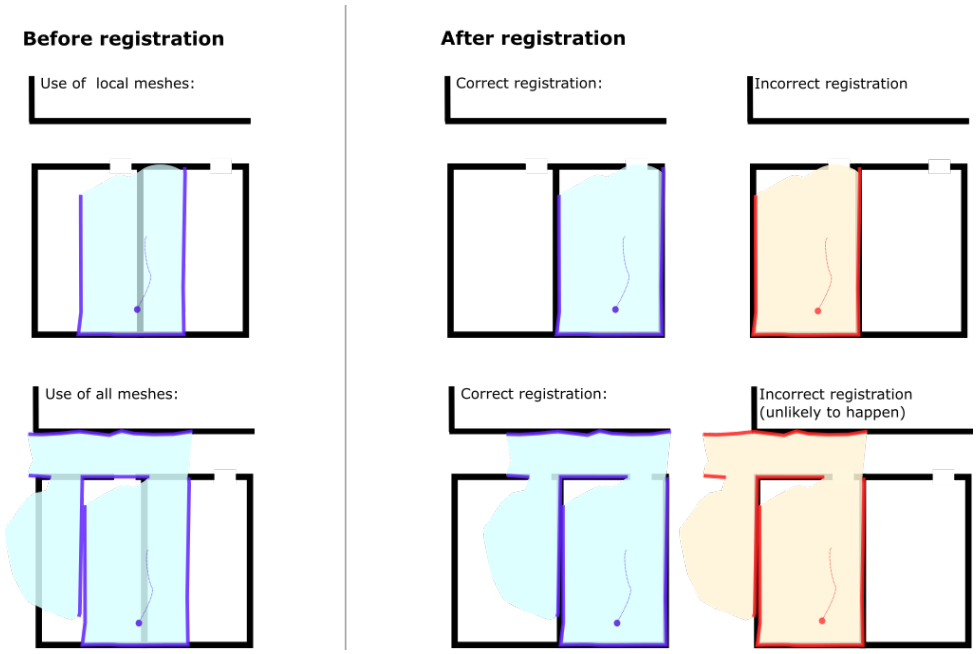


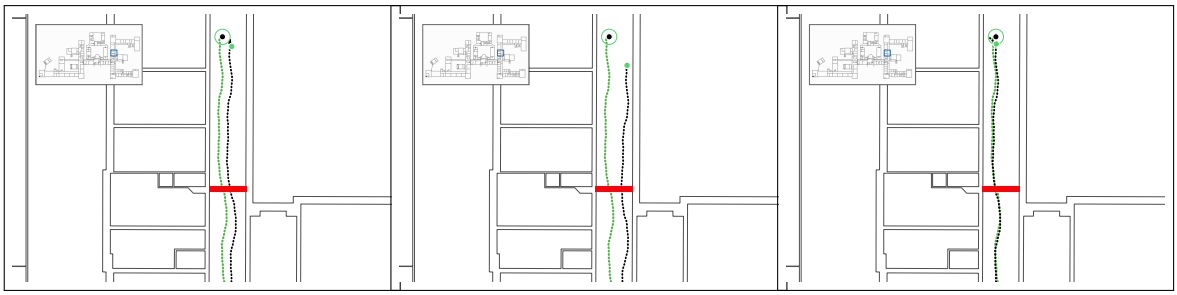
Figure 7.12: Suppose, before a registration, the scanned object(in blue) has a incorrect placement on the floor plan. If two adjacent rooms have similar characteristics, using only the local area around the current position for the registration (first row in the figure) will equally likely result in the correct as in the incorrect transformation. However, using all meshes that are scanned up to that moment (second row in the figure), it is more likely to compute the correct transformation.

Figure 7.13: If one of the registrations gives an incorrect transformation, it is difficult to correct this with a 0.5 meter buffer. In this case, the y-value is off by approximately 5 meter. All walls (dark blue) inside the red rectangle will be removed because of the buffer.

The circumstances that the 'global' configuration is the most accurate overall give some complications in researching if the algorithms behave well under these edge cases. The portion of the registration that is determined by the edge case is very small and will have little effect on the transformation parameter. That said, some could be determined by looking closer to certain cases. The configurations that are researched are the most accurate results of each algorithm, found in the previous sections. For ICP and

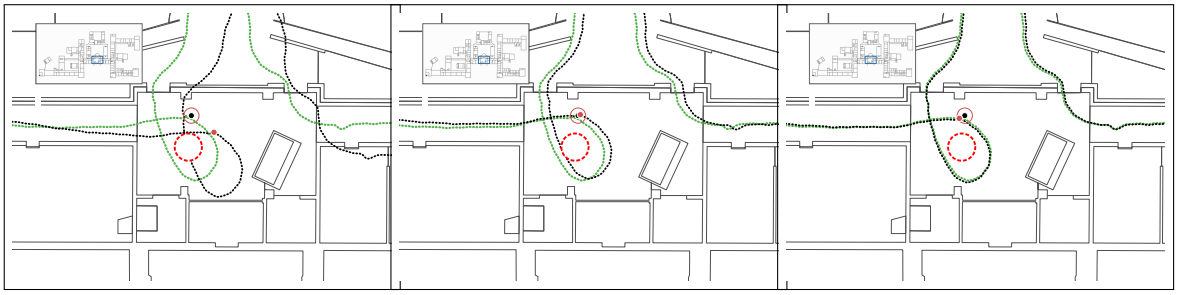
HT that means global meshes / no buffer. IK has the best results in global meshes / 0.5 meter buffer. Figures 7.14 to 7.17 show the respective behaviours of each algorithm spatially. Due to Covid-19, it was not possible to make reference photos on the exact locations of the degenerate cases. These have not been not made on the day the experiments were performed and after that the Architecture building has close for students due to national restrictions. Therefore, positions of the degenerate cases are shown on a mini-map on the figures themselves.

The the first case, shown in fig. 7.14, the route goes through a door. In this experiment, the direction of the route is from bottom to top of the figure. The HT algorithm is perfectly on line with the estimated



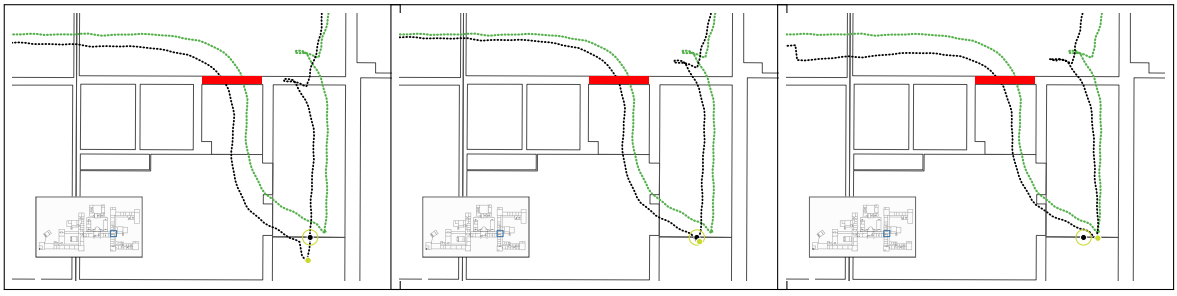
(a) ICP (b) IK (c) HT

Figure 7.14: Behaviour of algorithms on when walking through doors. The green line is the trajectory of the best fit after SVD, while the black line is the algorithm trajectory. The red area is the location of the door.



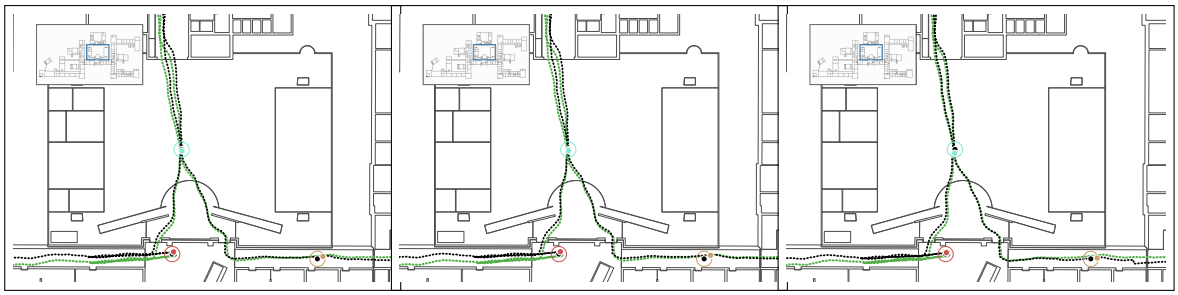
(a) ICP (b) IK (c) HT

Figure 7.15: Behaviour of algorithms on incongruence between the scanned mesh and the floor plan. The green line is the trajectory of the best fit after SVD, while the black line is the algorithm trajectory. The red area is the location of the door. The red circle is a large object that is scanned but not present in the floor plan.



(a) ICP (b) IK (c) HT

Figure 7.16: Behaviour of algorithms on incongruence between the scanned mesh and the floor plan. The green line is the trajectory of the best fit after SVD, while the black line is the algorithm trajectory. The red area is a wall that is present on the floor plan, but not in reality.



(a) ICP (b) IK (c) HT

Figure 7.17: Behaviour of algorithms when positioning inside large spaces ($<$ 20x20 meter).The green line is the trajectory of the best fit after SVD, while the black line is the algorithm trajectory.

trajectory after SVD. The other two algorithms do have an error, but the error before and after walking through the door is similar, therefore it is not likely that it is of any influence.

Fig. 7.15 shows the behaviour of the algorithms, when there is an object present in the real-life situation that is missing on the floor plan. The ICP algorithm has most error, but it is not likely that the cause has to do anything with the object in the middle of the space. The scanned object, transformed with the ICP parameters would still be in the middle of the entrance hall. If the object did influence the algorithm, the object would likely be on the sides of the hall, because it would be interpreted as a wall. The other two algorithms show a minimal positioning error, indicating that the algorithms have no problems with the special case.

In fig. 7.16, the red area shows a wall in the floor plan that is not present in the current situation. Again, the biggest error is found in the ICP algorithm. The figure suggests that the IK algorithm in this case is better than the error after SVD, or the drift error of the MH. While there are some accuracy errors in the other two algorithms, it does not seem to be caused by the wall present in the floor plan.

The behaviour of the algorithms in large spaces (<20x20 meter) is shown in fig 7.17. None of the algorithms show any trouble at all to give a correct position. None of the algorithms have a large deviation from the best fit trajectory. For the HT algorithm, it is likely that no registrations have been performed during this part of the trajectory, since it only takes the meshes in proximity into account and uses a 0.5 meter buffer around the walls. There are no walls inside the space, hence there are no points to compute a registration with. The other two algorithms use the meshes scanned since the start of the scan. Therefore, any registration that is computed will likely be similar to the registration computed before.

There are a few observations that can be made:

- The algorithms do not have more trouble with these cases, than with standard positioning (for a full view of the algorithms over the whole trajectory see appendices D to F)
- ICP is the least accurate in these cases. This is similar to the behaviour of the algorithms over the whole trajectory.

7.8 Validation with TU Library and CGI buildings

The method has been developed with use of the Architecture building. To generalize the results, more buildings need to be tested. Therefore, additional experiments have been performed in the TU library and the building of CGI, Rotterdam. First, the results after initial transform and the results using SVD are examined (see table 7.9). The initial transformation errors are expected to be lower than in the Architecture building. Trajectories in both new buildings are in a circle: they end at a position close to where they end. Next to that, while the travelled distance in the Architecture building exceeds 300 meter, the total distance in the library and the CGI office are respectively around 200 and 180 meter. Not only the error after initial transformation, but also the Hololens drift, computed using SVD is lower in

the case of the library. The CGI offices, however, relatively has a bigger error after initial transformation and a bigger drift error. A maximum drift error of 2.38 meter is found in this case, which is more similar to what Hübner et al. (2020) found.

