FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Path Planning in Augmented Reality Indoor Environments

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Media Informatics and Visual Computing

by

Karl Platzer, BSc
Registration Number 0626768

to the Faculty of Informatics

at the TU Wien

Advisor: Assoc. Prof. Dr. Hannes Kaufmann
Assistance: Dipl.-Ing. Mag. Georg Gerstweiler

Vienna, 12th December, 2017

Karl Platzer	Hannes Kaufmann

Erklärung zur Verfassung der Arbeit

Karl Platzer, BSc
Meravigliagasse 1/9, 1060 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 12. Dezember 2017

Karl Platzer

Kurzfassung

Legt man einen virtuellen Pfad auf die reale Umgebung innerhalb eines Gebäudes, kann sich eine Person darin besser zurechtfinden und somit schneller an den gewünschten Ort gelangen. Mit der Veröffentlichung von Augmented Reality (AR) Bibliotheken für Android- und Apple-Smartphones und der Verfügbarkeit von Head Mounted Display (HMD), erlangt AR immer mehr Aufmerksamkeit am Massenmarkt. Der größte Nachteil von modernen AR-Visualisierungsgeräten ist jedoch das kleine Sichtfeld. Als Sichtfeld bezeichnet man den Bereich der Umgebung, der von der Kamera eines mobilen Geräts erfasst und auf dem AR-fähigen Gerät dargestellt wird. Dies korrespondiert mit dem Bereich, in dem virtuelle Objekte über die reale Umgebung gelegt werden können. Den berechneten Pfad innerhalb des Sichtfeldes des AR-fähigen Geräts zu halten, ist der besondere Ansatz dieser Masterarbeit. Der Pfad sollte sich an der Blickrichtung des mobilen Gerätes ausrichten, sodass dem Benutzer durchgehend ein Weg angezeigt wird. Die vorliegende Arbeit präsentiert das Design und die Implementierung des Field of View (FOV) assisted path planning Algorithmus. Ein 3D-Modell des Inneren eines Gebäudes und Lokalisierungsdaten der Benutzer werden hierzu verwendet, um den FOV assisted path zu berechnen. Die begehbaren Bereiche der Umgebung werden aus dem 3D-Modell der Umgebung berechnet. Begehbare Bereiche beschreiben all jene Bereiche, die ein Benutzer auch wirklich begehen kann. Die begehbaren Bereiche werden in einem sogenannten Navigation Mesh dargestellt. Pfade werden nur innerhalb begehbarer Bereiche berechnet. Die Erzeugung eines Navigation Mesh ist nicht Teil der vorliegenden Masterarbeit. Diese Arbeit setzt die Recast Library von Mononen zur Navigation Mesh-Generierung ein. Für den implementierten Pfadfindungsalgorithmus werden Positions- und Orientierungsdaten der mobilen Endgeräte aufgezeichnet und bereitgestellt. Der berechnete Pfad besteht aus einem Pfad außerhalb und innerhalb des Sichtfelds des mobilen Geräts. Das Hauptaugenmerk der vorliegenden Masterarbeit liegt darauf, den Pfad innerhalb des Sichtfeldbereichs zu berechnen und ihn mit dem Pfad außerhalb des Sichtfelds zu verknüpfen. Im Zuge der Forschung wurden dazu verschiedene Strategien zum Berechnen und Verknüpfen der Teilpfade erarbeitet. Abhängig von der Blickrichtung werden verschiedene Pfadberechnungsmethoden und Spezialfälle implementiert und die daraus entstehenden Teilpfade miteinander verbunden. Außerdem sind im Pfadplanungsprozess dynamische Objekte, also Objekte die nicht im 3D-Modell des Gebäudes inkludiert sind, berücksichtigt. In dieser Arbeit wird ein laufzeiteffizienter Ansatz zur Einbettung von dynamischen Objekten in das statische Navigation Mesh einer 3D-Umgebung vorgestellt. Ein anderer Fokus liegt

hier auf einem für den Benutzer intuitiv verfolgbaren Pfad. Dafür wurde der FOV assisted path mit ausreichend Abstand zu Wänden und Objekten berechnet. Weiters wird der Pfad am Ende des Algorithmus geglättet, um einen anschaulicheren Pfad zu visualisieren. Zur Evaluierung der Performance sowie der Benutzerfreundlichkeit des implementierten Algorithmus wurde eine Performance-Analyse und eine Benutzerstudie durchgeführt. 22 Teilnehmer nahmen an einer Aufgabe teil, bei der sie durch ein Gebäude navigieren mussten. Die Teilnehmer der Studie trugen dabei eine Microsoft Hololens. Die Microsoft Hololens erfasste die Position und Orientierung der Teilnehmer und visualisierte den berechneten Pfad. Ein Vergleich zwischen dem implementierten FOV assisted path und der Implementierung eines shortest path-Algorithmus ohne Berücksichtigung von FOV-Informationen wurde dabei durchgeführt. Die Ergebnisse geben einen Hinweis darauf, dass der implementierte Pfad die Orientierung der Benutzer innerhalb eines Gebäudes vereinfacht, wenn die Richtung zum Ziel nicht bekannt ist. Darüber hinaus wurde die Benutzerfreundlichkeit des FOV assisted planning Algorithmus und die Eignung von AR-Visualisierungsgeräten für Navigation innerhalb von Gebäuden evaluiert.

Abstract

Superimposing a virtual path onto a real indoor environment assists a person walking through a building towards a requested destination. With the release of AR toolkits for Android and Apple mobile devices and the availability of see-through head mounted displays, AR gains attention on the mass market. The main drawback of state-of-the-art AR visualization devices is the small FOV. The FOV represents the area of the environment displayed on the mobile device. This corresponds with the area where virtual objects are superimposed onto the real environment. Keeping the calculated path inside the FOV of the AR visualization device is the novel approach introduced in this thesis. The path should adopt to the view direction of the mobile device; hence, a path is visualized to the user at all times.

This thesis presents the design and implementation of the FOV assisted path planning algorithm. A 3D model of the indoor environment and tracking data of the user are applied to calculate the FOV assisted path. The 3D model of the environment is processed to represent the walkable areas of the environment. Walkable areas describe all surfaces of the indoor environment a user is able to walk on; they represent the areas in a so-called navigation mesh. Paths are only calculated inside walkable areas. Generation of a navigation mesh is not part of the work at hand. This thesis applies the Recast Library by Mononen. Position and orientation data of the mobile device are tracked and provided to the implemented path planning algorithm. The FOV assisted path consists of a path inside and outside the FOV. The main focus of the work at hand is to calculate the path inside the FOV area and concatenate it to the path outside the FOV. Concatenating the two sub paths introduces different special cases for path calculation. Depending on the view direction, different path calculation stages and special cases are implemented and the resulting sub paths are concatenated. Moreover, dynamic obstacles not represented in the 3D model of the indoor environment are incorporated in the path planning process. A runtime efficient approach to include dynamic obstacles into the large static navigation mesh of a 3D environment is presented in this thesis. Another focus of this thesis is a visually pleasing and intuitive walkable path. Therefore, the FOV assisted path is calculated with clearance to walls and obstacles. Furthermore, a path smoothing algorithm is presented in this thesis.

A performance analysis and a user study was conducted to assess the performance and usability of the implemented algorithm. 22 participants were guided through an indoor environment wearing a Microsoft Hololens. The Microsoft Hololens tracks the participants'

movements and visualizes the calculated paths. A comparison between the implemented FOV assisted path and a shortest path implementation not including FOV information was conducted. Results indicate that the implemented path increases orientation of the user within the indoor environment when the direction towards the destination is unknown. Moreover, the usability of the FOV assisted path planning algorithm and the suitability of AR visualization devices for indoor navigation are assessed.

Contents

CHAPTER 1

Introduction

Navigation from a current location to a destination is a common task in many areas of application. In computer games virtual characters or other moveable entities are guided through a virtual environment for an immersive experience or to simulate human behavior [8]. In real life scenarios the Global Positioning System (GPS) assists a person walking through a city on foot or driving a car to reach a destination [23]. A path connects two locations, considering the walkable areas of the environment as valid areas for the path, whereas unwalkable areas are excluded from the path planning process. This thesis focuses on planning a path within an indoor environment. The path guides the users along the floor of the environment representing walkable areas and avoids unwalkable areas like walls or obstacles. Locating shops inside a shopping mall, finding the departure gate of a flight in an airport or assisting a service technician to reach his next maintenance task inside a warehouse are areas of application for this algorithm.
Due to the development of see-through HMD and the release of AR toolkits for mobile devices, AR technologies push into the mass market. AR is the combination of virtual objects with the real environment [3]. Superimposing a virtual path onto the real indoor environment emerges as the visualization goal of the work at hand. A major drawback of state-of-the-art mobile devices and see-through HMD's is that these devices are only able to superimpose virtual information in a small area of the real environment. In other words, the mobile device has a narrow FOV. Calculating a path that is inside the device's, hence the user's, FOV at all time is the main topic of the work at hand.
Path planning inside a building has three different fields of focus. It starts with the localization of the user inside the indoor environment, then the calculation of the path from a predefined starting point or current position to a destination and the visualization of the calculated path. Many publications on navigation inside buildings focus on tracking the user. Algorithms using fiducial markers [33, 40] or markerless visual features [28] are presented to provide accurate positions of the user. Markers or feature hotspots are spread around the environment and are used as reference points to locate the user. The

final path consists of a chain of reference points and the user is guided from reference point to reference point with directional information. The focus of the work at hand is to use position and orientation information as well as a 3D model of the indoor environment to calculate a continuous path to the destination. The 3D model data is used to separate walkable areas from unwalkable areas to provide the user with a collision-free path. The orientation information is applied to calculate paths with regard to the small FOV of state-of-the-art AR visualization devices. The narrow FOV introduces special path planning cases. Different solutions for these special cases are presented in this thesis. Furthermore, the path is optimized so that it enhances the advantages of AR visualization.

1.1 Motivation and Aim of the Thesis

The motivation behind this thesis is to analyze the challenges and possibilities that state-of-the-art AR technology introduces to path planning within indoor environments. The biggest challenge when using AR technology is the limited FOV of AR devices. Due to the small FOV of the mobile device, different situations lead to entirely different paths. Pointing the mobile device towards the destination, distractions that shift the focus of the user away from the optimal path or loosing orientation inside the environment entirely describe only some of the possible situations. Providing a reliable path to the user for different situations is a challenge for this thesis. Comparing different path planning publications led to an idea for the algorithm at hand. Some indoor navigation publications use tracking data to calculate coarse directions towards the destination, whereas other publications use precise knowledge about the indoor environments to plan accurate paths. In this thesis the position and orientation data of a localization technique and the 3D model of the indoor environment are combined and used in the path planning algorithm. Combining the results of an initial literature research and the ideas of the author of this thesis led to the following research goal:

A path planning algorithm that uses localization data and the 3D model of the indoor environment, with special focus on AR challenges like limited FOV, is implemented to provide a final path suitable for visualization on AR capable devices.

The implemented algorithm is referred to as FOV assisted path planning algorithm for the remainder of this thesis. The algorithm has to adapt to the view direction of a mobile device and has to have clearance to walls and obstacles to avoid collisions. The algorithm has to work inside buildings with multiple floors and has to adapt to dynamic obstacles. Existing path planning algorithms are reviewed and analyzed, and the question whether it is possible to use them in an AR setting is answered. The algorithm is developed in a simulated environment and further evaluated in a real world environment. Besides the evaluation of the FOV assisted path planning algorithm against the defined requirements, the algorithm is compared to the often used A* algorithm introduced by Hart et. al. [20]. In addition to performance tests a user study is conducted. The user study results provide

feedback on the usability of the implemented path planning algorithm and the following questions should be answered. Are the user test participants able to navigate through the test environment and reach the destination at all? Is the FOV assisted path intuitive enough, so that the user is able to reach the destination? Is the path calculated with enough clearance or is the user hitting obstacles or walls along the path? Is the user able to react to changes in the environment, hence a change in the calculated path?
The algorithm uses tracking data provided by a localization technique. Implementation of a localization method or the discussion of advantages and disadvantages of different localization methods is not part of this thesis. The generation of the used 3D model is also outside the scope of this thesis.

1.2 Structure of the Thesis

In Chapter 2 state-of-the-art methods with regard to AR path planning algorithms are reviewed. The review is split in a review of AR related path planning algorithms and a review of general path planning concepts used to implement the FOV assisted path planning algorithm. Basic algorithms and their extensions are presented. The literature review is followed by the definition of requirements and the system design for the algorithm itself in Chapter 3. A short introduction on the localization of the user within the indoor environment and visualization devices for the FOV assisted path is also presented. In Chapter 4 every stage of the implemented FOV assisted path planning algorithm is presented in detail. The algorithm is implemented in C++. Next, a technical evaluation and the results of a user study conducted as part of this thesis are presented. The focus of the technical evaluation are the effects of certain input parameters on the overall runtime of the algorithm. The algorithm has to be fast enough to allow the user a delay-free navigation towards the destination. During the user tests the system is evaluated in a real life scenario on a Microsoft Hololens. The user study provides feedback on the usability of the system and the user workload to solve a path planning task. The thesis concludes with a summary of the work at hand and supplies an outlook on possible areas of improvement for future work in Chapter 6.

CHAPTER 2

Related Work

This chapter focuses on related work associated with this thesis. The chapter starts with a review on indoor navigation algorithms in the context of AR. AR indoor navigation starts with locating the user inside the environment, then a path to the destination is calculated before visualizing the path on an AR capable device. The work at hand uses localization data without introducing a unique localization technique and focuses entirely on the path planning algorithm. Particular algorithms in relation to the path planning algorithm implemented as part of this thesis are reviewed in the second part of this chapter.

2.1 Indoor Navigation in an AR Setting

Indoor navigation describes the concept of finding a route between two points and in addition the visualization of the route. AR capable devices introduce new opportunities for the visualization of calculated paths. Azuma defines AR as virtual objects superimposed onto or combined with the real world [3]. Azuma further states three characteristics to describe AR [3]:

- Combines real and virtual
- Interactive in real time
- Registered in 3D

All of the above points have a connection to the work at hand. The combination of real and virtual implies that the user is able to see the real environment superimposed with a visualization of the virtual path. As the user moves through the indoor environment, the path is updated or recalculated in real time. Thus the visualization of the path has to

be updated in real time as well. Registration of the virtual path in respect to the real environment deals with the third point in Azuma's definition. Errors in the registration lead to paths cutting through walls or paths colliding with obstacles. The position of the visualization of the virtual path is also related to registration with the real world. When the path is visualized slightly above the ground level of the real environment it should never cut through the environment as users get irritated by visualization errors related to registration.

A main factor for correct registration in relation to the real environment is accurate tracking of the user. A majority of publications for indoor navigation in an AR setting present localization techniques and add path planning on top of the localization method. Localization data and directional information is often correlated to each other as Rehman et. al. [39] and Alnabhan et. al. [2] describe in their publications. Prerecorded features are matched with the features captured from the live feed of a mobile device camera to track the user. At specific reference points of the indoor environment directional information guiding the user towards the destination is added to the prerecorded localization features. As the user moves through the environment the user's position is detected and the prerecorded direction information is visualized to the user.

Rehman and Cao use feature matching along predefined paths in their tracking algorithm [39]. Visual features, specifically 3D point clouds, are captured with a mobile device and matched with prerecorded features stored in a database to localize the user. At certain points along a predefined path directional information is added to the prerecorded localization features. Alnabhan and Tomaszweski present a similar approach in their indoor navigation algorithm called Indoor Navigation System Using Augmented Reality (INSAR) [2]. The features used for localization differ between Rehman's approach and Alnabhan's approach. Whereas Rehman uses visual features, Alnabhan uses Wi-Fi fingerprints. Every reference point entry consists of a Wi-Fi fingerprint as a localization reference and directions towards every destination point in the indoor environment for path planning. For both approaches the paths are predefined and directional information is added to every reference point. When new paths or destinations are added, all reference points inside the environments have to be updated. For the work at hand paths should be calculated on the fly for all reachable destinations within the indoor environment.

In addition to the static direction information at reference points INSAR uses the orientation of the mobile device to calculate the direction towards the destination. Applying both orientation values assures that the path is always pointing towards the destination independent of the rotation of the mobile device. The adaption of the path direction in relation to the view direction of the mobile device is also implemented in the FOV assisted path planning algorithm of this work, but with a fundamental difference. Instead of keeping the path direction the same regardless of the mobile device rotation, the path direction is adapted to the mobile device's rotation in the implemented algorithm. This means that for INSAR the user has to turn around to look towards the destination again, whereas for the implemented algorithm the calculated path adjusts towards the destination. Figure 2.1 visualizes the difference between INSAR and the FOV assisted path. The red arrow is the reference direction, where the user is looking straight forward

towards the door. The blue arrow visualizes the concept of INSAR. In INSAR the path always points towards the destination hence the door, regardless of the rotation of the mobile device. The FOV assisted path adapts the path in relation to the view direction of the mobile device before adjusting back to the door.

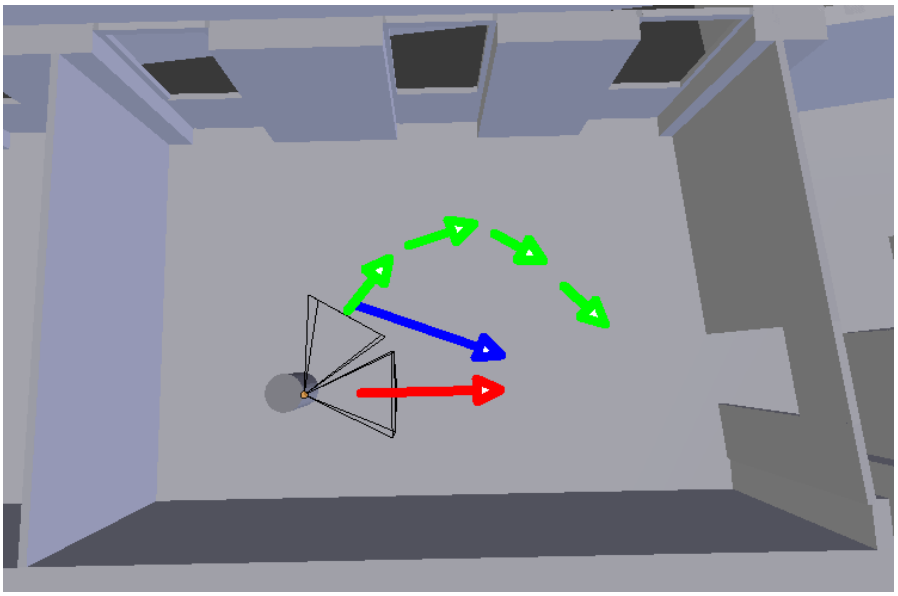


Figure 2.1: Three different directions towards the door, with the reference direction (red arrow) and two directions for different concepts (INSAR - blue arrow, FOV assisted path - green arrow)

Instead of predefined paths or directions towards the destination a variety of algorithms generate a graph based on the indoor environment and facilitate a graph search to calculate a path in real time. Rooms, corridors or other points of interest represent the nodes of the graph. Neighboring nodes are connected with edges. Indoor navigation algorithms relying on graph-based path planning algorithms differ in the techniques they use to track the position of the user inside the environment. Localization with fiducial markers [33, 40, 25], markerless visual features [28] or hybrid methods applying a variation of marker and markerless, inertial and ultrasonic tracking schemes [24, 9, 31] are used to track the users. Based on the graph, shortest path algorithms like Dijkstra's algorithm [12], the A* algorithm [20] or variations of these algorithms are implemented to calculate a path. Path planning starts at the graph node closest to the current user position and heads towards the graph node closest to the destination. Figure 2.2 visualizes an indoor environment with a user defined graph. Nodes for the graph are placed in the middle of rooms, hallways or at hallway branches. This graph is the input for shortest path algorithms to calculate a path. Although the resulting path describes the briefest connection between points on the graph, no detailed directions how to follow the path are

calculated. Reitmayr and Schmalstieg present a two part indoor navigation system called Signpost [40], where the shortest path is extended with detailed directional information. At first, Reitmayr generates an adjacency graph connecting neighboring rooms (rooms are nodes of the graph connected by edges). A shortest path algorithm is performed on the graph to calculate the route to the destination. Furthermore, directional feedback to find the connection of the user's current room towards the next room on the path is provided. Every connection between two rooms is called a portal. In Signpost an arrow points towards the next portal on the path to assist the user with orientation inside the environment. The implemented FOV assisted path planning algorithm also extends an initial shortest path. The path is extended to provide a path with clearance to objects, adjusting to the view direction of a mobile device and avoiding dynamic obstacles. A shortest path is the briefest connection between two points resulting in paths heading along walls or close to obstacles. For the implemented algorithm clearance to obstacles is required to assure the user is not hitting the objects. The work at hand calculates a path with clearance based on the initial path. As stated in the introduction the effect of the small FOV of AR visualization devices is the main focus of this thesis. Therefore, this thesis expands the initial path and alters the path with FOV area information. This results in a path adapting to the view direction of the mobile device.

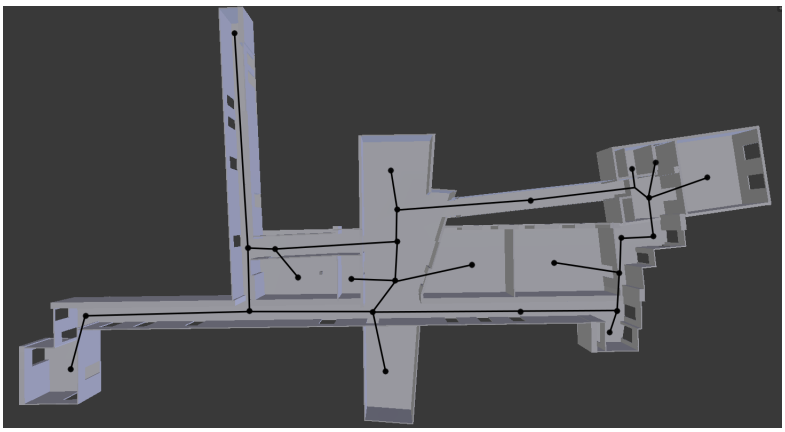


Figure 2.2: Indoor environment with a user defined graph

Another active field of research for AR path planning applications is the visualization of the path. A suitable visualization assists the user following the path. Common visualization forms are arrows pointing towards the destination [39, 2, 33, 40], a continuous line visualizing the path or more unique solutions like the particle system presented later in the user study for the FOV assisted path (see Figure 5.4a). Directional Arrows only indicate a coarse direction towards the destination. No feedback about avoiding walls or obstacles along the path is visualized either. The user has to decide the exact movements without visual assistance as only the direction is visualized. Another drawback with directional arrows are irritating directional visualizations. In Figure 2.3 it is not clear

if the user has to go through the door or follow the corridor and turn to the left. To prevent irritations the work at hand uses a continuous path visualization.
The implemented path planning algorithm adapts to the view direction of the user. The path points towards two different directions. First, the path points towards the user's view direction. Then the user is led towards the destination. To visualize the combination of both directions a continuous path visualization is chosen. The final path is a continuous path visualization with separation to walls and obstacles.

Figure 2.3: Irritating visualization with directional arrows

The following techniques are not implemented in the work at hand but should provide an overview on additional techniques used in path planning publications. For better orientation in the indoor environment Mulloni et. al. use a spatially registered World-In-

Miniature (WIM) map view at info points [33]. The WIM map view visualizes the current location, a path to the destination and the destination. In Signpost a WIM map displays the shortest path from the current room to the destination room as well [40]. All rooms along the shortest path are colored in the same unique color. Delail et. al. visualize room numbers or room names in addition to path information to support orientation within the indoor environment [9]. Other techniques such as activity-based instructions like "move 10 steps" or "change floor" [33] or voice commands [39] are used in addition to AR visualization techniques.

2.2 Path Planning with Navigation Meshes

The idea for the implemented FOV assisted path planning algorithm is a continuous path that is walkable for the user at all times. All walkable surfaces are extracted from the 3D model of the indoor environment. The FOV path is calculated inside the walkable area. All walkable areas of the indoor environment are represented with a data structure called navigation mesh. The definition of the term navigation mesh was first introduced by Snook [41] and Tozour [46]. Different navigation mesh representations characterize the indoor environment with unique properties. Various navigation mesh techniques and their effects on path planning algorithms are presented in this section.
Wein et. al. present a navigation mesh combining shortest path information and clearance to obstacles information [52]. The navigation mesh data structure is called Visibility-Voronoi complex. The navigation mesh combines a visibility graph and a voronoi diagram. The visibility graph represents shortest paths between obstacles in the indoor environment. The voronoi diagram on the other hand represents paths with maximum clearance to obstacles. The combination of the two graphs, hence the Visibility-Voronoi complex, represents all global shortest paths between two points in the indoor environment with arbitrary clearances [52]. The work at hand splits up the generation of the navigation mesh and the path planning algorithm. As a consequence the navigation mesh for the indoor environment will typically be calculated only once, whereas the path planning algorithm is repeated whenever the position of the user changes. Breaking up the navigation mesh generation process and the path planning algorithm speeds up the path planning algorithm if the navigation mesh generation process is slow. Furthermore, navigation meshes often do not change, as they represent the static walkable areas of an indoor environment. Repeating the navigation mesh generation process for every path planning task is not necessary in such cases.
Kallmann [26, 27] and Geraerts [14] also divide the navigation mesh generation process and the path planning algorithm into separate stages. Kallmann uses Delaunay triangulation of the input geometry with an added clearance value to generate a triangulated navigation mesh called Local Clearance Triangulation (LCT) [26, 27]. Geraerts and van Toll et. al. generate a navigation mesh called Explicit Corridor Maps (ECM) [14, 48] based on the medial axis [37]. The medial axis is related to the voronoi diagram and represents points which are equidistant to two or more polygon obstacles. In other words, the medial axis

represents maximum clearance to obstacles and walls of the environment. The medial axis in combination with exact clearance values at unique points on the medial axis form the navigation mesh called ECM. The LCT and the ECM work on planar surfaces. A 3D model of a multistory indoor environment is split up in multiple planar layers and a separate navigation mesh is generated for each layer. Van Toll et. al. extend the ECM for multilayer environments [47, 48] through connecting different layers at connection points. Both the LCT and ECM require planar surfaces to begin the navigation mesh generation process. Extracting planar surfaces from the 3D model of the indoor environment is not topic of the cited algorithms. Separating multistory buildings into separate layers and later connecting them is not preferred for the algorithm at hand as the FOV of the user might be spread over two floors.
Near optimal generator of navigation meshes (NEOGEN) by Oliva and Pelechano [34] and the Recast library by Mononen [32] present algorithms which generate multilayer navigation meshes based on a 3D model of the indoor environment. In a voxelization stage the 3D model is registered to a voxel grid. The voxelized indoor environment is processed and the walkable surface is represented as convex polygons. For an in depth review of the navigation mesh process the reader is referred to [34] for a detailed description of NEOGEN. In Section 4.2 an overview on the navigation mesh generation process of the Recast library is presented. The Recast library is used to generate the navigation mesh for the FOV assisted path planning algorithm.
NEOGEN and the Recast library combine multistory buildings in one navigation mesh, whereas LCT and ECM have a navigation mesh for every floor of a building which might be combined. Due to the voxelization process in NEOGEN and the Recast library the precision of the walkable surfaces in relation to the real 3D structure of the indoor environment depends on the voxel grid size. Contrary to that, LCT and ECM produce an exact representation of the walkable areas in their navigation meshes [49]. For a more in depth analysis of different navigation mesh generation algorithms the reader is referred to a comparative study on different navigation meshes [49]. Van Toll et. al. present different navigation meshes and define metrics to compare navigation mesh generation algorithms [49].
Dynamic obstacles alter walkable surfaces of an indoor environment, thus the navigation mesh is altered. LCT [27] and ECM [50, 48] locally alter the navigation mesh to include dynamic obstacles. Including dynamic obstacles implies a change of the static navigation mesh. Especially for large navigation meshes this is a time consuming task. For the work at hand a novel approach including dynamic obstacles is implemented. Dynamic obstacles are only considered inside the FOV area of the user.

The path planning stage of the work at hand is based on planning paths inside path corridors [27, 14]. A path corridor is a subset of a navigation mesh and characterizes the walkable area of the indoor environment from the current location to the destination. Path planning algorithms based on path corridors often use a global path planning algorithm followed by a local path planning algorithm. Global path planning calculates the shortest path with sufficient clearance to obstacles between the current location and the destination. Local path planning adapts the initial path to avoid dynamic obstacles

or other users in crowd simulation scenarios [27, 14, 48]. The FOV assisted path planning algorithm calculates a global path to find the shortest path corridor with clearance between the current location and the destination. Local path planning includes the FOV information of the user and avoids dynamic obstacles.
Shortest path algorithms are applied for the global path. The A* algorithm introduced by Hart, Nilsson and Raphael in 1968 is a graph-based shortest path algorithm [20]. A common A* implementation is to split up the environment into a grid-based representation of the environment. Based on the generated graph a shortest path between two points on the grid is calculated. A* always finds the shortest path as long as a connection exists at all. Due to the fact that the A* algorithm results in the shortest path to the destination, the path lacks clearance to obstacles or walls. Additional techniques are used to provide paths with clearance.
Kallmann generates an adjacency graph of the triangulated indoor environment in the LCT path planning algorithm [27]. The A* implementation of LCT calculates a corridor of triangles from the start point to the destination. Besides using the Euclidean distance measure to calculate the shortest triangle corridor, the clearance values of the triangles are checked. Validating the clearance results in a corridor where all triangles have a clearance value of at least the user's radius. All triangles with to little clearance are excluded from the A* algorithm. Once a global triangle corridor is found, a local path planning algorithm is used to find the shortest path inside the triangle corridor. Figure 2.4 shows the triangulated input geometry of LCT. Every triangle of the navigation mesh has a clearance value describing the distance to the closest obstacle. The algorithm uses the radius of the user, a start point and a destination to generate a path corridor. The path corridor is represented through a sequence of triangles calculated with the A* algorithm. Figure 2.4 visualizes the shortest path for a certain radius from the start point to the destination.