Table 7.9: Errors between measured and ground truth points after initial transformation and after applying SVD for all buildings

	Initial transformation		After SVD	
Building	**Mean error (standard deviation)**	**Max error**	**Mean error (standard deviation)**	**Max error**
Architecture	4.1m (4.17)	18.16m	0.79m (0.46)	1.78m
Library	2.24m (1.74)	6.8m	0.47m (0.34)	1.36m
CGI	2.8m (4.19)	17.51m	0.57m (0.51)	2.38m
All	3.04m (3.62)	18.16m	0.61m (0.46)	2.38m

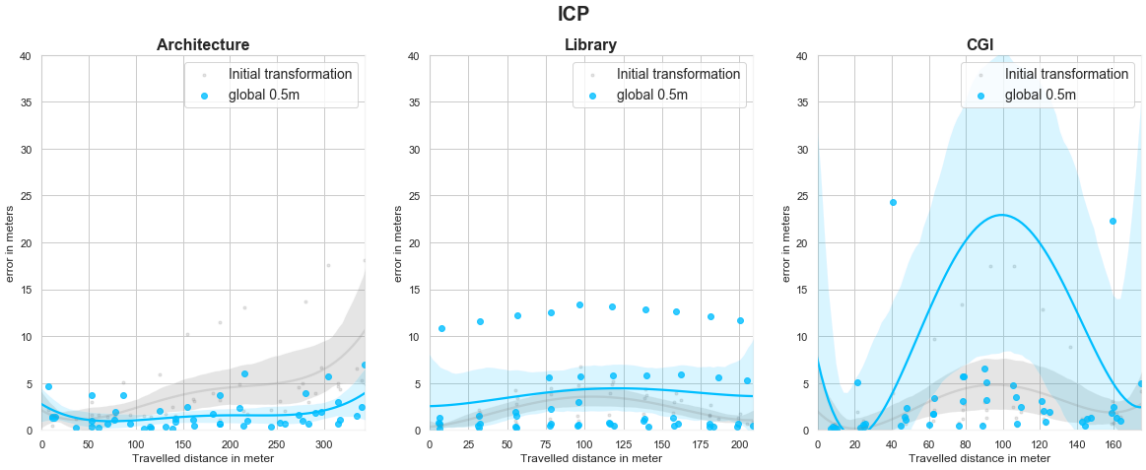
Table 7.10 shows the accuracy and computation times of the algorithms in the new experiments. As can be seen, while the results of the Architecture building look promising, the other two buildings show less bright results. The TU Library has a mean error varying between 3.4-5.5 meter for different configurations, with a maximum error of 13.4-19.3 meter. The CGI building has a mean error varying between 13-16 meter, while the total travelled distance is only 200 meter. The maximum error is 45 meter for the IK algorithm, and more than 100 for the ICP and HT algorithms.

Table 7.10: Comparison of tests in the Architecture, Library and CGI building of errors between measured and ground truth points for every algorithm and its best configuration.

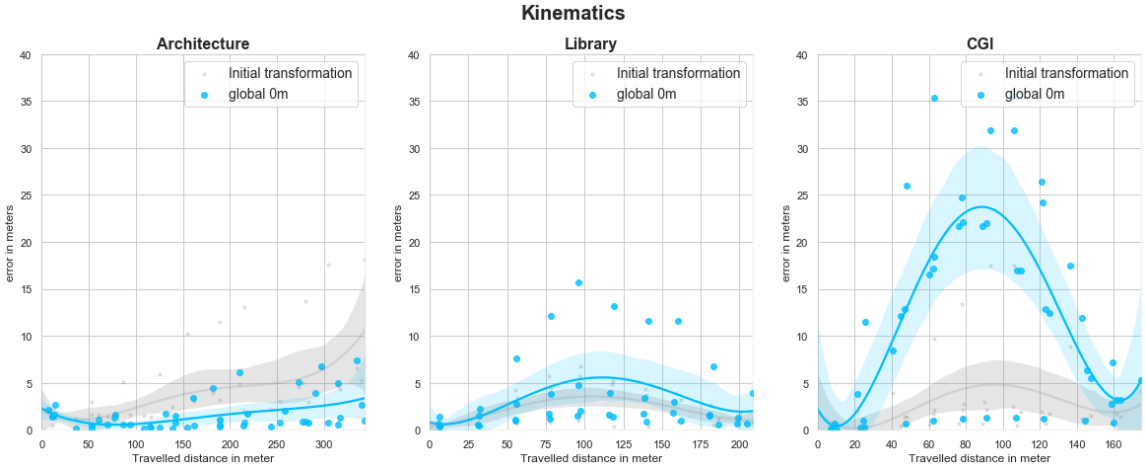
Building	**Configuration**	**mean error (sd)**	**max error**	**Registrations**	**Correct registrations (%)**	**Computation time (sd)**
Architecture	IK global 0m	1.61m (1.8)	7.34m	3.4	1.2 (35.3%)	117.99 (37.77)
	ICP global 0.5m	1.67m (1.6)	6.95m	23.8	8.6 (36.1%)	18.31 (18.86)
	HT local 0.5m	0.79m (0.51)	2.13m	23.6	15 (63.6%)	6.74 (3.81)
Library	IK global 0m	3.37m (3.99)	15.66m	1.6	1.4 (87.5%)	107.9 (51.56)
	ICP global 0.5m	3.79m (4.65)	13.37m	14.8	5 (33.8%)	62.21 (49.58)
	HT local 0.5m	5.43m (5.99)	19.31m	14.2	9.6 (67.6%)	9.94 (8.71)
CGI	IK global 0m	12.98m (12.6)	45.84m	2	2 (100%)	105.87 (46.79)
	ICP global 0.5m	12.08m (26.04)	102.31m	15	6 (40%)	22.24 (22.86)
	HT local 0.5m	15.86m (33.11)	113.74m	9.8	4.8 (49%)	4.13 (3.63)

There are a few reasons why the algorithms perform worse on the other buildings. First of all, the method is constructed with the Architecture building as testing data. That forms some bias in the developed program, as it will be adapted to the floor plan and mesh typology of that building. However, this cannot explain the huge difference in performance between the buildings. Some other causes can be identified:

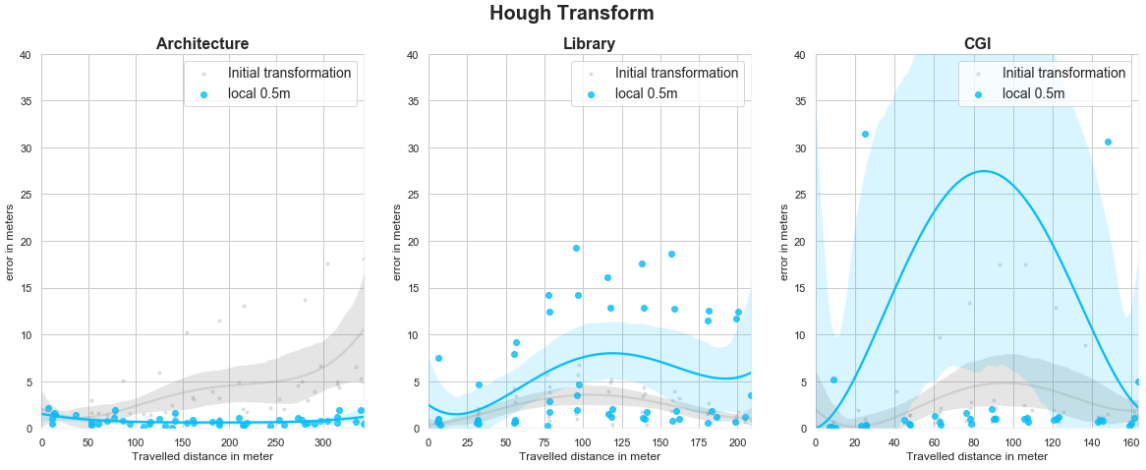
- A large part of the walls in the scanned areas of both buildings are made of glass. The sensors of the MH have difficulties scanning transparent materials. Therefore, the mesh output constructed by the MH does not have the same quality as it has in the Architecture building. Only columns and opaque areas on the glass (areas where posters or other papers are fixed on the wall) will be scanned properly. Therefore the amount of useful data for the algorithms is greatly reduced.
- The scanned area in the CGI building is office space. It is full of furniture: desks with computer screens, chairs, cupboards, scanners are all present in the scanned mesh. Moreover, the trajectory in the CGI building traverses two floors. Also stairs contribute to the amount of false positives in



(a) Performance of ICP (use of all meshes and a 0.5m buffer) in each building



(b) Performance of IK (use of all meshes and a 0m buffer) in each building



(c) Performance of HT (use of local meshes and a 0.5m buffer) in each building)

Figure 7.18: Best performing configuration of each algorithm, separated by buildings. Graphs show the error in meter over the travelled distance.

the shape registration. While there is some basic outlier detection, in the form of using only vertical elements and a 0.5m buffer around the walls, there is no proper method that filters furniture or other artefacts. This is considered to be the main reason of the poor quality in the case of the CGI

building.

- Lastly, it is important to note that these are the aggregated results of five experiments per building. The outcome varies a lot per experiment. In fig. 7.18 it can be seen that the results after most experiments actually have an accuracy comparable to the Architecture building.

These causes are educated guesses, that have not been tested. They seem like logical explanations for the deviation in results, but to proof these, additional tests should be done.

7.9 RMSE as estimate for accuracy

The root mean square error is a measure that calculates the mean euclidean distance between the point cloud constructed from the scanned object and the floor plan. A high RMSE indicates distant shapes, while congruent shapes have a low RMSE. There is no one-on-one relation with the position accuracy however. After registration an incorrect local fit could be found,: The point cloud could locally align perfectly with the floor plan, but in reality be at another place on the map. In such case, the RMSE will be low, while the position is incorrect. The opposite is also possible: a high RMSE, while the position is very accurate could be possible if there are many artefacts in the point cloud. Artefacts in this case are all points that are not a wall, but classified (e.g.furniture, humans passing).

A test is needed to identify the relationship between RMSE and the accuracy of the position. A linear regression (see fig. 7.19) shows that the correlation is significant (p¡0.001), with $R^2 = 0.25$. To ensure outliers have little impact on this regression, all position errors >25m are removed. Approximately 12% if the positioning error can be explained by the RMSE, which is not enough to estimate the accuracy.

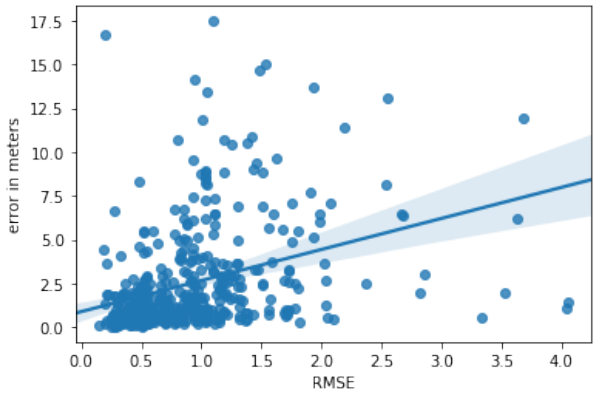


Figure 7.19: Scatterplot of error in meters between ground truth and measured point (y) and the root mean square error (RMSE) between the scanned mesh and the floor plan (x)

8 Conclusions

This chapter reflects on the results found in chapter 7. Firstly, the research questions are answered and general conclusions will are drawn in section 8.1. The strengths and limitations, implications and recommendations for future research are discussed in section 8.2.