The Recast library also generates a navigation mesh with clearance to obstacles. The navigation mesh with clearance is used in the implemented algorithm and thus a global path corridor with clearance to obstacles is generated. Calculating a valid global path with enough clearance saves time as the local path planning is started only if a global path with clearance exists.
Similar to LCT a global path corridor is generated for path planning in ECM(yellow colored area in Figure 2.5a). The path corridor relies on the medial axis (red line in Figure 2.5a) of the navigation mesh. The part of the medial axis inside the corridor is referred to as backbone path. Inside the corridor the local path planner calculates a smoothed path with arbitrary clearance. The smoothed path (blue doted line in Figure 2.5b) is based on the backbone path. Path smoothing generates an intuitive and visually pleasing looking path for the user. A smoothed path is easier to follow for the user in the AR setting of this thesis. The implemented FOV assisted path uses a potential field approach for path smoothing. Potential field algorithms make use of different forces which steer the user towards the destination [29]. Attracting forces steer the agent towards the destination, whereas rejecting forces push the agent away from

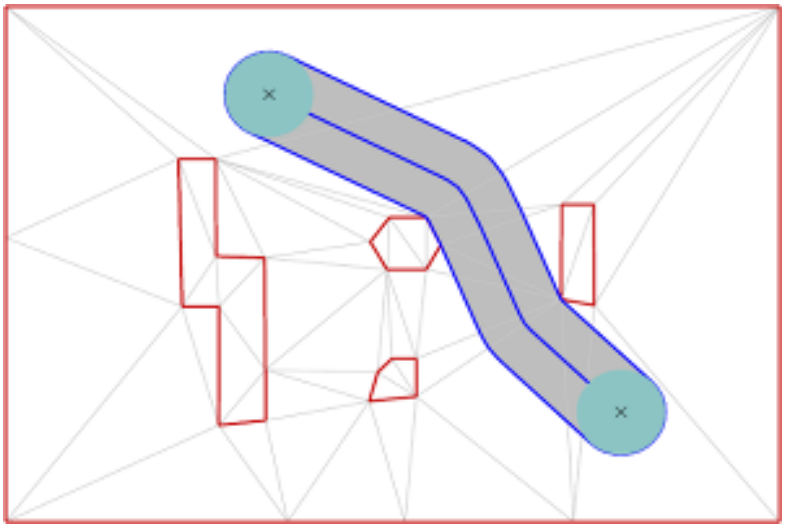
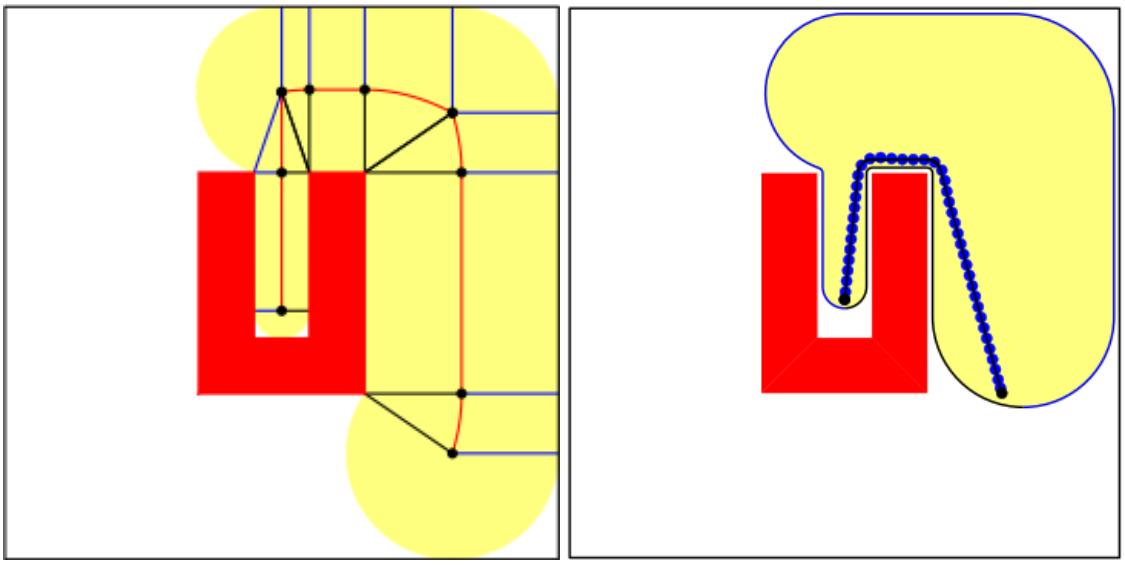


Figure 2.4: LCT navigation mesh and path. Source: [26]

walls and obstacles. The implemented FOV assisted path planning algorithm steers the user towards the destination based on a backbone path.



(a) Backbone path (red), clearances to obsta-(b) Smooth path with minimal clearance based cles (black, blue) and path corridor of ECM.on the backbone path. Source: [14]
Source: [14]

Figure 2.5: Visualization of global and local path planning in ECM

CHAPTER 3

System Design

The main contribution of this thesis is the implementation of the FOV assisted path planning algorithm. The implemented algorithm is part of an AR indoor navigation system consisting of the following three parts:

- Localization of the user
- Path planning between the user's current position and the destination
- Visualization of the path

Figure 3.1 visualizes the main parts of the navigation system. A Unity3D project reads tracking data from a localization technique and loads the 3D model of the indoor environment. This information is provided to the path planning library. The implemented path planning algorithm extends the Recast library by Mononen [32]. The resulting path is then prepared for visualization in the Unity project and displayed on AR visualization devices.

This chapter presents the Unity3D project components first. An introduction of Unity3D is followed by requirements for localization techniques. Next, an AR visualization toolkit and device is presented. The second big block of this chapter represents the implemented path planning library. It starts with an introduction of the Recast library. At the end of this chapter the main contribution of this thesis, the FOV assisted path planning algorithm, is presented. Information about basic requirements of the algorithm and the design of the unique stages are discussed.

3.1 Unity3D Project

Unity3D is a cross-platform game engine developed by Unity Technologies [45]. It supports over 20 different platforms i.e. Windows, Linux, macOs, iOs, Android and many

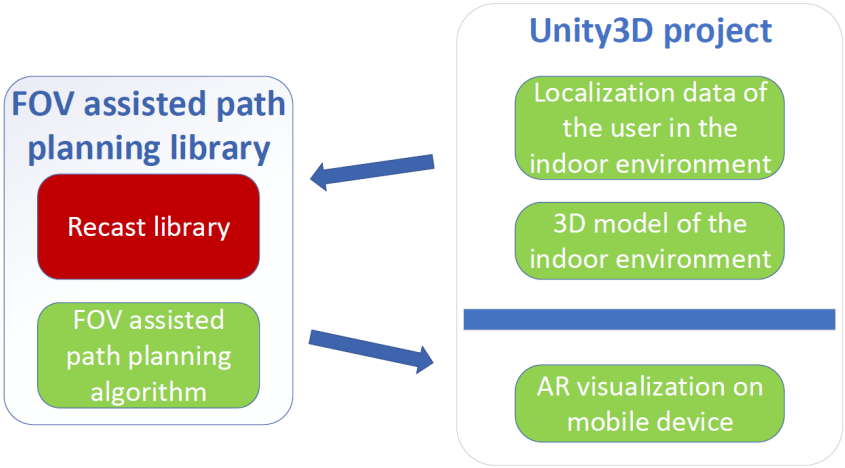


Figure 3.1: Visualization of the different parts of the AR navigation system. Green blocks are implemented as part of this thesis, red blocks are existing libraries

more. It also supplies easy access to HMD's like the Occulus Rift or the Microsoft Hololens. This feature makes Unity3D a suitable tool for this thesis as the user tests presented in Chapter 5 feature the Microsoft Hololens. Unity3D provides a powerful editor to design 2D or 3D environments. It also features scripting through C# and JavaScript. The scripts for this project are written in C#. Figure 3.2 visualizes all the functions the Unity3D project handles. In Unity3D the 3D model of the indoor environment is loaded and prepared for the path planning library. The library uses the vertex and triangle index information of the imported 3D model. Besides the 3D model the path planning library also requires position and orientation data from the user. The Unity3D project processes tracking data from a localization technique and provides it to the library. In the user study tracking data is provided by the Microsoft Hololens, processed by the Unity3D project and then used in the path planning library. The path recalculation logic controls how often the path planning library is invoked. The remaining functions in Figure 3.2 are implemented for visualization of the results of the path panning library. The recalculation logic and the visualization methods are presented in detail in Section 4.4.

3.2 Tracking

Localization data is a requirement for the FOV assisted path planning algorithm. The algorithm uses the position and orientation of the user, therefore, a six degree of freedom (DOF) tracking system is necessary. Without the position information of the user the path planning algorithm is unable to start the path calculation and no path is calculated to begin with. Discontinuity of tracking data once a path is calculated leads to confusion of the user and might misguide the user towards an incorrect direction. Therefore,

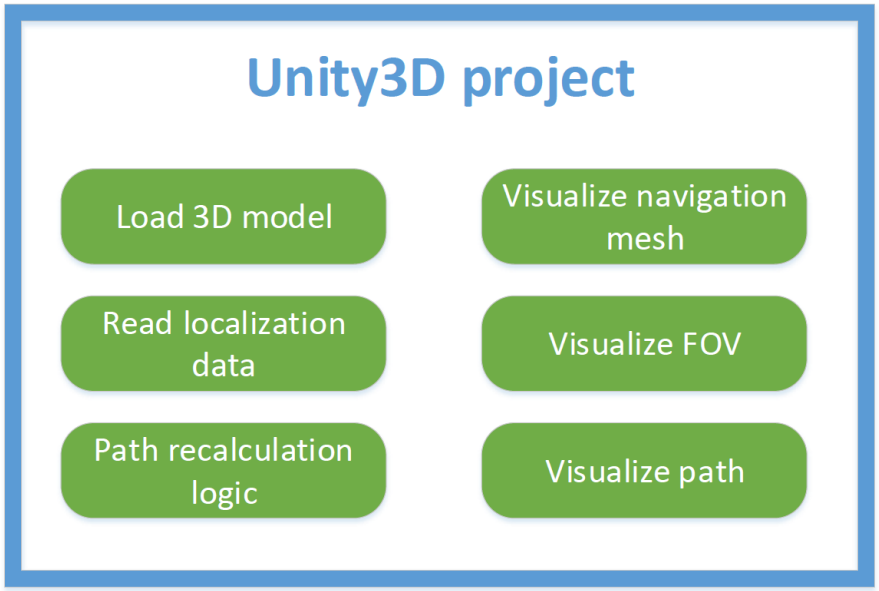


Figure 3.2: Different functions of the Unity3D project

reliability of the tracking algorithm is a base requirement for the FOV assisted path planning algorithm.
Another requirement is the accuracy of the tracking data. Accurate tracking is necessary for a correspondence of the user's position in the real indoor environment and the true to scale 3D model of the indoor environment. The AR visualization of the resulting path also requires accurate tracking. The following problems arise with inaccurate tracking:

- Faulty positioning leads to orientation problems for the user especially when wearing a see-through HMD.
- A deviation of the user's position in the real environment and the 3D model falsifies the calculated path and confuses the user.
- Erroneous tracking of the user's current position misplaces the user and leads to the user potentially hitting a wall or an obstacle and no path is calculated.
- Inaccurate positioning of the user leads to paths leading through walls in the real environment, hence the user is unable to traverse the path.

For the remainder of this section, a tracking algorithm called HyMoTrack [16] is presented. Gerstweiler et. al. combine three separate tracking approaches to provide a reliable and accurate six DOF indoor tracking algorithm.
HyMoTrack combines visual feature tracking with inertial feature tracking as a fallback, if the visual feature tracking fails to provide continuous tracking data. For visual feature

tracking two separate feature types are used to provide centimeter accuracy[16]. The two visual feature tracking methods are based on mapping 3D point features and 2D image markers of signs, advertisements or paintings. For a detailed description of the tracking methods the user is referred to [16].
As does the FOV assisted path planning algorithm, HyMoTrack is developed to run on mobile devices in real time. To provide real time tracking, the system is split up in an offline map generation to generate a feature map and an online tracking part. The online tracking part uses the features to localize the user. In a potential fusion of the HyMoTrack tracking system and the implemented path planning algorithm, the navigation mesh generation could be added to the offline section of the combined system and the path planning algorithm could be added to the online part.

3.3 Output Devices for AR Applications

There are two interesting topics for the visualization of a path in an AR setting. The first topic focuses on the visualization of a path in an AR setting. The second topic are toolkits and devices applied for AR visualization. There are different ways to visualize the path itself, like arrows, continuous lines or special visualization techniques like particle systems. Besides the visualization form, the position of the visualization has to be defined. The implemented FOV assisted path is visualized slightly above the floor of the indoor environment with continuous lines or a continuous particle system. Comparing different visualization forms and positions is not part of this thesis. The feedback of the conducted user study confirms the chosen visualization form and position as the participants gave positive remarks.

Google AR Toolkits

Google offers two different AR toolkits named Google Tango [18] and the newer Google ARCore [17]. Google Tango was introduced first and focuses on calculating the mobile device's position in relation to the real world [19] and visualizing AR content on the mobile device. Google Tango requires mobile devices with special hardware like depth sensors or additional cameras to locate the user inside the environment. Devices featuring Google Tango are the Lenovo Phab2 Pro or the Asus Zenfone AR. Google Tango is based on three core technologies [19]:

- **Motion Tracking**: Responsible for tracking the mobile device's position and orientation. Google Tango provides 6 DOF tracking.
- **Area Learning**: Registers and recalls the indoor environment previously scanned. This technology also improves the accuracy of motion tracking in known environments.

- **Depth Perception**: Learn shape of the indoor environment. Virtual objects are registered with real world objects. Thus the virtual objects can interact with the real environment.

Google ARCore is the successor of Google Tango. ARCore differs from Tango as it works without requiring mobile devices with special hardware. Google ARCore uses the mobile device's camera for motion tracking and depth perception. These two technologies have the same functionality as in Google Tango. In addition to that ARCore uses light estimation to calculate the current lighting conditions of the environment.

Microsoft Hololens

For Virtual Reality (VR) environments HMD's are the state-of-the-art technology for VR visualization. The Microsoft Hololens is a see-through HMD designed for AR applications. The Microsoft Hololens(see Figure 3.3) is a stand-alone holographic computer that visualizes 3D holographic objects, allows the user to move around the objects and enables interaction with these objects [44].