8.1 Research questions

Four research questions were introduced in section 1.3. These are answered in the following section.

8.1.1 Feasible the spatial matching techniques

The first research question reads: *What spatial matching techniques are feasible to match a 3D mesh acquired real-time with the Microsoft Hololens with an existing 2D floor plan?* Seven techniques were discussed. Three techniques were deemed to be fit, based on a few considerations. Considerations that were taken into account were (1) the ability to perform partial registration; (2) the registrations should be able to find a registration in unordered point sets / shapes; (3) the complexity of the techniques. The three favoured registration techniques are:

1. Iterative Closest Points (ICP). As the algorithm has been constructed many years ago, it has been optimized and many variants have been constructed to improve the method. As this method can be seen as the foundation of spatial matching algorithms, it is chosen as a feasible technique in this research.
2. IK is chosen because it comes to an optimum faster than ICP, according to Pottmann et al. (2004) (and therefore should be computationally faster than ICP as well).
3. Hough Transform (HT) is chosen because the method that is constructed by Ni et al. (2013) is similar to method that is used in this thesis: it tries to register a 3D shape to a 2D reference shape. The ICP and IK algorithms on the other hand, are methods for 3D/3D or 2D/2D matching.

Not only three registration algorithms have been compared, but also two parameters that have effect on the registration. Two conditions have been identified to have an impact on the results of the algorithms. These have their (dis)advantages:

1. The use of all meshes (global) or the use of only the meshes that are mapped since the last registration. There are advantages to both methods:
    - Using the 'local' configuration, any drift that has happened before will not affect the registration, therefore possibly getting better (local) fits. On top of that, computation times will be faster, since less points are used for registration.
    - The 'global' configuration could be more robust, as it computes the registration over a bigger space.
2. The use of a buffer around the walls on the floor plan can affect the registration as well:
    - Using a buffer around walls can filter the artefacts scanned by the MH, such as people or furniture. This reduces noise in the registration and speeds the computation.

- If the fit of the initial (or previous) registration is off, using a buffer around the floor plan walls can remove many points that are actually on walls. That makes the case for computing a registration without a buffer.

8.1.2 Performance of algorithms in terms of accuracy and speed

The next question goes more in depth on the matching techniques: *What is the most promising spatial matching technique in terms of accuracy and speed?* A good trade-off between these two should be found: an algorithm that is too slow, cannot be used in the case of real-time indoor positioning. However, if the resulting position is inaccurate, this might even be worse if the MH carrier relies on that position.

No algorithm configuration resulted in an accuracy over the 15 experiments in all three buildings that is lower than the goal of $<$1 meter. However, there is one configuration that has been able to maintain an maximum error $<$5 meter and a mean accuracy of $<$1 meter over 12 out of 15 experiments. This result is achieved using a Hough Transform (HT), with a 0.5m buffer and meshes in proximity. Also, over all configurations, the Hough Transform (HT) algorithm has the least varying accuracy. However, the remaining three experiments have maximum errors between 45-113 meter. ICP and IK both have worse results in the Architecture building, but the results in the validation buildings are more similar.

A maximum duration of 15 seconds is defined as a feasible computation time. However, the only configurations that meet this requirement are the 'local' use of meshes configurations. It is not surprising that this parameter influences the computation time: not only is the size of the point cloud smaller, the floor plan area that is used in the registration is also smaller. On top of that, using a 0.5 meter buffer reduces the amount of points in the point cloud as well, also decreasing the computation times. Lastly, the fastest algorithm is the Hough Transform (HT) algorithm, probably because, contrary to IK and ICP, the HT algorithm is not iterative. Surprisingly, this implies that the most accurate configuration (HT / local registration / 0.5m buffer) is also the fastest. This configuration meets the computation time requirements, with an average of only 5 seconds per algorithm. The computation times are measured using the HP laptop, with much better processing power than the MH. Therefore, while it could be concluded that the general requirement of 15 seconds is met, this does not necessarily indicate that the computation time is below 15 seconds on the MH.

Concluding, the Hough Transform algorithm is able to meet the computation time requirements and is most accurate. Therefore it is the most promising spatial matching technique in comparison to the other two.

8.1.3 Suitability of the method for indoor positioning

The previous two questions concerned the specific spatial matching techniques. The following question concerns the positioning method as a whole: *How can the indoor positioning of the Microsoft Hololens be improved by making use of the researched positioning method?* We can compare the performance of the positioning method with the found initial transformation, but also with the 'perfect' transformation that is computed after the experiments, that is in essence the drift error of the MH.

A rotational error of only two degrees can give mean errors of around 2 meter, with the error at the last measured point up to 6.6 meter. A rotational error of nine degree gives errors of 10 meter, with at the last measured point even 18 meter. Comparing the algorithms with this accuracy resolves into the fact that even the 'worst' performing algorithm, ICP has a better positioning estimate than the initial transformation. Therefore, it could be said that the method is a suitable method for indoor positioning.

An accuracy below 1 meter was defined as a goal. None of the registration algorithms meets that goal. Only one configuration of one registration algorithm has an average accuracy below 1 meter (HT / local / 50cm buffer) in the case study. In the validation sets, in only 7/10 experiments, the same result could be achieved. The drift error of the MH is on average only 68 centimeter over a trajectory of more than 300 meter. This is less than we thought beforehand. Therefore, it should be possible to get a better result using registration than is achieved now, with use of more robust techniques and outlier detection.

8.1.4 Method failures

The last sub-question is: *In which cases does the researched positioning method fail to estimated the position on a 2D floor plan?* This seems to be more difficult to measure than we initially thought. The degenerate cases that were defined in the methodology have little to no impact on the performance of the registration, but other problems did occur. The limited impact of the degenerate cases on the performance of the registration can be explained by a number of reasons, changing per case:

- Big spaces without walls, such as large halls, did not change the behaviour of any of the algorithms. This has a simple reason: there was a threshold of 100 points that should be in the point cloud, to even perform the registration. As there are no walls within 3 meter of the device carrier in the middle of a big hall, there is no new registration. Therefore the positioning only relies on registrations done beforehand and the SLAM of the Hololens. The latter also worked well in the big spaces.

- There were some walls present in the floor plan, that were not present in real-life. However, this is methodologically the same as the un-mapped geometry that is on the floor plan, but has no 3D geometry yet. As the point cloud is registered to the floor plan and not the reversed, it did not bring any issues in the registration.

Incorrect registrations are rather due to other reasons. An issue showed up when walking through hallways in the case of IK. Despite having the best result, this method does not work when the reference lines all have the same orientation. This occurs when walking in a hallway. This problem is related to the maths of the registration algorithm: it makes use of the normals of the floor plan lines. If all these normals are the same, too many zeros occur in a system of equations that need to be solved. Resulting numbers can therefore be arbitrary, which leads to wrong registrations. As soon as normals that go in different directions are added, this problem does not occur anymore and a good fit could be found.

Not surprising, the quality of the scan affected the accuracy most. This became clear in the validation cases. Failures of the method are caused by general artefacts / noise in the point cloud, instead of the previously listed degenerate cases. Registrations incorrectly use points in the scan that in reality is furniture. This confused some of the registrations at start, especially with theICP algorithm. This algorithm is most susceptible for outliers. In the case of the office building, some algorithms found a fit that was rotated 90 degrees from reality and were not able to recover these errors in the remaining process. This resulted in huge errors > 100 meter. See-through instead of solid walls also were suggested as one of the main reasons why the registrations in the validation cases were worse than in the Architecture building. These are not scanned adequately by the MH and therefore influence the quality of the scan. More testing is needed to ground these conclusions.

8.1.5 General conclusions

The main research question to be answered is: *How can the Microsoft Hololens improve indoor positioning, using the on-the-fly produced mesh and an existing floor plan?* The developed method is able to position someone indoors, without any pre-installed infrastructure or communication from outside. The only needed information beforehand is a floor plan. This is a major advancement over previously existing methods in the case of emergency response, where the responders are not able to use existing techniques with Bluetooth or Wi-Fi. It works stand-alone, without any communication with other agents inside the building. An accuracy <1 meter is not achieved, but in 80% of the experiments and accuracy <5 meter is met, even in the case of multiple buildings. Other situations, such as the use of multiple floors should be tested to give a overarching view on the indoor positioning method using the Microsoft Hololens (MH).

8.2 Discussion

We have seen that there is a lot of potential in the implemented method. However, there is too much variation between the results in the configurations, to draw strong conclusions on the suitability of this positioning method in real-life. In order to make such a system available in practice, many aspects could be (and some need to be) improved. This discussion will elaborate on the limitations, but also the implications of this research, followed by some recommendations for future work.

The Microsoft Hololens (MH) SLAM was found to be working surprisingly well, with an accuracy of around 70cm during a 350 meter trajectory. This should be placed in context of other researches, that found different results: Landgraf (2018, in Hübner et al., 2019) has shown that drift effects visibly occur in large-scale buildings and Hübner et al. (2020) measured a total drift of 2.39 meter over a course of 287 meter. The positive results could be due to the building, that might be better suitable for the SLAM, a better calibration of the particular MH, or just pure chance. However, such a result could make the case for the use of manual, instead of automatic registration. If virtually no drift occurs, the position of the carrier only has to be set once, to be accurate over the whole trajectory. Instead of using one wall for the initial registration, one might use two, to come to a better initial fit. This takes more time, but might be more reliable, taking into account the minimal drift. However, if someone is using some other device than the MH with more drift, a spatial matching technique remains more desirable than a manual one.

Noisy point clouds are a well-known problem in the field of shape registration (Gelfand et al., 2003). In this case, the noise is caused by objects present in the scanned mesh, that are absent in the floor plan. Improving outlier detection and removal could have been a valuable addition. The Hough Transform algorithm could benefit from image processing literature on outlier removal in Hough images (see Ni et al., 2013). Also in the case of ICP, outlier detection methods are already developed as well. These can also be used in the case of IK. For various outlier detection methods see Rusinkiewicz and Levoy (2001).

Serious optimization needs to be done to really get the method working during a computation time of 15 seconds per registration. The illustrated computation times are the result of computing the registration in the simulation program, on the computer. The computer has a better processor and more cores than the MH. Moreover, the MH needs to compute its SLAM simultaneously, which also takes processing power. This could be achieved, by optimizing the registration algorithms. While the ICP is implemented with a number of features that optimize the algorithm, IK and HT do not. However, all optimizations that are present in the ICP algorithm, are in theory also applicable to the IK algorithm. Therefore, there are some ways to speed up the IK algorithm. In the case of HT it is somewhat more difficult. The translational component of HT is similar to the ICP algorithm, so also that part could be optimized as well using the same techniques. The rotational component is now found by iterating over both the reference data and the measured data. Using calculus, there might be more efficient solutions than for-loops. For example, converting the rotation to a Taylor expansion and finding the phase difference between the reference and measured points.

The results described in sec 8.1.3 should be placed in context of other positioning methods that do not make use of present infrastructure. Both IK and HT have similar to better results as the multi-agent multi-sensor approach of (Rantakokko et al., 2011). One additional advantage over the AR approach, is the lack of need for multiple agents, something that Rantakokko et al. (2011) and Nilsson et al. (2014) require.