Figure 3.3: Microsoft Hololens

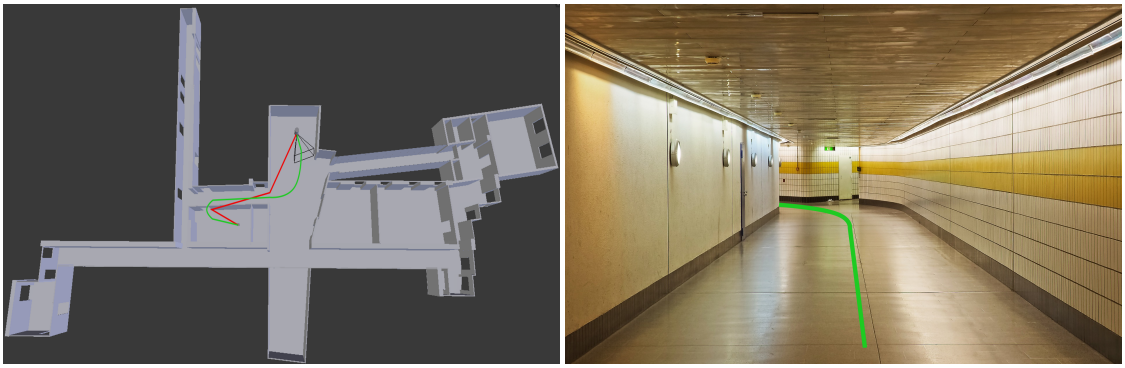
The Microsoft Hololens features three processor units. Whereas a CPU and a GPU are common parts in state-of-the-art PC's, the (holographic processing unit (HPU)) is a unique processor developed exclusive for the Microsoft Hololens.
The Hololens is equipped with a depth camera to record depth information, four additional cameras to track head movement in relation to the environment, an ambient light sensor to measure the environments light intensity and a forward facing video camera, which is used for head tracking and also to capture videos and images of the real environment. The video data of the front camera is superimposed with the virtual objects in the user's view to provide the mixed reality experience for external viewers. The depth camera is based on structured light technology. A special infrared pattern is emitted onto the environment and scanned with an infrared sensor to calculate depth information of the

environment.
In addition to the cameras the Microsoft Hololens has additional sensors like an accelerometer, a gyroscope and a magnetometer. A microphone for voice commands and speakers to simulate spatial sound are also included.
A waveguide based display visualizes the virtual objects. The holograms are emitted from a light source onto a transparent lens consisting of an array of small grooves. The grooves reflect the light beams into the user's eyes and the holograms appear at varying positions and distances in the environment. Holograms are projected at a minimum rate of 30 frames per second and the FOV of the lenses is around 30° horizontal times 20° vertical. The small FOV of the Microsoft Hololens compared to the human visual field which is over 180° horizontal times 140° vertical [42] is the major drawback of current AR visualization devices. The small FOV was the main motivation to introduce the FOV assisted path planning algorithm presented in the next section.

3.4 FOV Assisted Path Planning Library

The AR indoor navigation system guides a user through an indoor environment. The system visualizes the calculated path on an AR visualization device like a smartphone, tablet or see-through HMD. One advantage of using AR is preserving the relation to the real environment while adding guidance with the virtual path. The mobile device superimposes the path onto the live camera feed. The implementation of the navigation system in an AR setting also introduces the main drawback of current AR output devices. Virtual objects can only be visualized in a small area of the environment as the FOV of current AR devices is small. E.g. the Microsoft Hololens has a FOV of approximately 30° horizontal times 20° vertical. In comparison the human visual field is over 180° horizontal and 140° vertical [42]. The main focus of the implemented path planning algorithm is to calculate a path visualized inside the FOV of the mobile device at all times. The path guiding the user towards the destination has to adapt to the view direction of the mobile device. In algorithms where the FOV is not included in the calculation, the user has to locate the path by turning around before following the path. Possible areas of application for the implemented algorithm are the location of shops in a large shopping mall, finding departure gates at an airport or assisting a maintenance worker to locate the next service task in a warehouse.
Figure 3.4 shows two concept images of path planning in an AR indoor environment. Figure 3.4a shows the 3D model of an indoor environment. The red path in Figure 3.4a is the shortest path from the user's current location to the destination. The green path is a concept path featuring path properties like clearance to obstacles or using the view direction for path calculation. Figure 3.4b displays a drafted virtual path superimposed onto a real world environment. The aim of the implemented FOV assisted path planning algorithm results in a path similar to the green path in Figure 3.4a. The calculated path is visualized on an AR visualization device as drafted in Figure 3.4b.

The shortest path (red path in Figure 3.4a) visualizes an unintended result for the implemented path planning algorithm. Although the red path represents the shortest



(a) 3D model of an indoor environment with two paths. Red path visualizes shortest path, green path visualizes concept path of the FOV assisted path planning algorithm

(b) Concept visualization of a path superimposed onto a real environment

Figure 3.4: Concept visualizations for the FOV assisted path planning algorithm

distance from the current position to the destination, the path does not offer the requested guidance for the implemented AR scenario. The following requirements have been defined for the implemented FOV assisted path planning algorithm:

FOV assisted path

The implemented algorithm calculates the final path with regard to the viewing direction of a person respectively a mobile device and gives feedback via an user interface. The algorithm not only has to take account of the position of a person, but rather apply the orientation of a person. The calculated path has to be updated with regard to the viewing direction of the person. The final path has to be inside the user's FOV at all time. This means that the path adapts to the user's FOV, instead of the user having to adapt to the calculated path. A FOV assisted path is calculated, if the view direction faces towards a walkable surface of the environment. When the user faces a wall, fallback paths are calculated.

Clearance to walls and obstacles

The final path has to have sufficient clearance to obstacles or walls along the way. The user should be able to traverse the path without the danger of hitting any objects. Due to this requirement the final path is not the global shortest path. The algorithm should still prevent unnecessary detours.

Dynamic path updates

The algorithm has to adapt the calculated path in a dynamic scenario. Whether it is dynamically appearing obstacles or a person changing the path on purpose, the algorithm has to adapt to new circumstances. Dynamic obstacles are objects which are not part of the 3D model of the indoor environment. The algorithm alters the path, if dynamic obstacles are provided. Detecting dynamic obstacles is

not scope of this thesis. The path is calculated on the fly, meaning that neither precalculated paths, nor predefined reference points between the current position and the destination are used.

AR Visualization
The calculated path is visualized on an AR capable mobile device like a smartphone, tablet or see-through HMD. The limited FOV of a mobile device is a challenge for visualizing the path. Proper visualization of the path is necessary for an easy to use navigation experience for the user.

Smooth path
The calculated path for an AR setting must have smooth motions with clearance to an obstacle. Smooth motion means that a person is able to follow the path in an intuitive way. If a straight hallway represents the indoor environment, the algorithm has to calculate a straight line through the hallway, not a staircase-shaped path.

Real time performance
Real time performance in the sense of this thesis is defined as a person has to be able to traverse the path without visible delays during the visualization of the path. When the user changes the view direction, the path has to be updated and visualized without any visible delay. It is also important that the user is not confused with too many path updates. Very frequent path updates result in jumping paths which confuse the user. A middle ground on the amount of updates for the final path is requested.

These properties are considered in the design for every stage of the FOV assisted path planning algorithm shown in Figure 3.5. At first, the 3D model of the indoor environment is applied to generate a representation of the walkable areas of the indoor environment. The data structure representing these walkable areas is called a navigation mesh. The navigation mesh represents the foundation for path planning and holds information about the clearance to static obstacles and walls. An overview on navigation meshes is introduced later in this chapter (Section 3.4.2). The Recast library by Mononen [32] generates the navigation mesh for the FOV assisted path planning algorithm. The Recast library is presented in Section 3.4.1 and an in depth introduction of the navigation mesh generation process is provided in the implementation chapter(Section 4.2). The second step of the path planning algorithm is to calculate the FOV area of the mobile device. The FOV area represents the part of the indoor environment displayed on the display of the AR visualization device. The FOV area is represented with a separate navigation mesh. The generation of the FOV area hence the FOV navigation mesh is described in Section 4.3.3. Dynamical obstacles are only considered while inside the FOV of the user and thus are only part of the FOV navigation mesh. The FOV navigation mesh is altered to include dynamic obstacles. The last step is the calculation of the FOV assisted path based on the walkable areas of the indoor environment and the FOV area. The FOV assisted path consists of a path inside and outside the FOV area. The generation of the

final path is composed in several stages with special cases for certain conditions. The final path is smoothed to provide an intuitive and comprehensible path for the user. The unique stages to compose the FOV assisted path are presented in Section 3.4.3 and the implementation details are described in the implementation chapter(Section 4.3).

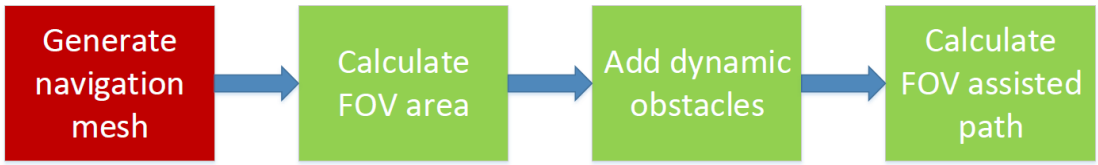


Figure 3.5: Concept stages of FOV assisted path planning algorithm

3.4.1 Recast Library

The Recast Library is a toolset for navigation mesh generation and is used to generate the navigation mesh for this thesis. Recast also includes a toolset for path planning and crowd simulation [32]. These two core features are split up into two packages: The Recast package handles the navigation mesh construction and the Detour package is responsible for path planning and crowd simulation. Detour is a standalone package and is able to process different kinds of navigation meshes, although it is optimized to fit the Recast data. The library is open source and licensed under ZLib license [32]. Key features of the library are [32]:

- Automatic: Recast is able to construct the navigation mesh from the 3D model of an environment without any user interaction. This process will be discussed in Section 4.2.
- Solo and tiled navigation mesh: Navigation meshes are computed in one big navigation mesh (solo) or can be split up in several smaller navigation meshes (tiled) to allow changes and recreation of smaller parts of the navigation mesh.
- Multi-layer navigation mesh: Multistory buildings are represented in one connected navigation mesh
- Temporary Obstacles can be added and deleted from tiled navigation meshes.
- Path planning: The library provides an implementation of A* shortest path algorithm
- Off-Mesh Connections: Create connections between physically not adjacent areas
- Crowd Simulation: Create multiple agents with independent path finding and collision avoidance

The implemented FOV assisted path planning algorithm extends the existing Recast Library. The algorithm at hand uses the navigation mesh generation pipeline and the shortest path implementation of Recast. These core methods are expanded to include the FOV information in the final path.

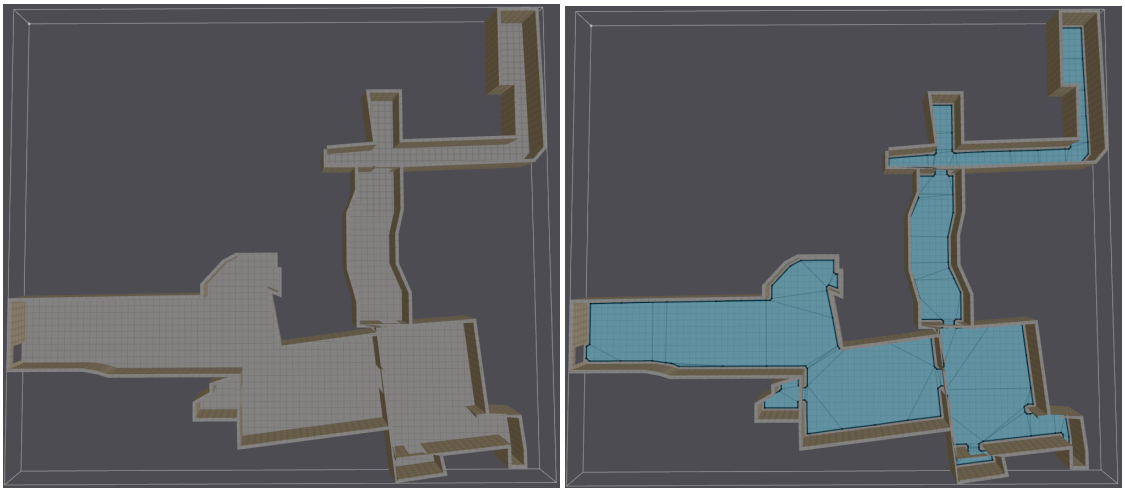
3.4.2 Navigation Meshes

The navigation mesh data structure is the basis for Recast path planning hence for the FOV assisted path planning algorithm. The 3D model represents the entire indoor environment hence the floor, walls, ceilings and other objects within the indoor environment. Usually not all the objects of the 3D model are necessary for the path planning algorithm. A path planning algorithm is only processing the areas of the indoor environment the user is able to walk on. Using only walkable areas of the 3D model saves time and memory for the path planning algorithm. Once the walkable surface is extracted, a representation of the area is required. The most basic representation of walkable space is to divide the environment into a grid of squares(Figure 3.6a). Large environments result in a high amount of squares resulting in high memory usage and longer path calculation times. These drawbacks make a grid based representation of large indoor environments less suitable for mobile devices. A data structure called navigation mesh [41] solves these disadvantages and represents the indoor environment. A navigation mesh is defined with the following attributes:

- A navigation mesh represents the walkable surface of a 2D or 3D environment. A walkable surface is defined as all areas of the environment traversable for the user.
- The walkable surface is described as a set of connected polygons.
- In addition to the representation of the walkable surface a graph with adjacency information of neighboring polygons is generated.

Figure 3.6 shows a square grid based representation and a navigation mesh based representation of the 3D model of an indoor environment. The amount of squares in Figure 3.6a is much higher than the amount of blue polygons in Figure 3.6b. The sum of all blue polygons represents the walkable surface of the indoor environment.

In their comparative study of navigation meshes van Toll et. al. classify two different types of navigation meshes named exact methods and voxel-based methods [49]. Exact methods assume that the 3D model of a multistory building is split into multiple exact defined layers before the construction of the navigation mesh. Van Toll et. al. describe exact methods as more accurate than voxel-based methods, whereas voxel-based algorithms represent multistory buildings in a single navigation mesh [49].
The Recast library used for this thesis belongs to the category of voxel-based methods. Voxel-based methods represent the 3D model of the indoor environment in a 3D grid of voxels. After the voxelization of the 3D model, all voxels belonging to the walkable



(a) 3D model with a square based grid representation of the indoor environment

(b) 3D model with a navigation mesh representation of the indoor environment

Figure 3.6: Comparison of the representation of walkable space of the 3D model of an indoor environment

surface of the indoor environment are flagged for further processing. At the end of the navigation mesh generation process the walkable surface of the indoor environment is represented with a set of convex polygons. For a more detailed description of the navigation mesh generation process the reader is referred to Section 4.2.

3.4.3 FOV Assisted Path Planning Algorithm Stages

The implemented path planning algorithm consists of two different navigation meshes. The first navigation mesh represents the walkable surfaces of the entire 3D model of the indoor environment. The second navigation mesh represents the mobile device's FOV. Just as there are two navigation meshes, there are also two different paths, a path inside and outside the FOV area. The narrow FOV of state-of-the-art AR visualization devices is the main reason to retain the path inside the FOV of the AR visualization device. A path has to be visible on the display of the mobile device at all times. Thus the user is able to follow the path constantly, instead of searching for the path before following it. This means the path has to adapt to the view direction of the mobile device resulting in different path planning scenarios. Figure 3.7 visualizes three different scenarios of FOV assisted paths. The red line visualizes the shortest path calculated with the Recast Library. The shortest path represents a fixed path without accounting for the view direction of the mobile device. When a user does not point the AR visualization device towards the direction of the shortest path, no path is visualized on the display. The green and blue arrows are outlines of possible FOV assisted paths that keep the path inside the FOV of the mobile device.

Figure 3.7a emulates a simple scenario, where the user's view direction is similar to the direction of the shortest path but the shortest path is outside the FOV of the AR device. The path outline follows the FOV and then diverts back to the shortest path. The user should be able to follow the path towards the current view direction, but should also have an indication where the destination is located. Figure 3.7b illustrates a scenario, where the user's view direction is in the opposing direction compared to the direction of the shortest path. One solution would be to visualize a loop for the user that indicates a continuous traversable path towards the view direction (green arrow in Figure 3.7b). Another solution is to calculate a path starting in the middle of the FOV navigation mesh area, heading towards the destination via the user's current position (blue arrow in Figure 3.7b). Figure 3.7c visualizes a scenario simulating the user's view direction heading towards a different direction than the shortest path. The shortest path directs the user towards the right side of the hole in the 3D model. Following the view direction could guide the user towards the left side around the hole (green arrow in Figure 3.7c). This might lead to unwanted detours for the user. The blue arrow in Figure 3.7c visualizes a solution similar to the blue arrow in Figure 3.7b. The path starts in the middle of the FOV navigation mesh area and heading towards the shortest path via the user's current position. These are only three scenarios indicating the challenges of adding FOV information to the path planning algorithm.