8.2.1 Strengths and limitations

A strength of this thesis is the generalizability of the method. It should function in circumstances that meet the following two requirements: (1) There is some spatial reference data. (2) there is a SLAM algorithm, or something similar that maps the environment, finds a position in it and could communicate both in real-time. While floor plans are used in this case, the method is not limited to that. There is only

limited adaptation needed for a 3D variant. The system will work on any AR device that meets requirement (2). This is valuable, since the MH is not able to work in harsh environments. It is not resistant to the hot or humid places that a emergency responder is regularly working in.

One building has been selected as case study and two buildings have been used as validation cases. The results of the latter were found to be considerably worse than for the main study. Consequently, it is not possible to generalize the conclusions to multiple buildings. While suggestions have been made why the positioning method fails to be accurate in some cases, there is no proof of these. It is suggested that the presence of glass walls and a lot of furniture are detrimental for the method. Limited by time restrictions and Covid-19 measures, additional research on this matter has not been carried out. To make any further conclusions, more research is needed into the different typologies of buildings and how the positioning method behaves in these circumstances.

8.2.2 Implications

This thesis shows that it is possible to do indoor positioning without any use of infrastructure or communication between multiple agents, using SLAM. While this has been speculated by other researches (Rantakokko et al., 2011; Nilsson et al., 2014; Fonnet et al., 2017; Alshawa, 2007), to our knowledge this has not been implemented and proved. It opens the door for new research to improve on this positioning method. If the indoor positioning accuracy can be further improved, by for example better outlier removal, it makes way for navigation apps as well. Not only is positioning with the MH useful in the case of ER, but also other use cases have been found. Fonnet et al. (2017); Alshawa (2007) found their use case in the renovation of heritage. Adjustments that should be made to a building can be measured and annotated in 3D and immediately be related to a virtual 3D model (or floor plan).

This thesis takes Emergency Response (ER) as its starting position. Even taking aside the ER requirement that the positioning accuracy needs to be below one meter at all times, there are limited implications for the case of ER, by cause of a number of reasons:

- The MH is a fragile device. It cannot stand water, nor can it stand heat. It will fail in the case of smoke development, as the SLAM is based on RGB-D cameras, that do not work in smoke. Therefore, there is limited use of the device in ER. That is also the reason why this thesis is set up to work dynamically with the data. But the limitation remains that the method cannot be used directly, because of the MH.
- Currently, there is limited support of data formats for the reference floor plan. Manual work has been done to filter an AutoCAD file on its walls. This is not possible in the limited time frame that exists within ER. Best would be to have an integration with online datasets that are currently in use by the emergency responders. This could for example be queried using WFS statements, creating an automated pipeline and resolving this issue.
- The initial transformation requires manual work: selecting a wall on the floor plan / in real-life. A suggestion will be done in section 8.2.3 that could resolve this limitation.
- A real-time positioning accuracy estimator has not been found. The RMSE did not estimate the accuracy adequately.

8.2.3 Future work

This research proposed a method to position indoors with the use of only the MH and a floor plan. Some future work could be identified:

- Now the positioning with the MH is possible, this gives possibilities for integrating it with navigation apps as well. It is interesting to research the added value of AR to navigation. New use cases can be found here as well: visual impaired persons could use the MH and audio-based navigation to navigate in indoor spaces. Such a system can be integrated with already existing research on this subject, by i.e.Yamashita et al. (2017).

- Due to the limited availability of 3D models in ER, it is chosen to work with floor plans. It would be useful to compare the results in this thesis to the results using a BIM. Are there major differences using a BIM instead of floor plan? Is the position estimate better when using 3D data?
- The use of more than one floor in a building has been declared out of scope in this thesis. A recommendation for future work is to research the use of multiple floors and more complex forms. What if buildings have an entresol? Or slopes? Is it even possible to use floor plans for complex building types? Or should a 3D BIM be used? These questions remain to be answered.
- It has been mentioned a couple of times: a connection with country-wide system with indoor floor plans for public buildings, where the area of impact can be imported as reference file would greatly increase the usability of an indoor positioning system for ER. One recommendation for future research is to explore if such a system is achievable.
- The initial transformations is now computed with some manual input. In the future, a better method could be constructed. Automated door detection is a possibility. It is possible to detect doors in (voxelized) point clouds (Nikoohemat, 2016). If the door that a rescue team enters is known, one could select this door beforehand. When the team enters the building and the device detects a door, the coordinates could be matched to the floor plan.
- Finally, recently there has been work done in the direction of situational awareness in Emergency Response, using the Microsoft Hololens (Smit, 2020). This type of system can communicate the position inside the mapped 3D model of the MH to officers and other responders outside of the building, together with necessary information. Such a system can be enhanced by a position that is relative to a position on a floor plan of that building, for even better understanding of the situation. In this manner, it is not only known where the responders have been, but also how that relates to the whole size of the building, how far responders are from each other et cetera.

8.3 Epilogue

This thesis created as part of the study programme of the master Geomatics of the Built Environment. Therefore, I'd like to reflect on the project in the context of the master studies as a whole. As a result of doing additional courses and a second degree, the master took me a total of four years, instead of the intended duration of two years. The courses in the first year are a while ago, but I did not lose the connection with the master. If anything, doing a social science degree 'in between' could even give a clearer view on the master. For me, the master of Geomatics not only forms an academic ethos, but also educates some kind of spatial mindset. The study has a very practical programme: it tries to give hands-on tools with clear applications. Courses teach you python, databases or GIS. But at the end of the studies you are not just an expert in some spatial tool set, but someone that can solve real-world problems in a spatial way. A specialist that is able to look into complex problems in innovative manners and visualize these. I think this thesis is an example of this mindset. While the basis of the method, shape registration, is very familiar for a Geomatics student, the tools used are completely different than conventional: Unity's core application is game design and therefore functions different than GIS or BIM programs. The created application does have some affiliation with mini-maps that often are seen in games.

Indoor positioning is a large discipline within Geomatics. Many of my fellow students already graduated on a similar topic and I am grateful to be the next to add new knowledge to this study field. My true interests lie in the field of 3D analysis and modelling. Therefore I was delighted that I could combine these two fields in one research. On top of that, it was exciting to work with the Microsoft Hololens. It is not (yet) part of the most conventional devices used in the Geomatics field, as the courses focused on data retrieved from drones, satellite imagery or stationary laser scanners. Nonetheless, AR is a perfect example of how Geomatics is integrated into emerging technologies, together with other hot topics like driverless cars. Spatial understanding is detrimental for many systems and I hope this research pushed the knowledge boundaries on this matter a tiny bit further.

Bibliography

Alshawa, M. (2007). ICL: iterative closest line - a novel point cloud registration algorithm based on linear features. *Ekscentrar*, 10(10):53–59.

Bahl, P. and Padmanabhan, V. N. (2000). Radar: An in-building rf-based user location and tracking system. In *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No. 00CH37064)*, volume 2, pages 775–784. Ieee.

Bellekens, B., Spruyt, V., Berkvens, R., Penne, R., and Weyn, M. (2015). A Benchmark Survey of Rigid 3D Point Cloud Registration Algorithms. *International Journal on Advances in Intelligent Systems*, 8(1-2):118–127.

Besl, P. J. and McKay, N. D. (1992). Method for registration of 3-D shapes. pages 586–606, Boston, MA.

Birdal, T. (2020). ICP Registration using Efficient Variants and Multi-Resolution Scheme. Library Catalog: nl.mathworks.com.

Brito, C., Alves, N., Magalhães, L., and Guevara, M. (2019). BIM Mixed Reality Tool for the Inspeciation of heritage buildings. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W6:25–29.

Chatzopoulos, D., Bermejo, C., Huang, Z., and Hui, P. (2017). Mobile Augmented Reality Survey: From Where We Are to Where We Go. *IEEE Access*, 5:6917–6950.

Deak, G., Curran, K., and Condell, J. (2012). A survey of active and passive indoor localisation systems. *Computer Communications*, 35(16):1939–1954.

Dilo, A. and Zlatanova, S. (2011). A data model for operational and situational information in emergency response: The Dutch case. *Applied Geomatics*, 3(4):207–218.

Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localization and mapping: part I. *IEEE Robotics & Automation Magazine*, 13(2):99–110.

Elbaz, G., Avraham, T., and Fischer, A. (2017). 3D Point Cloud Registration for Localization Using a Deep Neural Network Auto-Encoder. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2472–2481, Honolulu, HI. IEEE.

Fonnet, A., Alves, N., Sousa, N., Guevara, M., and Magalhaes, L. (2017). Heritage BIM integration with mixed reality for building preventive maintenance. In *2017 24º Encontro Português de Computação Gráfica e Interação (EPCGI)*, pages 1–7, Guimaraes. IEEE.

Gelfand, N., Ikemoto, L., Rusinkiewicz, S., and Levoy, M. (2003). Geometrically stable sampling for the ICP algorithm. In *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings.*, pages 260–267, Banff, Alberta, Canada. IEEE.

Hightower, J. and Borriello, G. (2001). Location systems for ubiquitous computing. *computer*, 34(8):57–66.

Hübner, P., Clintworth, K., Liu, Q., Weinmann, M., and Wursthorn, S. (2020). Evaluation of HoloLens Tracking and Depth Sensing for Indoor Mapping Applications. *Sensors*, 20(4):1021.

Hübner, P., Landgraf, S., Weinmann, M., and Wursthorn, S. (2019). Evaluation of the Microsoft HoloLens for the Mapping of Indoor Building Environments. pages 44–53.

Hübner, P., Weinmann, M., and Wursthorn, S. (2018). Marker-based Localization of Microsoft Hololens in building models. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-1:195–202.

Jost, T. and Hugli, H. (2003). A multi-resolution ICP with heuristic closest point search for fast and robust 3D registration of range images. In *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings.*, pages 427–433, Banff, Alberta, Canada. IEEE.

Khoshelham, K., Tran, H., and Acharya, D. (2019). Indoor Mapping Eyewear: Geometric evaluation of spatial mapping capability of Hololens. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W13:805–810.

Kim, S. K., Kang, S.-J., Choi, Y.-J., Choi, M.-H., and Hong, M. (2017). Augmented-Reality Survey: from Concept to Application. *KSII Transactions on Internet and Information Systems*, 11(2).

Klein, G. and Murray, D. (2007). Parallel Tracking and Mapping for Small AR Workspaces. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–10, Nara, Japan. IEEE.

Kwan, M.-P. and Lee, J. (2005). Emergency response after 9/11: the potential of real-time 3D GIS for quick emergency response in micro-spatial environments. *Computers, Environment and Urban Systems*, 29(2):93–113.

Li, A., Ruan, X., Huang, J., Zhu, X., and Wang, F. (2019a). Review of vision-based Simultaneous Localization and Mapping. In *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, pages 117–123, Chengdu, China. IEEE.

Li, J., Gao, X., and Meng, J. (2019b). An accelerated ICP registration algorithm for 3d point cloud data. In Ma, X., Wu, F., Fan, B., Li, X., and Zhang, Y., editors, *9th International Symposium on Advanced Optical Manufacturing and Testing Technologies: Optical Test, Measurement Technology, and Equipment*, page 3, Chengdu, China. SPIE.

Low, K.-L. (2004). Linear Least-Squares Optimization for Point-to-Plane ICP Surface Registration. Technical Report TR04-004, Department of Computer Science, University of North Carolina, Chapel Hill.

Magic Leap, I. (2020). Magic Leap 1. Library Catalog: www.magicleap.com.

Makadia, A., Patterson, A., and Daniilidis, K. (2006). Fully Automatic Registration of 3D Point Clouds. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1 (CVPR'06)*, volume 1, pages 1297–1304, New York, NY, USA. IEEE.

Manuri, F. and Sanna, A. (2016). A Survey on Applications of Augmented Reality. 5(1):10.

Microsoft (2019). HoloLens (1st gen) hardware. Library Catalog: docs.microsoft.com.

Molina, B., Olivares, E., Palau, C. E., and Esteve, M. (2018). A Multimodal Fingerprint-Based Indoor Positioning System for Airports. *IEEE Access*, 6:10092–10106. Conference Name: IEEE Access.

Newcombe, R., Izadi, S., Molyneaux, D., Hilliges, O., Kim, D., Shotton, J. D. J., Kohli, P., Fitzgibbon, A., Hodges, S. E., and Butler, D. A. (2014). Mobile Camera Localization Using Depth Maps.

Ni, K., Armstrong-Crews, N., and Sawyer, S. (2013). Geo-registering 3d point clouds to 2d maps with scan matching and the Hough Transform. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1864–1868, Vancouver, BC, Canada. IEEE.

Nikoohemat, S. (2016). Permanent structure detection in cluttered point clouds from indoor mobile laser scanners (imls).

Nilsson, J.-O., Rantakokko, J., Handel, P., Skog, I., Ohlsson, M., and Hari, K. (2014). Accurate indoor positioning of firefighters using dual foot-mounted inertial sensors and inter-agent ranging. In *2014 IEEE/ION Position, Location and Navigation Symposium - PLANS 2014*, pages 631–636, Monterey, CA. IEEE.

nreal (2020). nreal. Library Catalog: developer.nreal.ai.

Pottmann, H., Leopoldseder, S., and Hofer, M. (2004). Registration without ICP. *Computer Vision and Image Understanding*, 95(1):54–71.

Ramadhanakbr (2016). Microsoft Hololens https://commons.wikimedia.org/wiki/File:Ramahololens.jpg.

Rantakokko, J., Handel, P., Fredholm, M., and Marsten-Eklof, F. (2010). User requirements for localization and tracking technology: A survey of mission-specific needs and constraints. In *2010 International Conference on Indoor Positioning and Indoor Navigation*, pages 1–9, Zurich, Switzerland. IEEE.

Rantakokko, J., Rydell, J., Strömbäck, P., Händel, P., Callmer, J., Törnqvist, D., Gustafsson, F., Jobs, M., and Grudén, M. (2011). Accurate and reliable soldier and first responder indoor positioning: multisensor systems and cooperative localization. *IEEE Wireless Communications*, 18(2):10–18.

Rubino, I., Xhembulla, J., Martina, A., Bottino, A., and Malnati, G. (2013). MusA: Using Indoor Positioning and Navigation to Enhance Cultural Experiences in a Museum. *Sensors*, 13(12):17445–17471. Number: 12 Publisher: Multidisciplinary Digital Publishing Institute.

Rusinkiewicz, S. and Levoy, M. (2001). Efficient variants of the ICP algorithm. In *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, pages 145–152, Quebec City, Que., Canada. IEEE Comput. Soc.

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

Smit, B.-P. (2020). *Creating Remote Situation Awareness of Indoor First Responder Operations using SLAM*. PhD thesis, TU Delft, Delft.

Snoeren, G., Zlatanova, S., Crompvoets, J., and Scholten, H. (2007). Spatial Data Infrastructure for emergency management: the view of the users. pages 1–12.

Tang, Y., Au, K., and Leung, Y. (2018). Comprehending products with mixed reality: Geometric relationships and creativity. *International Journal of Engineering Business Management*, 10:1847979018809599.

Tashakkori, H., Rajabifard, A., and Kalantari, M. (2015). A new 3d indoor/outdoor spatial model for indoor emergency response facilitation. *Building and Environment*, 89:170–182.

van Capelleveen, E., van Duijn, A., Smit, J., Broekhaar, M., van Wanrooij, M., Zijlstra, G., and Geurts, P. (2008). Digitale Bereikbaarheidskaart. page 47.

van der Meer, T. (2016). 3d-indoormodellen voor de brandweer – een verkennend onderzoek naar de toepassingsmogelijkheden binnen de veiligheidsketen. Technical report, TU Delft.

van der Meer, T. (2018). *Geovisualization for the Dutch fire brigade. A research about effective cartographic methods for assisting tactics choice and indoor deployments during building fires.* PhD thesis, Utrecht University, Utrecht.

van der Meer, T. W. T., Verbree, E., and van Oosterom, P. J. M. (2018). Effective cartographic methods for assisting tactics choice and indoor deployments during building fires- a case study of the Dutch Fire Brigade. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-4:655–660.

Yamashita, A., Sato, K., Sato, S., and Matsubayashi, K. (2017). Pedestrian Navigation System for Visually Impaired People Using HoloLens and RFID. In *2017 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, pages 130–135. ISSN: 2376-6824.

Yang, C. and Shao, H.-r. (2015). WiFi-based indoor positioning. *IEEE Communications Magazine*, 53(3):150–157.

Zinsser, T., Schmidt, J., and Niemann, H. (2003). A refined ICP algorithm for robust 3-D correspondence estimation. In *Proceedings 2003 International Conference on Image Processing (Cat. No.03CH37429)*, volume 3, pages II–695–8, Barcelona, Spain. IEEE.

Zlatanova, S. (2008). SII for Emergency Response: The 3D Challenges. *The international Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVII(B4):1631–1638.

Zlatanova, S. and Fabbri, A. G. (2009). Geo-ICT for Risk and Disaster Management. In Sui, D. Z., Tietze, W., Claval, P., Gradus, Y., Park, S. O., van der Wusten, H., Scholten, H. J., van de Velde, R., and van Manen, N., editors, *Geospatial Technology and the Role of Location in Science*, volume 96, pages 239–266. Springer Netherlands, Dordrecht.

Zlatanova, S. and Holweg, D. (2004). 3D Geo-Information in Emergency Response: A framework. pages 1–6.

Zlatanova, S., Vosselman, G., Koshelham, K., Peters, R., Beers, B., Voûte, R., and Djurrema, H. (2015). Slimme 3D Indoormodellen ter Ondersteuning van Crisismanagement in Grote Openbare Gebouwen. In *Informatievoorziening bij Veiligheidsregio's: Best Practices*, Institutional Repository, pages 213–221.

A Reproducibility self-assessment

A.1 Marks for each of the criteria

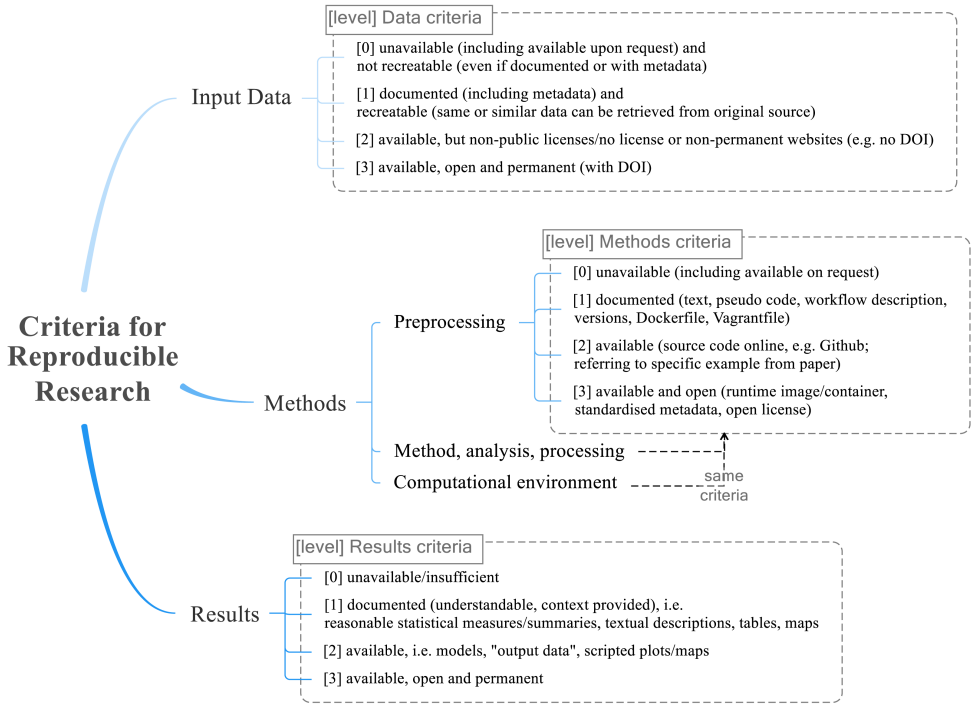


Figure A.1: Reproducibility criteria to be assessed.

Grade/evaluate yourself for the 5 criteria (giving 0/1/2/3 for each):

1. input data 1
2. preprocessing 3
3. methods 3
4. computational environment 1
5. results 2

A.2 Self-reflection

- This thesis uses two types of input data: floor plans and the Hololens scans. The floor plans are AutoCAD files, that are provided by the concerned parties. These might be requested by the source, but I am not allowed to share the CAD files myself. The Hololens data is saved into a format that is not easy to interpret. I also did not ask any of the parties permission to share the 3D data of their buildings. However, one could make a similar system and do the same scan another time.

- Preprocessing of the data has been described very clearly. The exact output of the floor plan after
- The methods are described using both pseudo-code and equations. That gives a lot of insight into the methods and therefore they should be easily reproducible.
- I did not share the code of the project, mainly because the pseudo-code gives a lot more insights into the method than the code itself, as it is long and full of unused functions and references. However, the structure of the application is described well in the Implementation section, together with how it looks like.
- All results are described with figures, graphs and tables. Additional figures are shown in the appendices. A CSV with the results is sent along with the thesis.

B Additional maps for results TU Library and CGI

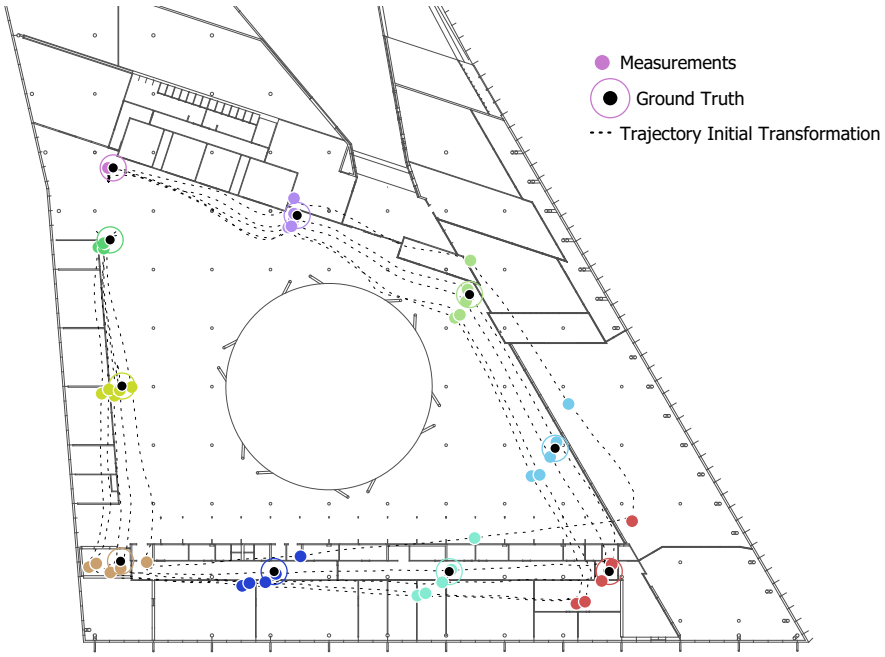


Figure B.1: The trajectories of the five experiments in the Library building after the initial transformation are shown in this figure. Every colour corresponds to one ground truth point. The coloured circles with the black dot is the ground truth, while the coloured dots are the measured points from one of the five experiments. The black line is the estimated trajectory.