The amount of different scenarios result in a final path concatenated of sub paths and several special cases included in the generation of the FOV assisted path. Figure 3.8 visualizes the individual stages including all special cases. A detailed description on the implementation of the algorithm is presented in Section 4.3.

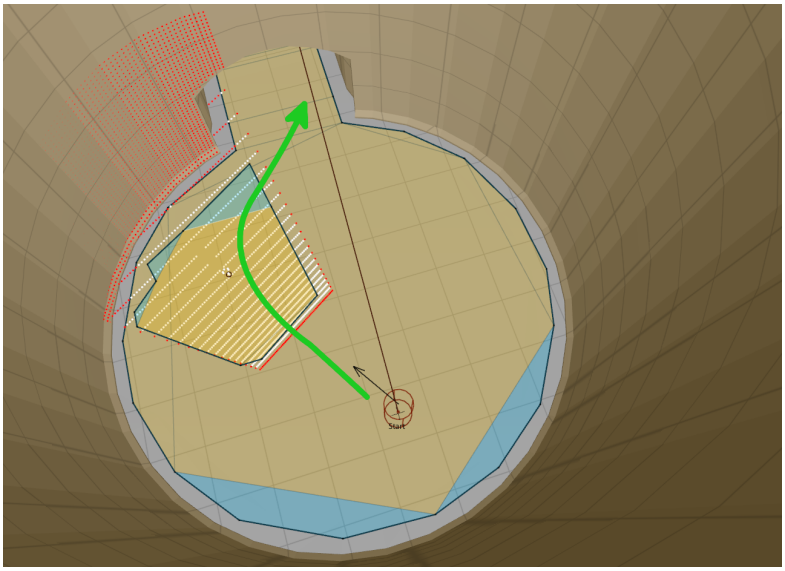
The main factor for the different stages and special cases of the algorithm is the view direction of the mobile device. The yellow boxes in Figure 3.8 visualize the standard path calculation process. The standard scenario represents the user looking towards a walkable surface of the indoor environment and towards the shortest direction to the destination (comparable to Figure 3.7a). In this case the path starts at the user's position heads towards and through the FOV of the user and then heads towards the destination once it exits the FOV of the user. The blue boxes in Figure 3.8 describe special cases, for which the standard path calculation process does not result in desired paths. The FOV assisted path planning algorithm distinguishes the following special cases:

- **No walkable surface inside FOV:** The user looks towards a wall or obstacle. There are no walkable surfaces inside the user's FOV. The path starts in the middle of the unwalkable FOV area, heads towards the nearest walkable surface of the indoor environment with regard to the user's position and then heads towards the destination. Implementation details for this case are introduced in Section 4.3.7.
- **Destination inside FOV:** The destination is located inside the user's FOV. The path is calculated from the user's current position towards the FOV area and then heading towards the destination inside the FOV area.
- **Destination between user and FOV:** The destination is located between the

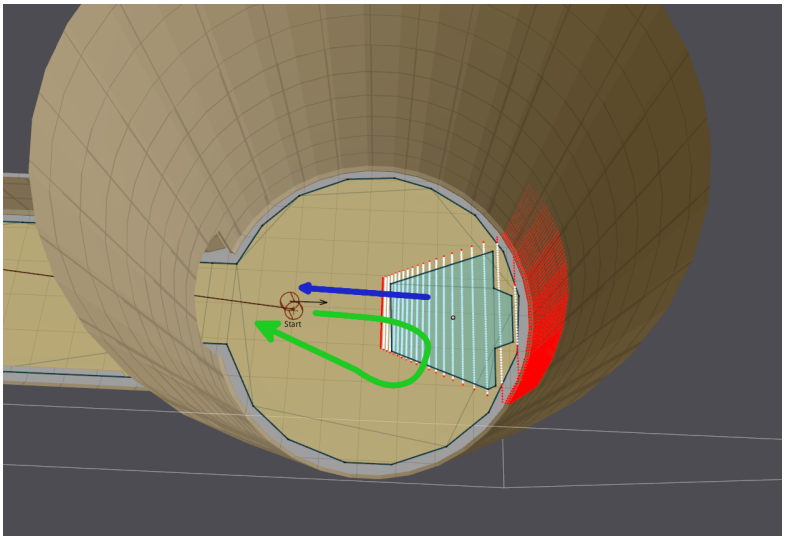
user's current position and the FOV area of the user. This special case often occurs when the user approaches the destination and looks past the destination. In this special case the start of the path is located in the middle of the FOV area and heads towards the destination.

- **User looks away from shortest path direction:** The ideal path direction is defined as the direction towards the shortest path to the destination. If the angle between the user's view direction and the ideal path direction exceeds a predefined threshold, the start point of the algorithm changes from the user's current position to the middle of the FOV area. Thus the path starts in the middle of the FOV area heading towards the user until it exits the FOV area and then heads towards the destination. The blue arrow in Figure 3.7b drafts the idea of this special case. Implementation details are presented in Section 4.3.5.

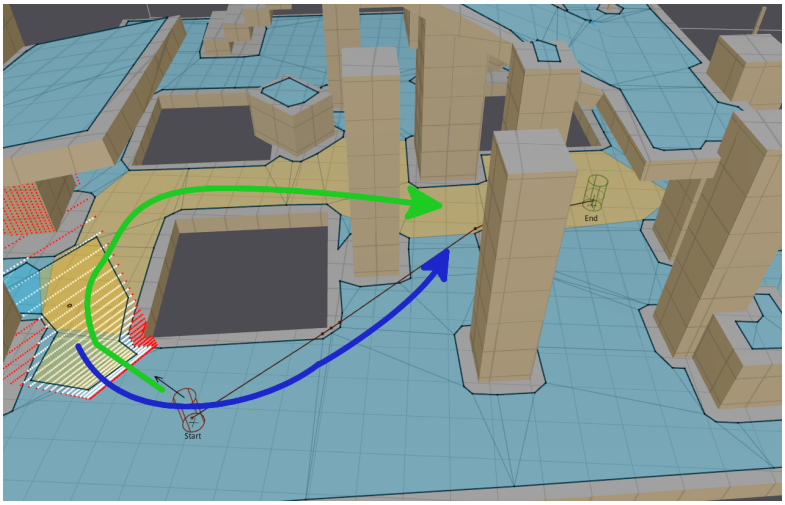
The standard path calculation process as well as the special cases always calculate paths with clearance to walls and obstacles. If the algorithm concept presented in this section fails to calculate a path altogether, fallback paths are calculated to provide a valid path to the user at all times. Implementation details for the fallback paths and the presented algorithm concept are described in Section 4.3.



(a) Outline of a simple scenario for a FOV assisted path



(b) Different outlines for a 180° direction change



(c) Different outlines to the same end point

Figure 3.7: Three scenarios of FOV assisted paths

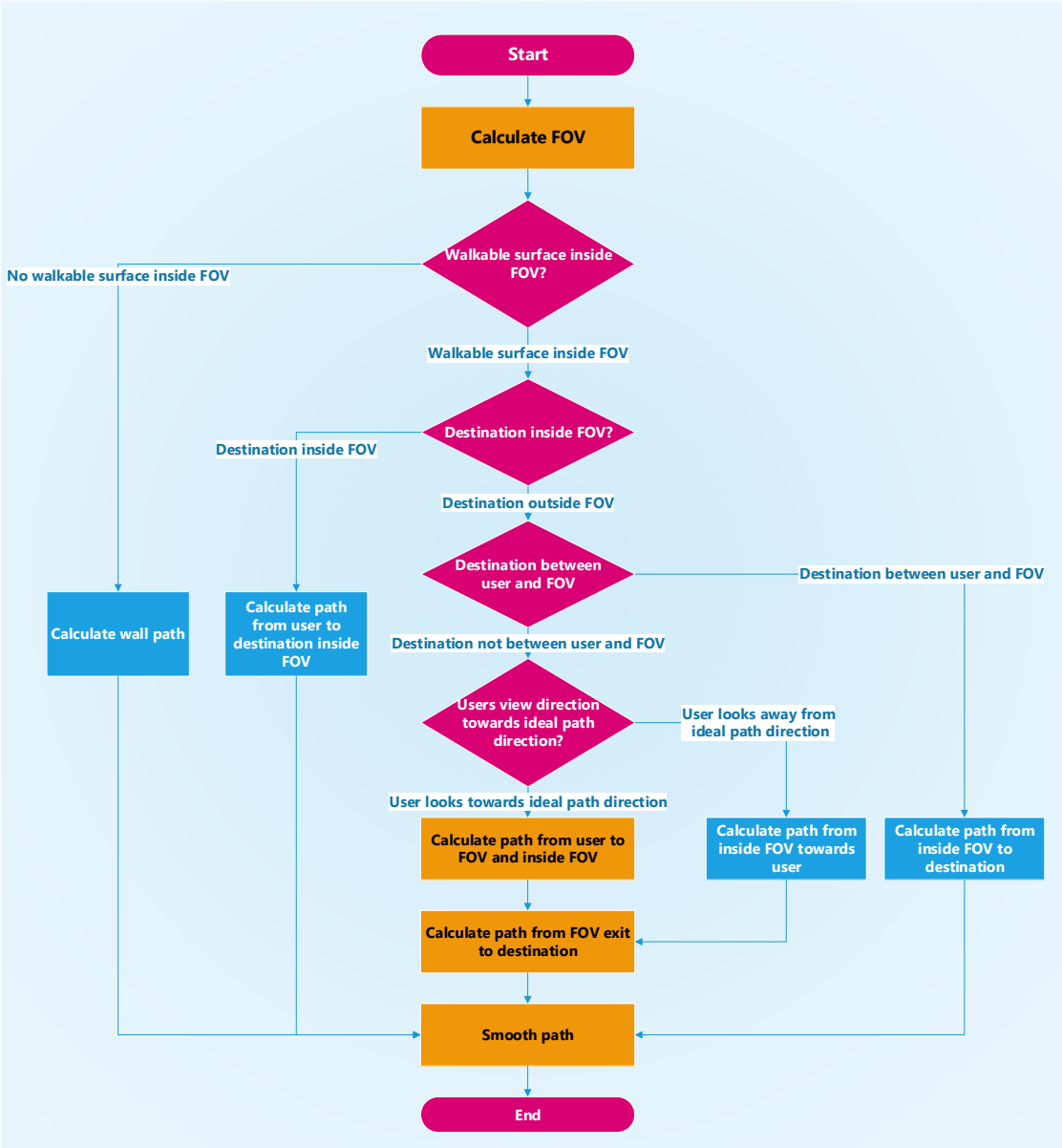


Figure 3.8: Path calculation stages (yellow) including special cases (blue) in the FOV assisted path planning algorithm

CHAPTER 4

Implementation

This chapter presents the implementation of the FOV assisted path planning library. The library is compiled to a Dynamic Link Library (DLL). This DLL is included in the Unity3D project to access the Recast library and the implemented path planning algorithm. The DLL is presented at the beginning of this chapter(Section 4.1). The generation of the Recast navigation mesh is discussed in Section 4.2. Next, a detailed description of the implemented FOV assisted path planning algorithm is presented (Section 4.3). The implementation of a path recalculation logic and details on the path visualization in Unity3D conclude this chapter (Section 4.4).

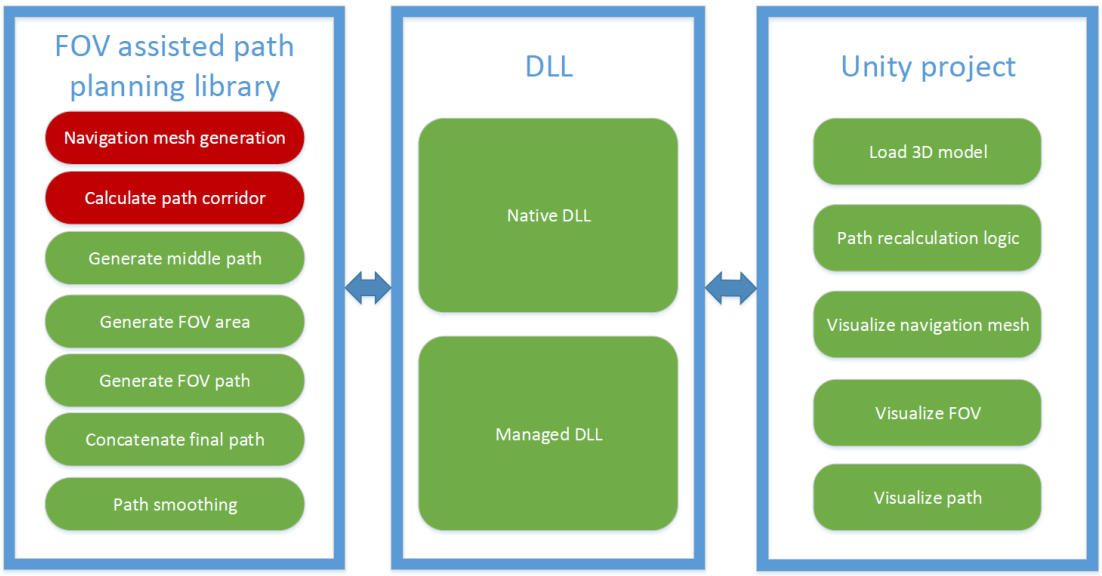


Figure 4.1: Three main modules of the implemented AR navigation system.