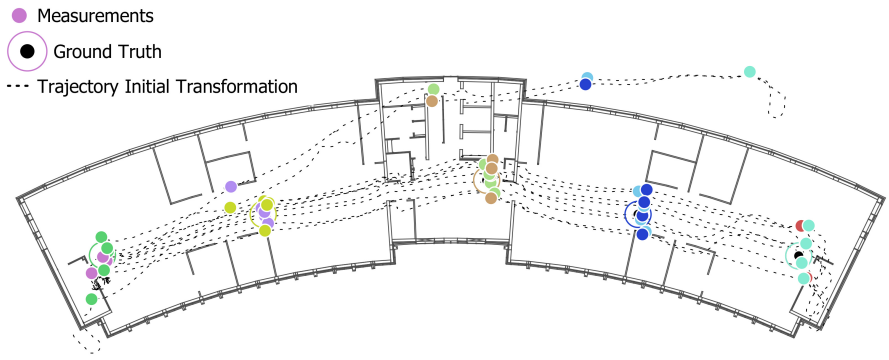


Figure B.2: The trajectories of the five experiments in the Library building after the initial transformation are shown in this figure. Every colour corresponds to one ground truth point. The coloured circles with the black dot is the ground truth, while the coloured dots are the measured points from one of the five experiments. The black line is the estimated trajectory.

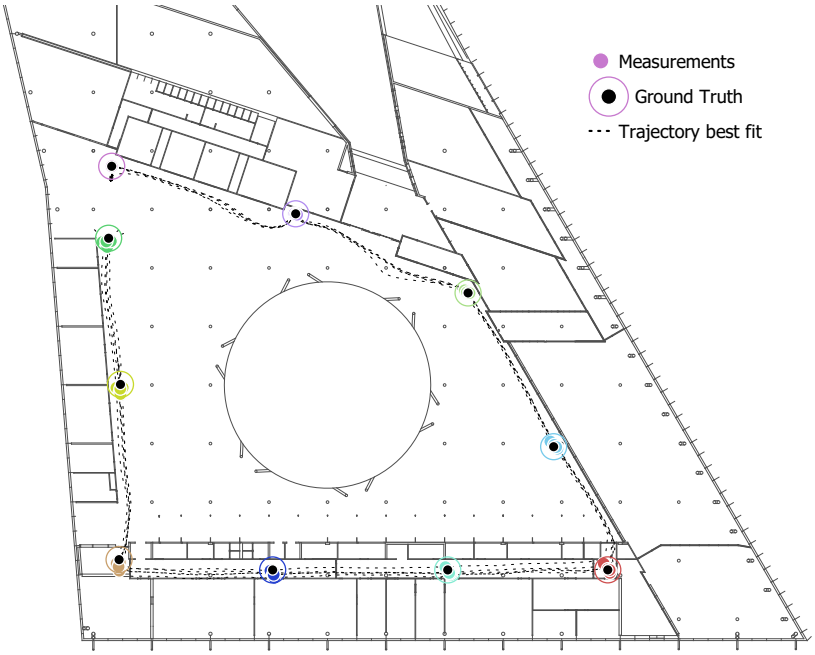


Figure B.3: The trajectories of the five experiments in the Library building after the applying SVD are shown in this figure. Every colour corresponds to one ground truth point. The coloured circles with the black dot is the ground truth, while the coloured dots are the measured points from one of the five experiments. The black line is the estimated trajectory.

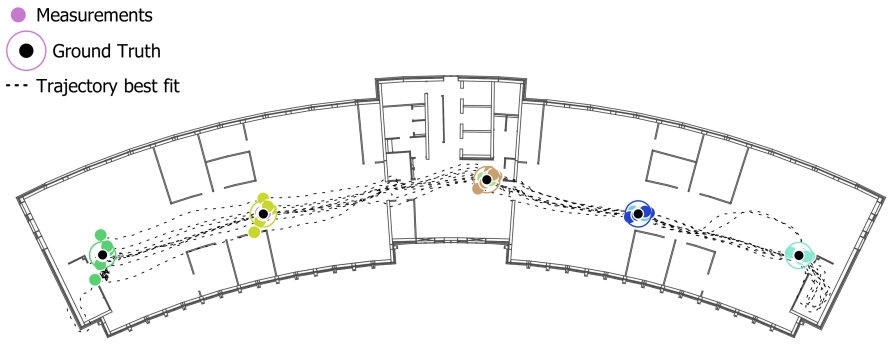


Figure B.4: The trajectories of the five experiments in the CGI building after the applying SVD are shown in this figure. Every colour corresponds to one ground truth point. The coloured circles with the black dot is the ground truth, while the coloured dots are the measured points from one of the five experiments. The black line is the estimated trajectory.

C UML Diagrams

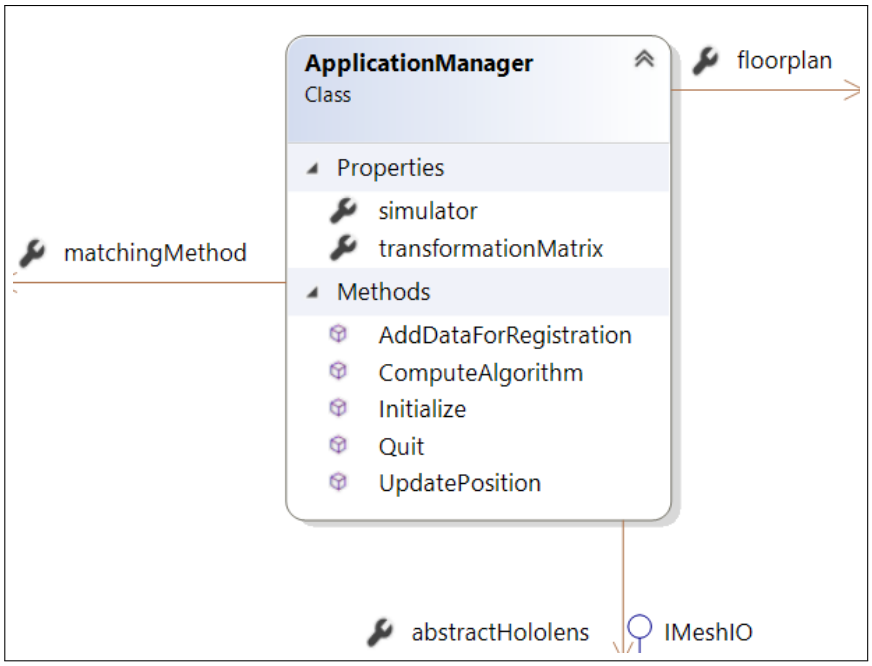


Figure C.1: Application manager UML

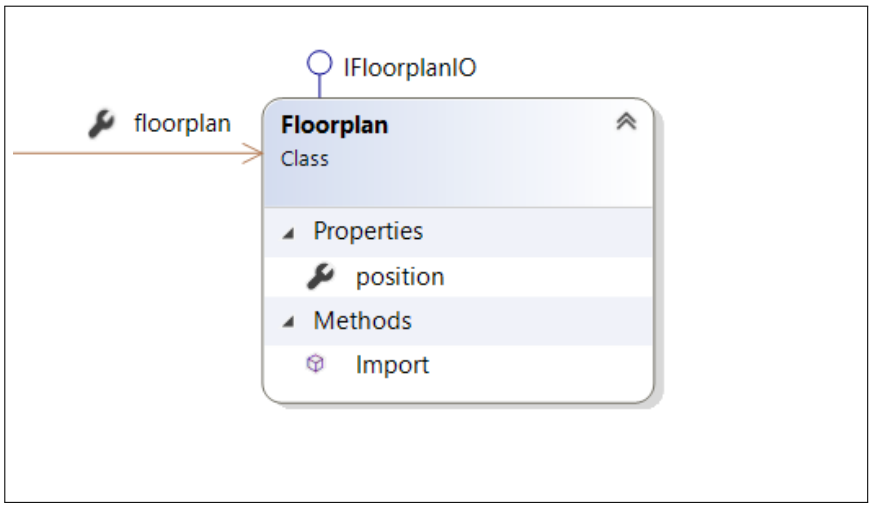


Figure C.2: Floorplan UML

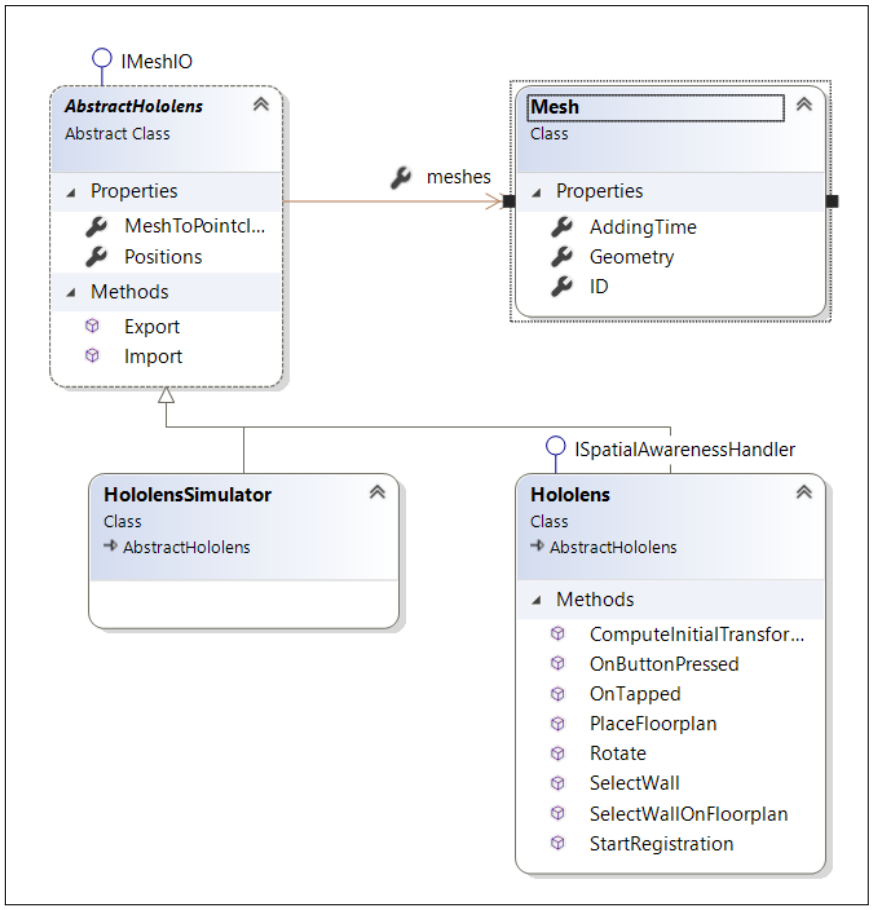


Figure C.3: Hololens UML

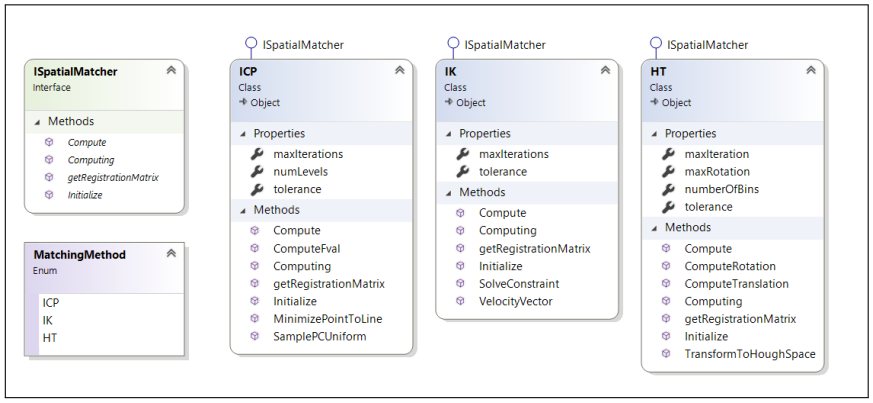


Figure C.4: Spatial matchers UML

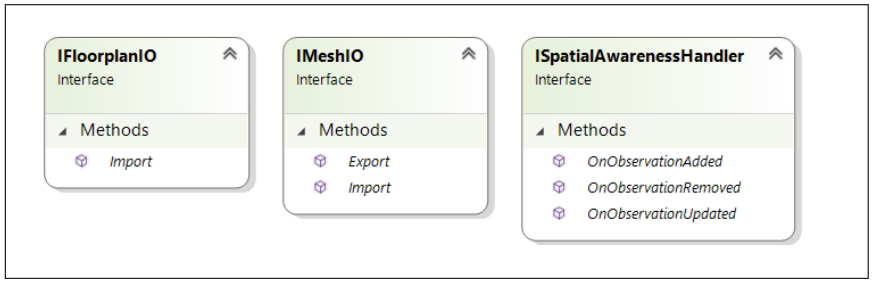


Figure C.5: interfaces UML

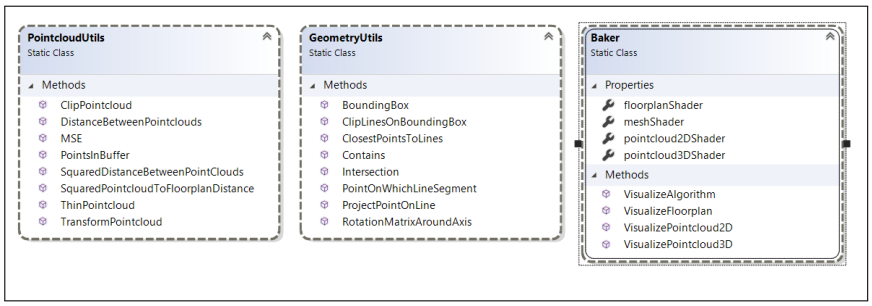


Figure C.6: Utility functions UML

D Iterative Closest Points maps

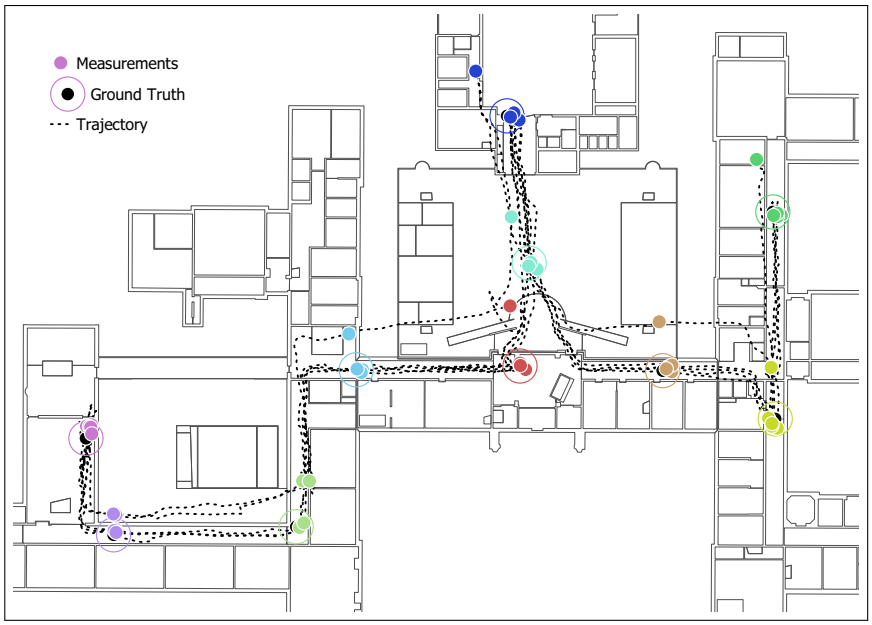


Figure D.1: ICP, configuration: global meshes without buffer

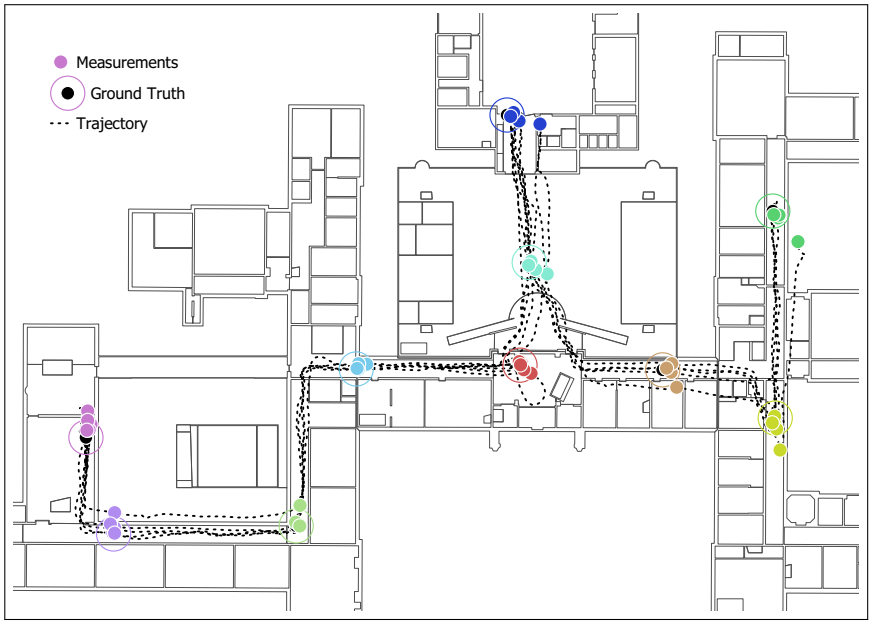


Figure D.2: ICP, configuration: global meshes and 50cm buffer

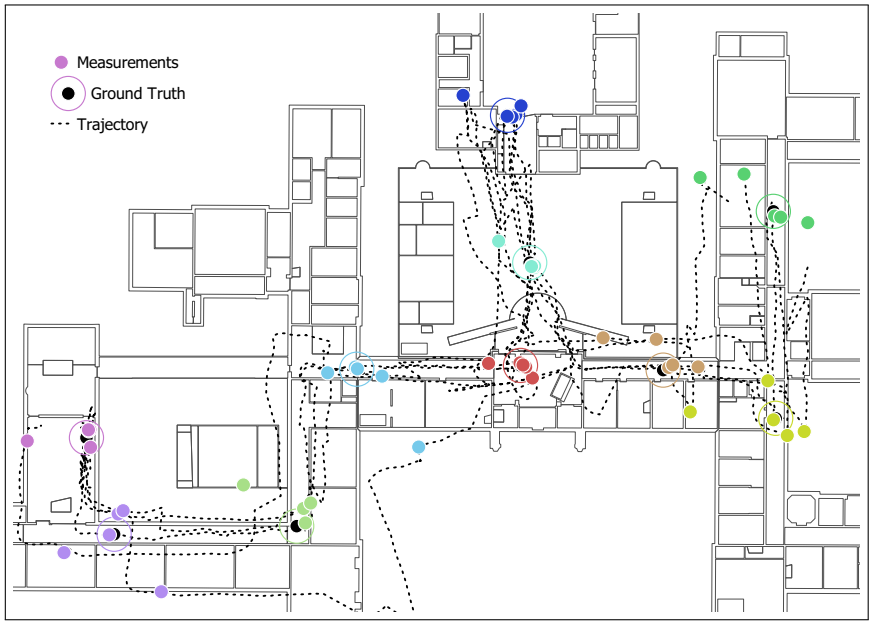


Figure D.3: ICP, configuration: local meshes without buffer

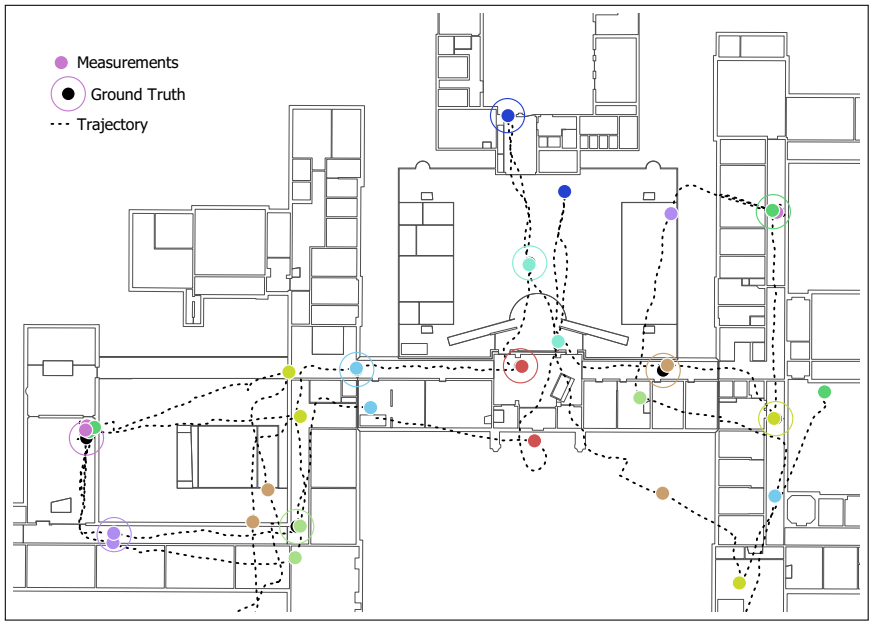


Figure D.4: ICP, configuration: local meshes and 50cm buffer