4.1 DLL

The FOV assisted path planning library is compiled to a DLL. The DLL facilitates access to generate the Recast navigation mesh and calculate the FOV assisted path. The navigation mesh generation is separated from the path planning. The navigation mesh is only generated once at the beginning of the algorithm. This speeds up the runtime of the algorithm since the generation of the navigation mesh in the library is slower than the path calculation. For the 3D model applied in the user study it can take several seconds or even up to a minute depending on the parameters used in the navigation mesh generation process. For a detailed analysis on the runtime the reader is referred to Section 5.1.
The DLL provides a method to calculate a navigation mesh for the 3D model of the indoor environment. The method generates the navigation mesh for the path planning algorithm inside the library and returns the navigation mesh for visualization in Unity3D. Recast requires the geometry of the 3D model of the indoor environment and a set of Recast specific parameters to start the navigation mesh generation process. The geometry consists of all vertices and triangles representing the 3D model. The Recast specific parameters are parameters needed in separate stages of the navigation mesh generation process which is explained in detail in Section 4.2:

- cell_size: width of a voxel for the rasterization stage (Section 4.2.1)
- cell_height: height of a voxel for the rasterization stage (Section 4.2.1)
- agent_height: height of the user (Section 4.2.1)
- agent_radius: radius of the user (Section 4.2.2)
- max_climb: the maximum step height the user is able to step up or down (Section 4.2.1)
- max_slope: the maximum slope of a surface the user is able to traverse (Section 4.2.1)
- min_region_size: size of a region before it gets excluded from the navigation mesh process (Section 4.2.2)
- merged_region_size: size of a region qualifying for merging with another region (Section 4.2.2)
- max_edge_length: length of an edge which is split up to generate shorter navigation mesh edges (Section 4.2.3)
- max_edge_error: parameter for the Douglas-Ramer-Peucker algorithm (Section 4.2.3)
- verts_per_poly: maximum amount of vertices per navigation mesh polygon (Section 4.2.4)
- sample_distance: sample distance to generate PolyMeshDetail triangles (Section 4.2.4)
- max_sample_error: offset between PolyMesh vertex and 3D model for PolyMeshDetail generation (Section 4.2.4)
- max_edge_error_fov: parameter for the Douglas-Ramer-Peucker algorithm (Section 4.3.3)
- tile_size_fov: tile size for the FOV navigation mesh (Section 4.3.3)

- step_size_smoothing: distance between backbone path points for path smoothing (Section 4.3.6)
- scale: scale between source geometry scale and recast library scale

The DLL also provides a method for the FOV assisted path planning algorithm. The method requires user specific parameters and optional dynamic obstacles information to calculate the FOV assisted path. The position and radius of dynamic obstacles are provided to the library, if dynamic obstacles are applied. The user specific parameters describe the position and size of the user, the destination and FOV related parameters of the mobile device:

- start point for the path calculation
- position from where the FOV is calculated
- view direction to calculate the FOV
- destination for the path calculation
- horizontal FOV of the mobile device
- vertical FOV of the mobile device
- nearViewPlane for the FOV
- farViewPlane for the FOV
- height of the user
- radius of the user
- maximum movement velocity of the user (see Section 4.3.6)
- Perimeter around destination to stop path planning (see Section 4.3.6)
- angle for path rebuild (see Section 4.3.5)
- enable wall path algorithm (see Section 4.3.7)

The path planning library returns three different paths. The specific paths and potential fallback paths are presented at the end of this chapter (Section 4.4). Fallback paths are returned when the generation of the FOV assisted path fails. In addition to the paths a status code and FOV information for visualization on the mobile device are returned. The status code provides information about the success of the path calculation or predefined error codes for error states to the Unity3D project.

4.2 Recast Navigation Mesh Generation Process

This section focuses on the generation of the navigation mesh inside the Recast library. The Recast library splits the navigation mesh generation process in several stages. An overview on every stage is presented to grasp the navigation mesh generation process. Moreover, the result of the navigation mesh process and thus the path planning algorithm relies to a great degree on parameters presented in this section.
Figure 4.2 shows the process from the rasterization of the 3D model of the indoor environment to the finished navigation mesh. The Recast navigation mesh process starts with a rasterization stage. The 3D model is rasterized into a 3D voxel grid. This 3D voxel grid is called a voxel heightfield. The next stage is a filter stage, where voxels are flagged as belonging to walkable or unwalkable surfaces of the indoor environment. Walkable surfaces are all areas of the 3D model, the user is able to traverse. A navigation mesh consists of a set of connected convex polygons. To form these polygons neighboring walkable voxels are merged to regions. The borders of these regions are simplified to speed up the final navigation mesh process. The following sections present the navigation mesh generation process in more detail.

4.2.1 3D Model Rasterization

The rasterization stage of the navigation mesh generation process rasterizes the 3D model of the indoor environment into a 3D voxel grid called the voxel heightfield. The amount of voxels in the heightfield is defined with the width and height of a single voxel (parameter cell_size and cell_height in Section 4.1). Lower width and height of a voxel thus a higher amount of voxels in the voxel heightfield result in a better match between the walkable areas of the navigation mesh and the real indoor environment. The drawback of a denser voxel heightfield are longer navigation mesh generation times and higher memory cost. Figure 4.3a visualizes the 3D model and the corresponding 3D voxel heightfield of the 3D model in Figure 4.3b.

After the initial rasterization the single voxels in the voxel heightfield are merged to so-called spans. A span is formed from connected voxels in a column of the 3D voxel grid. These spans are used to split the 3D heightfield into two categories, the walkable and unwalkable spans. A navigation mesh represents the walkable surfaces of the 3D model, therefore, only walkable spans are used for further processing. The walkable surface of a span is represented by the top voxel of the span. The spans are divided using the following 3 characteristics:

- The distance between two unconnected spans in one column is bigger than the agents height(parameter agent_height in Section 4.1). (e.g. The user is able to stand on the floor (first span) without hitting the ceiling(second span)
- The slope of a ramp in the 3D model represented by two neighboring spans is smaller than a predefined value(parameter max_slope in Section 4.1). (e.g. A user

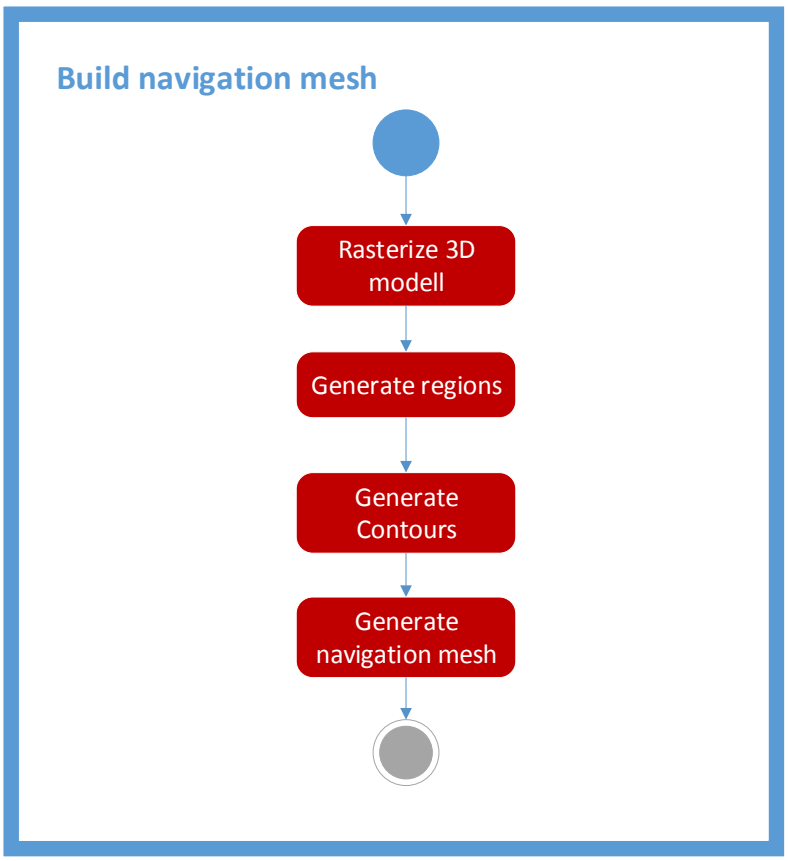
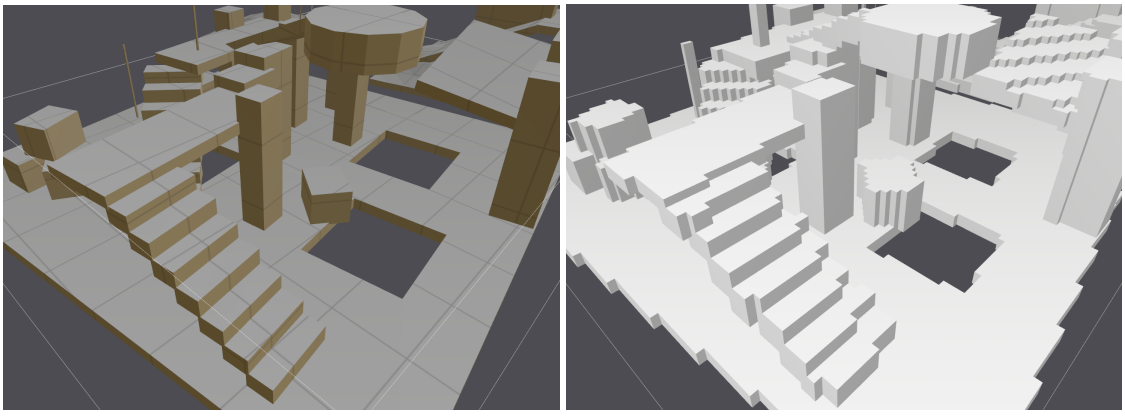


Figure 4.2: Activity diagram for the generation of a navigation mesh

is able to traverse a ramp from one span to the next)

- The height difference between two neighboring spans is smaller than a predefined value(parameter max_climb in Section 4.1). (e.g. An agent is able to traverse the ledge from one span to the next)

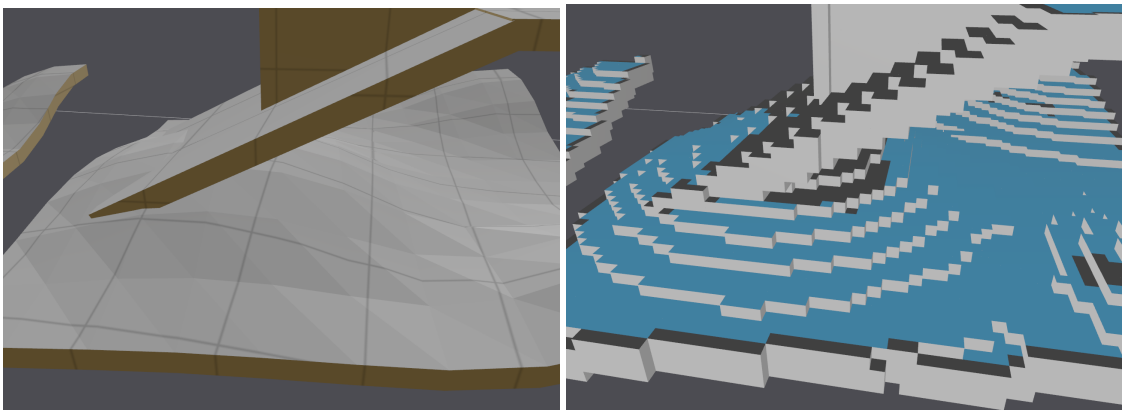
Figure 4.4a visualizes the 3D model and the corresponding walkable (blue) or unwalkable (gray) surfaces of the source geometry in Figure 4.4b. The area below the ramp close to the start of the ramp is gray because the height difference between the floor and the ramp is smaller than the agent_height. once the height difference is bigger than the defined agent_height the voxel color turns blue. In this particular scene the slope of the ramp is higher than the max_slope value thus the ramp is unwalkable. The gray voxels at the edge of the platform mark ledges and as a result the border spans are marked as unwalkable.



(a) 3D model of the indoor environment

(b) Voxel heightfield of the corresponding 3D model

Figure 4.3: Comparison between the 3D model of an indoor environment and its voxel heightfield.



(a) 3D model of the indoor environment.

(b) Voxel heightfield of the 3D model with walkable (blue) and unwalkable (gray) spans.

Figure 4.4: Walkable and unwalkable spans of 3D model.

This stage of the algorithm rasterizes the 3D model of the indoor environment into a voxel heightfield. The voxel heightfield is then split up into walkable and unwalkable spans and the walkable spans are used for further processing.

4.2.2 Region Generation

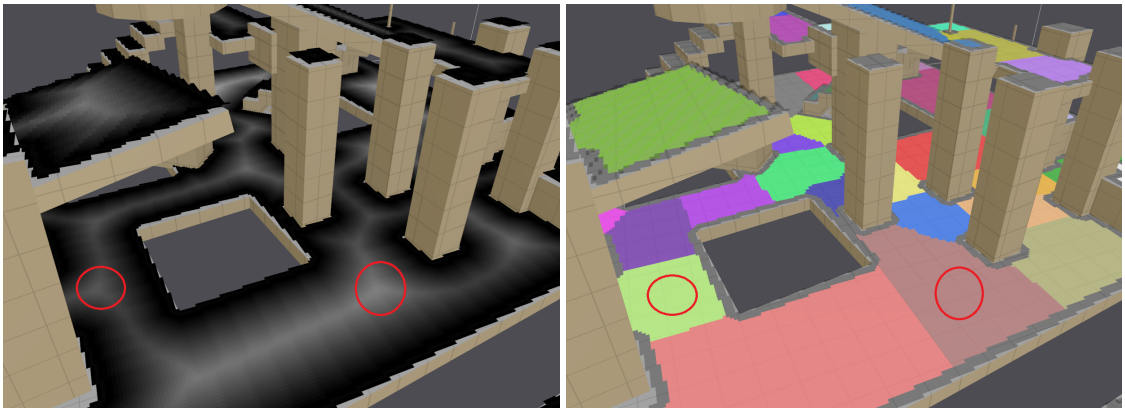
A recast navigation mesh is represented as a set of convex polygons. In this stage walkable spans are formed to walkable regions which are ultimately formed to convex polygons in later stages. At the beginning of the region generation stage separation to walls and other obstacles is assured for further processing. Every Recast navigation mesh provides

at least the user's radius(parameter agent_radius in Section 4.1) distance to the closest wall or other obstacles. As the FOV path is always inside the navigation mesh the user will never hit a wall or an obstacle traversing the FOV assisted path. All walkable spans which are closer than the user's radius to a wall or obstacle are marked as unwalkable.
The Recast library offers three different methods to generate regions. The implemented algorithm uses watershed partitioning, furthermore, Recast includes monotone partitioning and layer partitioning which are not discussed as part of this thesis. The watershed transform [11] is a method used for image segmentation [4, 22]. The method considers image information as a heightfield. The basins of the heightfield are center points of a region and are used as sources for a flooding process to fill the basins. During the flooding process two or more floods from different regions will merge. These border lines are defined as the region borders. The implemented watershed partitioning starts with calculating a distance field. The distance field represents the distance of a span to the closest obstacle using Chamfer distance transform [36, 5]. Chamfer distance transform is a two pass distance transform algorithm to approximate Euclidean distance. The resulting distance field is blurred to make it less prone to noise and generate less regions. The walkable span with the maximum distance of the entire distance field is the first basin of the watershed algorithm. The flooding process iterates through the distance values and assigns every walkable span to an existing connecting region. If the span has no connecting region a new region is generated.
After the flooding process is finished additional filtering is applied. Regions below a minimum size not connected to other regions are marked as unwalkable(parameter min_region_size in Section 4.1). Regions below a minimum size connected to other regions are merged with these regions(parameter merged_region_size in Section 4.1). Figure 4.5 shows the distance field of a 3D model and its corresponding regions. White areas in Figure 4.5a represent new region points for the watershed algorithm. These new region points are highlighted with red circles in Figure 4.5.

In this stage of the algorithm walkable spans are merged to walkable regions. These regions represent a basic partition of the walkable surface and are roughly correlated to the convex regions of the final navigation mesh.

4.2.3 Contour Generation

The contour generation stage detects border contours between walkable regions and simplifies those border contours to speed up the following navigation mesh generation stage. The borders of walkable regions consist of too many vertices (see staircase effect in Figure 4.6a), therefore, unnecessary vertices along region borders are excluded and simplified borders are generated. To simplify the contours the Ramer-Douglas-Peucker algorithm [38, 13] is used. The contour simplification starts with selecting the start and endpoints of contour edges between regions. A selection of mandatory points is highlighted with yellow circles in Figure 4.6. The simplification algorithm connects the mandatory vertices with a virtual line. All vertices along the contour edge are checked



(a) Distance field of the 3D model. (b) Walkable regions after watershed partitioning.

Figure 4.5: Distance field of a 3D model of an indoor environment and the corresponding walkable regions.

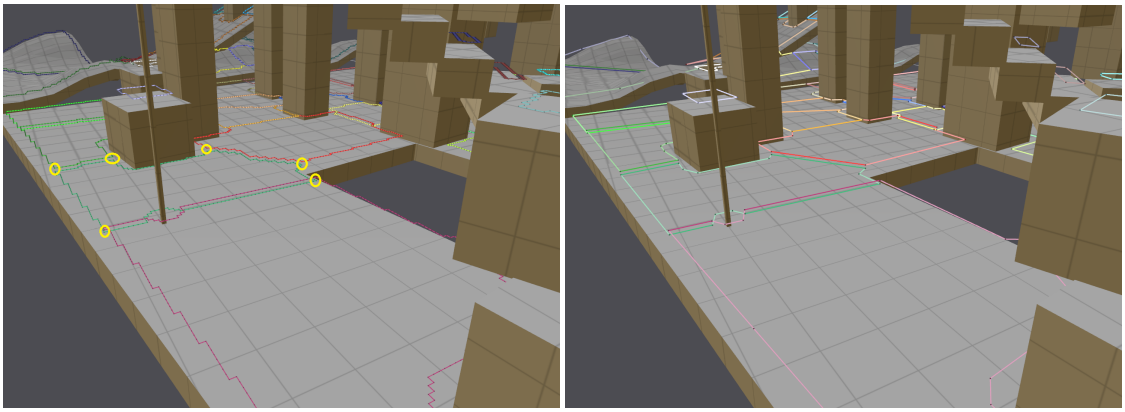
against the virtual edge. If the distance between a border vertex and the virtual line exceeds a predefined value(parameter max_edge_error in Section 4.1), the vertex is added to the virtual line. This continues until no vertex exceeds the deviation and the virtual line is the resulting simplified contour. A walkable region with simplified contours is called a simplified polygon.
In an additional step long thin contours are split up in shorter contour lines. The threshold for contour splitting is a predefined value as well(parameter max_edge_length in Section 4.1). The Recast Library only applies the contour splitting to contours at navigation mesh borders.

Figure 4.6 shows the contours of regions before and after simplification. Figure 4.6a highlights mandatory vertices used in the Ramer-Douglas-Peucker algorithm with yellow circles. The simplified contours are shown in Figure 4.6b. Every simplified polygon is shown in a different color. Edges that are shared between different regions have a higher color saturation.

This stage simplifies the borders of walkable regions to speed up the now following navigation mesh process.

4.2.4 Navigation Mesh Generation

This is the final stage of the navigation mesh generation process. Recast navigation meshes are represented in form of connected convex polygons and this information is stored in two data structures called PolyMesh and PolyMeshDetail. The PolyMesh data structure stores the convex polygons and neighborhood information to adjacent convex polygons. The PolyMeshDetail data structure contains triangle sub-meshes for every convex polygon in a PolyMesh and stores additional height information. Figure 4.7 shows



(a) Region contours before simplification process. (b) Region contour after simplification process.

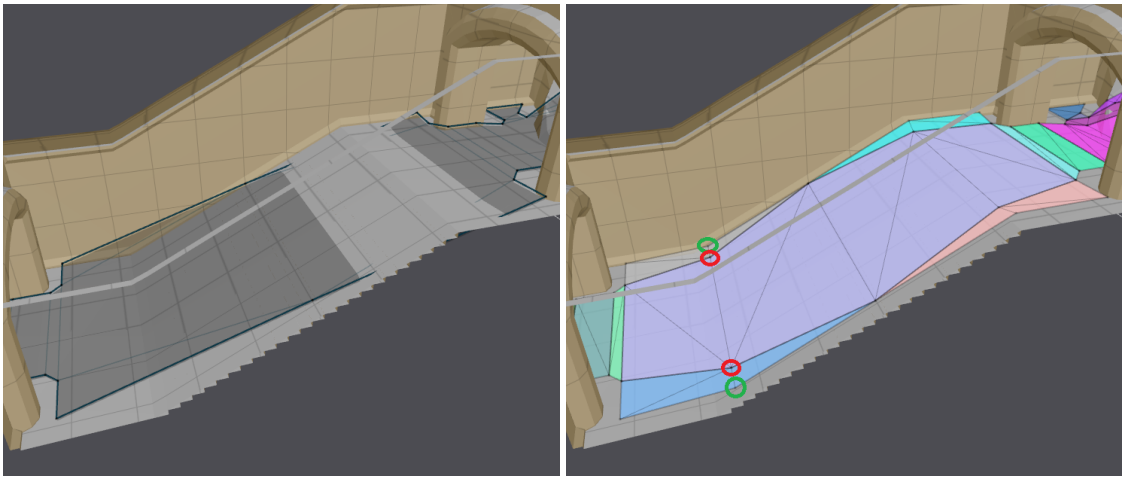
Figure 4.6: Region contours before and after the simplification algorithm

a visualization of both data structures.
The PolyMesh generation has two phases. At first, the simple polygons (convex and concave polygons) from the last stage are triangulated. The second phase takes triangles and polygons of already merged triangles and merges them to convex polygons using three conditions:

- Triangles or polygons of merged triangles must share an edge and the merging always starts with the longest shared edge.
- The resulting polygon must still be convex.
- The resulting polygon has a maximum amount of vertices(parameter verts_per_poly in Section 4.1).

Figure 4.7a shows a visualization of the PolyMesh information. The data structure stores all the vertices (black points) of the convex polygons, all the edges (thick and thin blue lines) and the neighborhood information for the convex polygons.
The triangulation for the PolyMesh is only done in the xz-plane, thus the PolyMesh might have a large offset from the original 3D model. Figure 4.7a visualizes the offset. On the lower end of the ramp the PolyMesh has a height offset from the 3D model and on the upper end of the ramp the PolyMesh cuts through the 3D model. To solve this offset problems, the PolyMeshDetail data structure is introduced.
The PolyMeshDetail subdivides the convex polygons of the PolyMesh to represent height offsets between the 3D model and the navigation mesh more accurate. The PolyMeshDetail generation starts with sampling the outer edges (thick blue lines in Figure 4.7a) of the PolyMesh (parameter sample_distance in Section 4.1) and adding vertices, where the height difference between sample point and the corresponding point in the 3D model exceeds a threshold (parameter max_sample_error in Section 4.1).

A triangulation algorithm is used to represent the 3D model with a set of triangles. The PolyMeshDetail data structure stores references to the vertices already used in the PolyMesh and adds the new generated vertices and triangles. The result of this process is shown in Figure 4.7b. Triangles with the same color belong to the corresponding convex polygon in a PolyMesh. Outer vertices (green circles) and inner vertices (red circles) are added at both sides of the ramp, thus the PolyMeshDetail data structure adds height information representing the 3D model more accurate.



(a) Visualization of convex polygons stored in the PolyMesh data structure

(b) Visualization of triangle sub-meshes stored in the PolyMeshDetail data structure.

Figure 4.7: Comparison of the PolyMesh and PolyMeshDetail data structure.

The navigation mesh generation process described in this section, generates one large navigation mesh representing the 3D model. The Recast Library also features tiled navigation meshes which represent the 3D model as tiled squares of sub navigation meshes. For the implemented algorithm the FOV navigation mesh is generated as a tiled navigation mesh as only tiled navigation meshes support dynamic obstacles.
This is the end of the navigation mesh generation process. The navigation mesh information is stored in the PolyMesh and PolyMeshDetail data structures. The Detour package of the library combines the PolyMesh and PolyMeshDetail information in one navigation mesh data structure called NavMesh. This NavMesh data structure is used in Section 4.3 for the FOV assisted path planning algorithm.

4.3 Implementation of the FOV Assisted Path Planning Algorithm

At this point a navigation mesh representing all the walkable areas of the indoor environment has already been generated. Furthermore, the current position and view direction of the mobile device and the destination are provided to the library. The position and size of dynamic obstacles are optional input values. For the remainder of this section the main contribution of this thesis, the implemented FOV assisted path planning algorithm is presented in detail. The aim of the work in hand is to calculate a path with clearance to walls and static obstacles, avoid dynamic obstacles and most important alter the path according to the view direction of the mobile device. The final path is calculated over several stages each introduced in detail as part of this section. Figure 4.8 shows the individual stages of the implemented path planning algorithm. The algorithm generates a path corridor first. The path corridor consists of all polygons along the shortest path connecting the user's current position to the destination. A Recast method is used to calculate the shortest path and the path corridor. At this point the Recast functionality is expanded by the author of this thesis to calculate the FOV assisted path. At first, a path improving separation to walls and static obstacles is calculated inside the corridor. This path is referred to as middle path for the rest of this thesis. The FOV assisted path consists of two concatenated paths. The first sub path represents the middle path through the FOV area of the user. The second sub path describes the middle path outside the FOV area. The implemented algorithm calculates the path inside the FOV area first. The position and view direction information of the mobile device is used to calculate the FOV area of the mobile device. A navigation mesh for the FOV area is generated to calculate the path inside the FOV area. The FOV navigation mesh also includes dynamic obstacles at this stage of the algorithm. After the calculation of the path inside the FOV is finished, a middle path towards the destination is created. This middle path represents the path outside the FOV area. According to the logic presented in Figure 3.8 the FOV assisted path is concatenated from the FOV path and the outside path. The final stage of the FOV assisted path planning algorithm is a path smoothing stage. In this stage the FOV assisted path is smoothed to provide a more intuitive looking path to the user. Figure 4.8 visualizes the standard pipeline to calculate the FOV assisted path.

4.3.1 Path Corridor and Shortest Path

The first stage of the FOV assisted path planning algorithm is the calculation of the path corridor from the user's current position to the destination. A path corridor consists of a sequence of navigation mesh polygons which represent the shortest polygon path from the current position to the destination. Only the polygons included in the path corridor are applied for further path planning. The path corridor represents the foundation for all subsequent path planning stages.
Recast uses an optimized A* algorithm implementation [10] to construct the path corridor. The A* algorithm is a graph searching algorithm introduced by Hart et. al. [20]. The

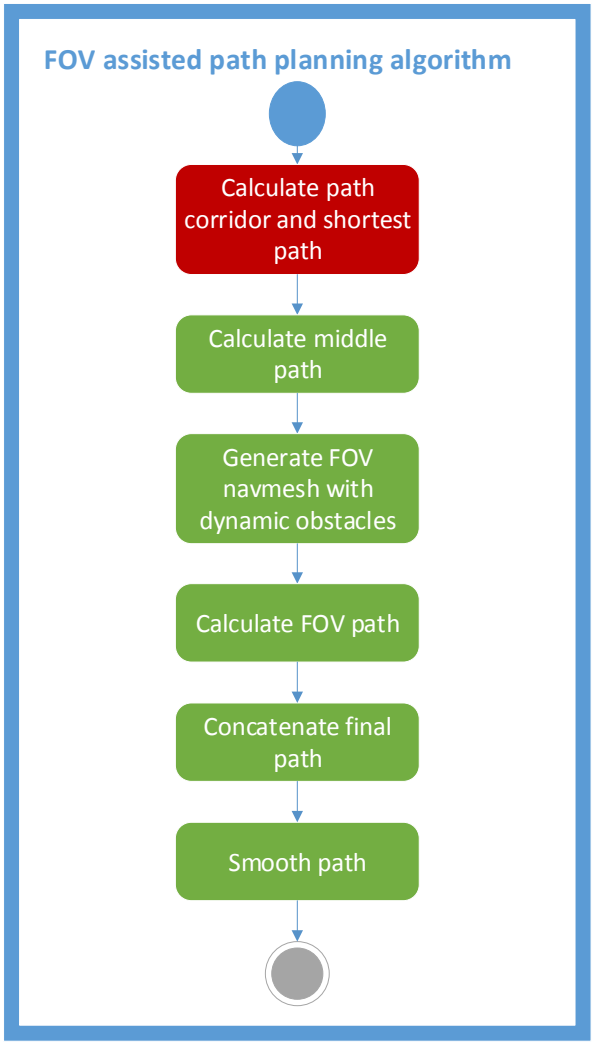


Figure 4.8: Activity diagram for the implemented path planning algorithm

algorithm combines known costs from a graph with estimated costs from a predefined heuristic to find the shortest path from a start node to a goal node. The cost function used for every node is:

$$f(x) = g(x) + h(x)$$

where $g(x)$ are the costs from the start node to the current node and $h(x)$ represents the estimated value from the current node to the goal node. In Recast the graph for the A* algorithm consists of the edge midpoints of border edges (called portals for the remainder of this paper) between two neighboring polygons. The edge midpoints of portals represent the nodes of the graph. Each polygon of the navigation mesh also provides information

to neighboring convex polygons in the navigation mesh (edges connecting the nodes). The Recast search graph is visualized in Figure 4.9. The cost function is based on the euclidean distance between the nodes. The optimized A* algorithm adds pre-allocated node lists for faster memory access and a priority queue to speed up the A* algorithm [10]. Figure 4.9 visualizes a path corridor. The dark yellow shaded polygons (including the polygons with the current position and the destination) visualize the path corridor. The brighter yellow polygons display polygons processed by the A* algorithm but not added to the path corridor. The blue polygons represent parts of the navigation mesh not used for path corridor calculation at all. The yellow graph shows the graph used in the A* algorithm. The nodes of the search graph visualize the edge midpoints of portals between two neighboring polygons. The red path displays the shortest path inside the path corridor from the current position to the destination.

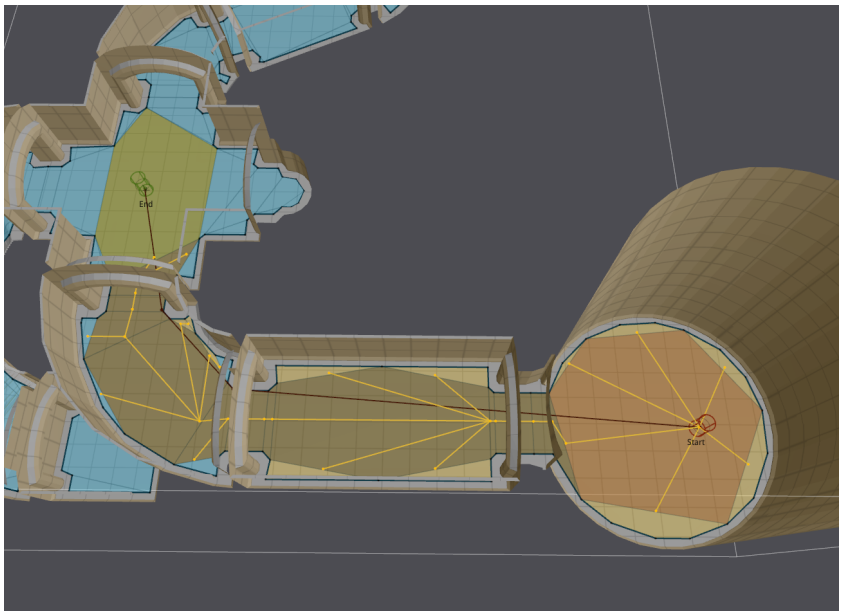


Figure 4.9: Path corridor with A* search graph

The funnel algorithm [7] is implemented in the Recast Library to calculate the shortest path. Figure 4.10 visualizes the steps of the funnel algorithm to generate a new vertex along the shortest path. The algorithm starts with creating a funnel (orange and blue lines in Figure 4.10) between the start point and the endpoints of the nearest portal (red dashed line in Figure 4.10). Portals are border edges between two convex polygons of the navigation mesh. The algorithm then iterates through all portals, creating new funnel edges between the start point and the portal end points and checks the new funnel edges against the existing funnel:

- Narrow the right/left side of funnel if the corresponding right/left testing edge is

inside the funnel (step 2,3 and 4 in Figure 4.10).