E Instantaneous Kinematics maps

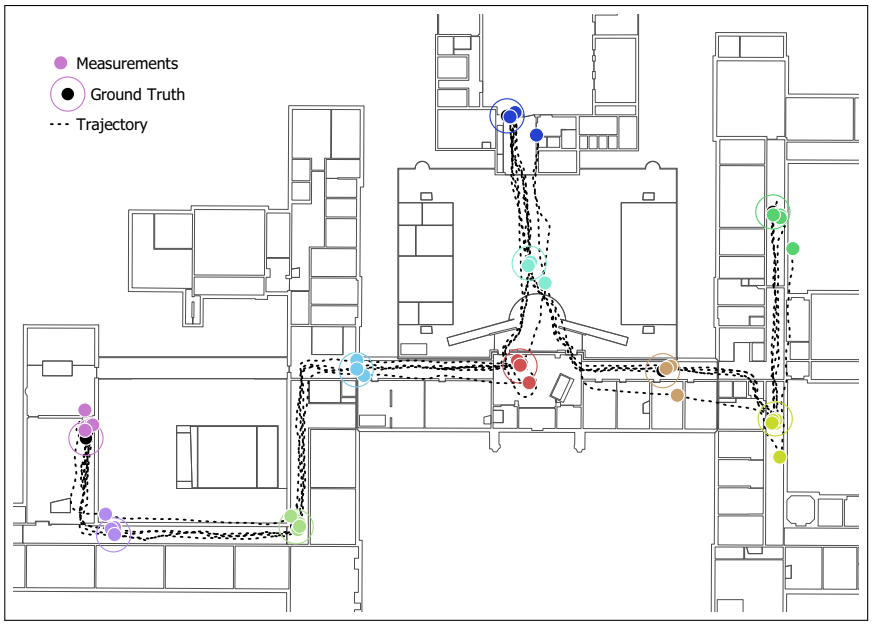


Figure E.1: IK, configuration: global meshes without buffer

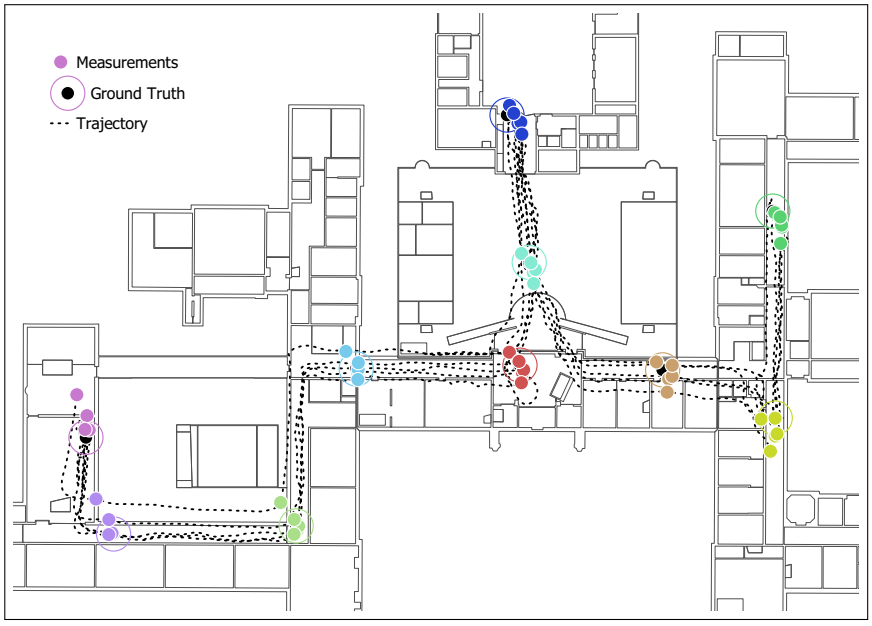


Figure E.2: IK, configuration: global meshes and 50cm buffer

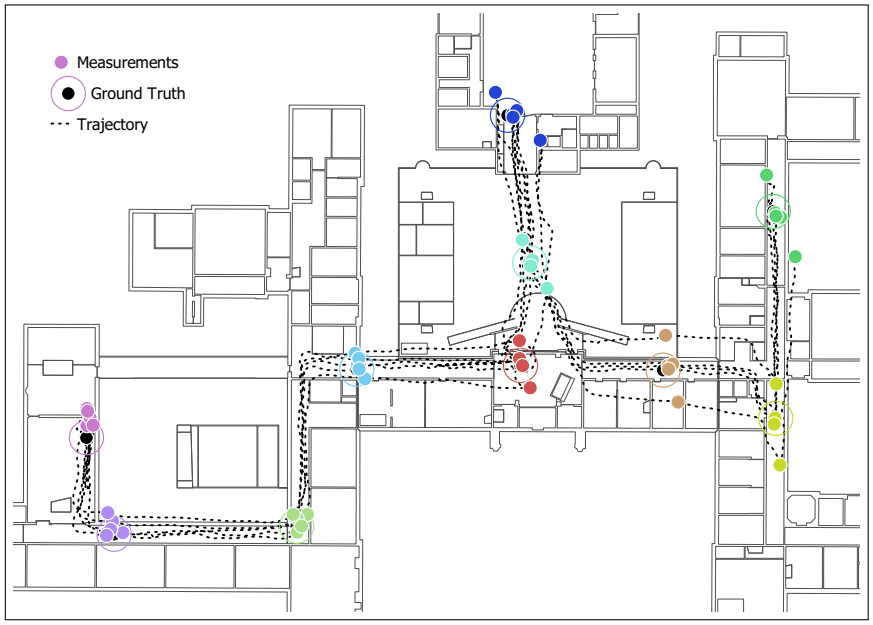


Figure E.3: IK, configuration: local meshes without buffer

Figure E.4: IK, configuration: local meshes and 50cm buffer

F Hough Transform maps

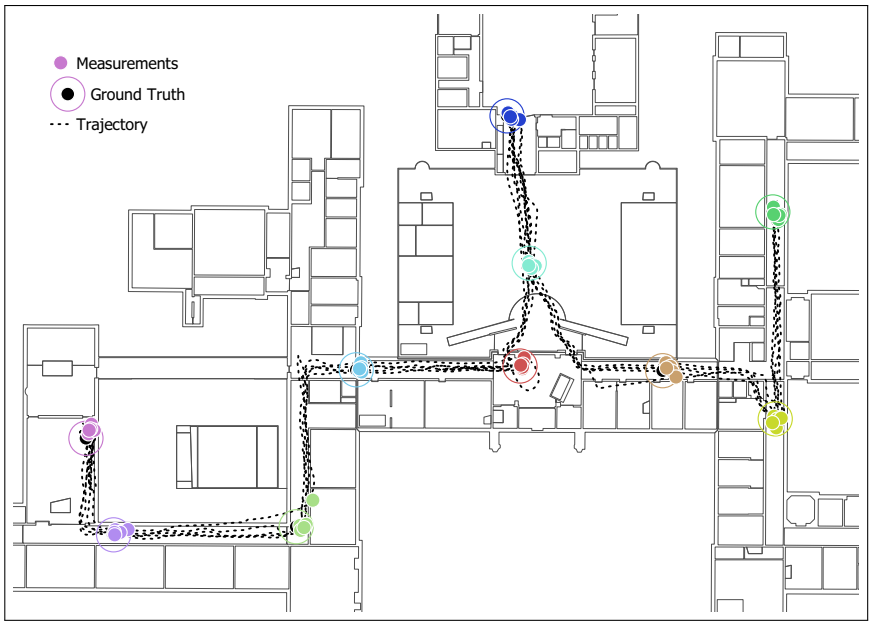


Figure F.1: HT, configuration: global meshes without buffer

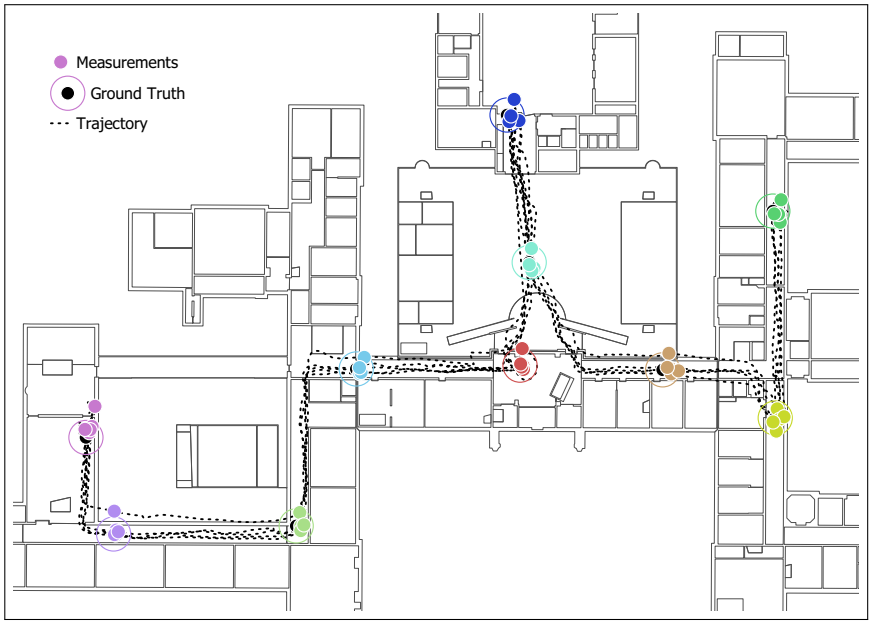


Figure F.2: HT, configuration: global meshes and 50cm buffer

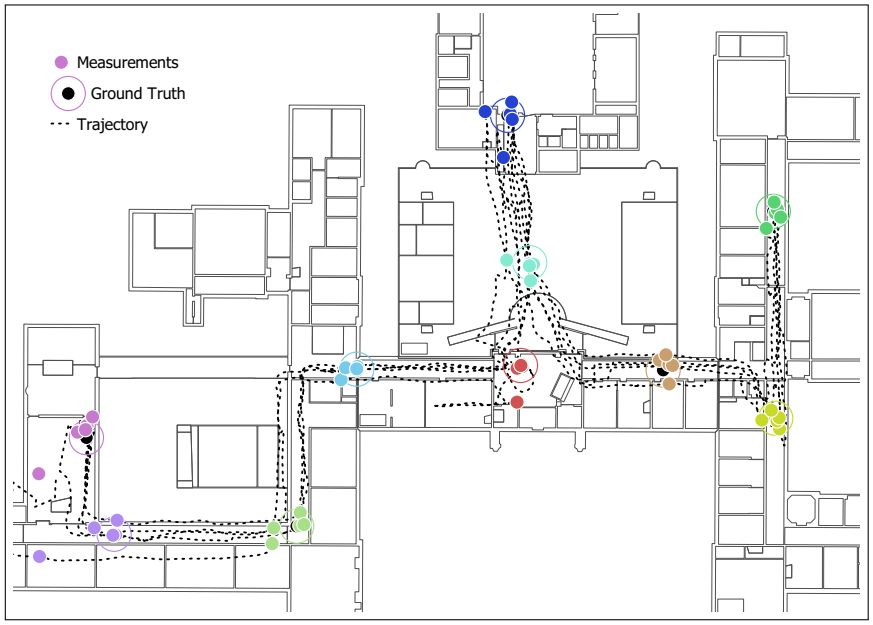


Figure F.3: HT, configuration: local meshes without buffer

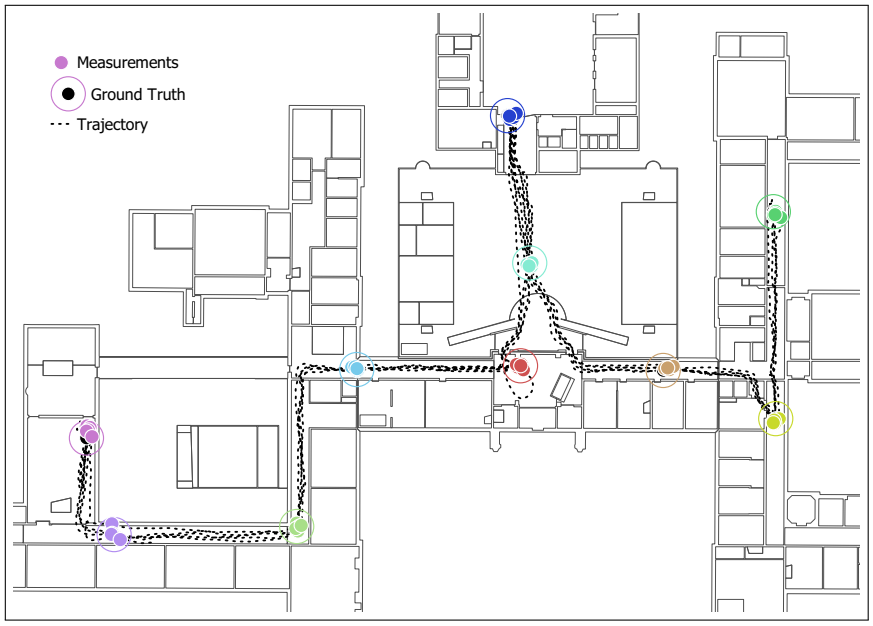


Figure F.4: HT, configuration: local meshes and 50cm buffer

Colophon

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