- Skip edge if the right/left testing edge is outside the corresponding right/left side of funnel (step 5 in Figure 4.10)
- Add the left funnel end point as path vertex if the right testing edge is on the left side of the funnel and vice versa with the right funnel end point (step 6 in Figure 4.10).

The new path vertex (image B in Figure 4.10) is the new start point and the algorithm continues until the end point is reached.

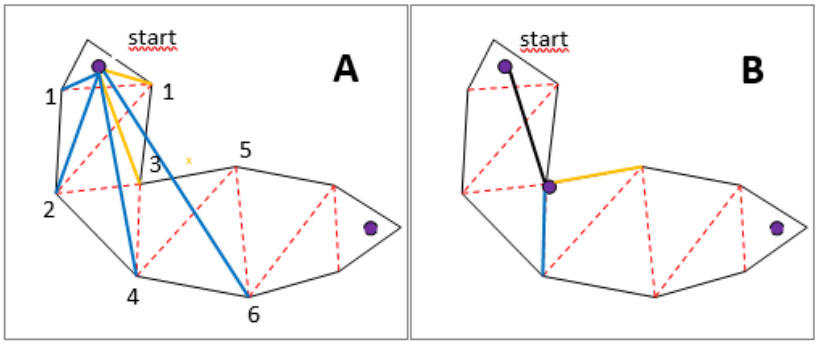


Figure 4.10: Steps of the Funnel Algorithm. Left image shows phases to first path vertex. Right image visualizes the first path vertex

In this section the path corridor and the shortest path representing the basis for the FOV assisted path planning algorithm are introduced. The path corridor is a chain of polygons which is the foundation for path planning algorithms like the shortest path and, moreover, for the FOV assisted path. The shortest path is one of the three return paths provided by the DLL to the Unity3D project. As long as a connection between the user and the destination exists, a shortest path is returned by the DLL. This path represents a viable fallback path if the calculation of the FOV assisted path fails.

4.3.2 Middle Path

Based on the path corridor a path with clearance to obstacles and static walls, referred to as middle path, is presented in this section. The middle path is calculated by traversing the path corridor and adding a vertex to the middle path on the edge midpoints of every portal of the path corridor. The blue path in Figure 4.11 shows the middle path and the blue points are the corresponding edge midpoints.

The advantage of this middle path calculation is that it is fast (see Section 5.1) and the middle path is in the middle of rooms and hallways with enough separation to walls and

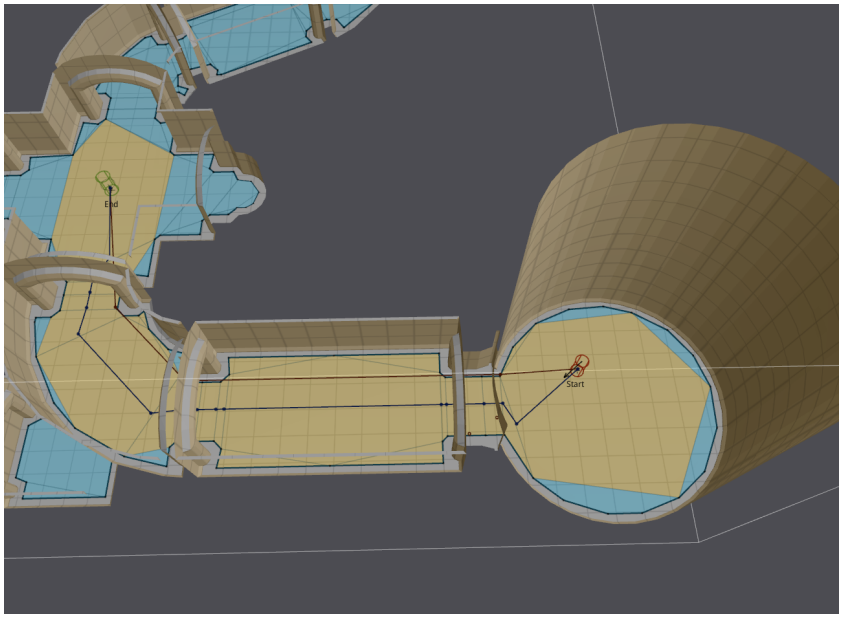
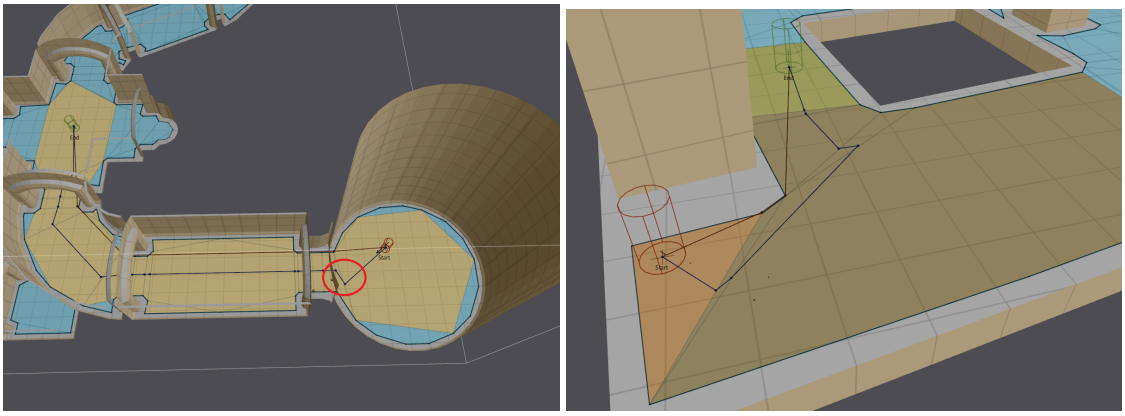


Figure 4.11: Middle path between current position and destination

obstacles. The disadvantage is a high dependence on the calculated navigation mesh and hence the path corridor. Figure 4.12 shows two scenarios in which the implemented middle path calculation shows its limitations. Figure 4.12a displays a detour of the middle path. Due to the fact that the middle path calculation adds a vertex on every edge midpoint of the path corridor, unfortunate path corridor portals may lead to detours. The smoothing algorithm from Section 4.3.6 solves the detour limitations of the middle path at the end of the algorithm, so that no countermeasures are introduced at this point. Figure 4.12b visualizes a middle path close to the border of the navigation mesh. As the path is still inside the navigation mesh, the path remains valid and has enough clearance for the user to follow the path without colliding with the wall. The navigation mesh is always at least the user's radius away from walls and obstacles. Hence as long as a path is inside the navigation mesh, the path is valid. Although the path remains valid, the unfortunate path corridor leads to minimum clearance. A potential workaround for unfortunate middle paths is to change the parameters (see Section 4.1) applied in the navigation mesh generation process.

The middle path adds clearance to walls and static obstacles which is a required feature for the FOV assisted path planning algorithm. The middle path is the second path provided by the DLL to the Unity3D project.



(a) Detour of middle path due to unfortunate navigation mesh

(b) Middle path with little clearance to navigation mesh boundary

Figure 4.12: Two scenarios where the middle path calculation shows its limitations

4.3.3 FOV Navigation Mesh with Dynamic Obstacles

At this point of algorithm the novel approach of the work at hand is added to the path calculation process. The path is altered to include the view direction of the mobile device. To implement the viewing direction an additional navigation mesh is generated. Whereas the main navigation mesh represents all walkable areas of the indoor environment, the FOV navigation mesh is only generated for the FOV of the mobile device. The global navigation mesh is only generated at the beginning of the FOV assisted path planning algorithm. The FOV navigation mesh is generated for each path calculation invocation. The 3D model of the indoor environment is rendered into a depth texture to generate the FOV navigation mesh. The render pass uses the position, view direction and FOV information of the mobile device to calculate depth information representing the mobile device's FOV. A navigation mesh only represents the walkable surfaces of the 3D model. The pixels from the depth texture are divided into floor pixels used for navigation mesh generation and other pixels no longer processed. The remaining floor pixels are filtered so that only the contour pixels of the depth texture remain (see blue pixels in Figure 4.14). These contour pixels are reduced again until only corner pixels of the contour remain. The remaining pixels are provided to calculate the FOV navigation mesh (Figure 4.14). Figure 4.13 shows a visualization of the depth texture pixels. White pixels are flagged as floor pixels and these floor pixels are applied for further processing. Red pixels on the other hand represent walls or other obstacles and are discarded. The border pixels of the depth textures are always flagged as red pixels. This states a requirement for the Moore-Neighbor tracing algorithm [35]. The Moore-Neighbor tracing algorithm and the Douglas-Peucker algorithm [38, 13] are used to reduce the floor pixels and speed up the FOV navigation mesh generation process.

The Moore-Neighbor tracing algorithm is used to generate the outer contour of the floor

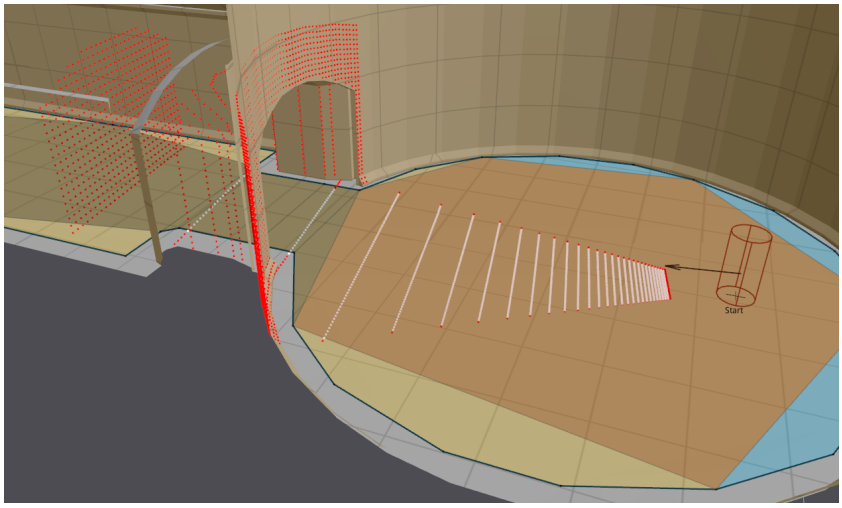


Figure 4.13: Projected depth texture to visualize FOV of user

pixels. The algorithm is based on the Moore neighborhood, which represents the 8 pixels sharing an edge or vertex with a middle pixel. At first, a start pixel (white floor pixel) for the contour is searched and added to the final contour. Then the algorithm checks one Moore-neighborhood pixel after the other in a clockwise rotation. The clockwise rotation stops when the first white floor pixel in the Moore-neighborhood of the starting pixel is detected. That pixel is added to the final contour and represents the next start pixel. This procedure is repeated until the initial start pixel is reached again. The result of the Moore-Neighborhood algorithm are all contour pixels of the floor pixels (see blue pixels in Figure 4.14).
The contour pixels are then reduced again in the contour generation described in Section 4.2.3 before a FOV navigation mesh is generated(see Section 4.2.4). Figure 4.14 shows the final FOV navigation mesh. The blue pixels are the contour pixels generated with the Moore-Neighbor tracing algorithm. The generated FOV navigation mesh is the foundation for the FOV path calculated in Section 4.3.4.

Contrary to the global navigation mesh representing the walkable surfaces of the entire 3D model of the indoor environment, the navigation mesh for the FOV also includes dynamic obstacles. Dynamic obstacles are obstacles not included in the static 3D model of the indoor environment. One of the requirements defined in Section 3.4 requires the FOV assisted path planning algorithm to incorporate dynamic obstacles in the path planning process. The Recast library only supports circular dynamic obstacles. Hence the position and radius of a dynamic obstacle are provided to the Recast Library via the DLL. Recast generates a navigation mesh without the dynamic obstacles in an initial pass and then adds dynamic obstacles to the navigation mesh. Figure 4.15 shows a FOV navigation mesh with two dynamic obstacles. The radius of the user is added to the radius of the dynamic obstacle, thus the algorithm always provides a minimum

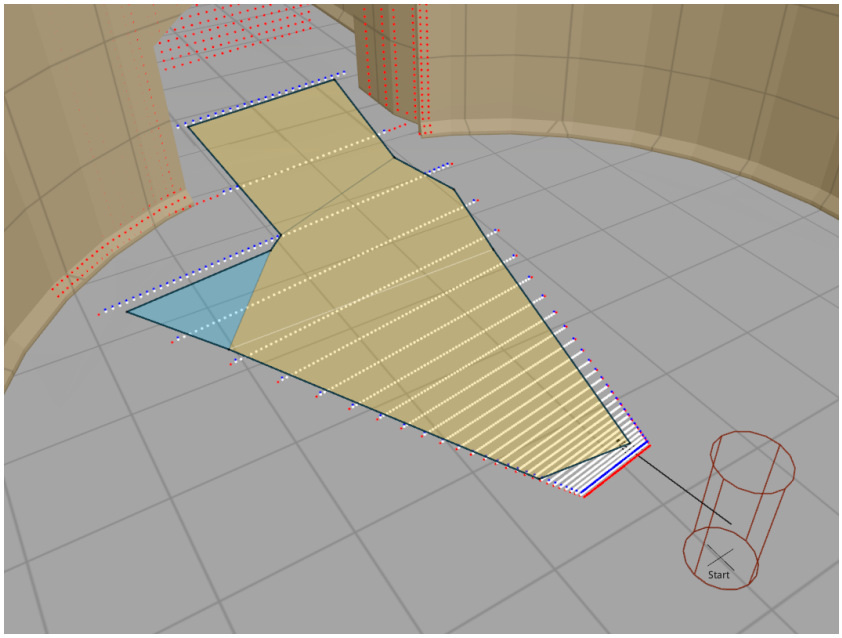


Figure 4.14: FOV navigation mesh with contour pixels

required clearance to dynamic obstacles. Generating the global navigation mesh is slow in comparison to the generation of the FOV navigation mesh. A runtime analysis presented in Section 5.1 reveals that it takes up to several seconds to generate the global navigation mesh, whereas the FOV navigation mesh generation only takes up to $100ms$. Generating two independent navigation meshes speeds up the path planning process, as the slow global navigation mesh remains static and is only generated once, whereas the FOV navigation mesh is generated continuously. The FOV navigation mesh is generated for every path calculation anyway, as it changes when the view direction of the mobile device changes. Thus adding dynamic obstacles to the FOV navigation mesh adds no additional navigation mesh generations. Furthermore, as the FOV assisted path planning algorithm only incorporates dynamic obstacles while inside the FOV, only dynamic obstacles inside the FOV have to be provided to the DLL.

4.3.4 FOV Path

Keeping the path inside the AR visualization device's FOV is the main incentive of the FOV path. The path is calculated based on the FOV navigation mesh generated in the last section. Another incentive for the FOV path is the seamless transition to the path outside the mobile device's FOV. A seamless transition provides an intuitive final path for the user. Three points of interest are calculated inside the FOV navigation mesh. The entry and exit point for the FOV are calculated to provide the seamless connection to the middle paths outside the FOV area. The centroid of the FOV navigation mesh

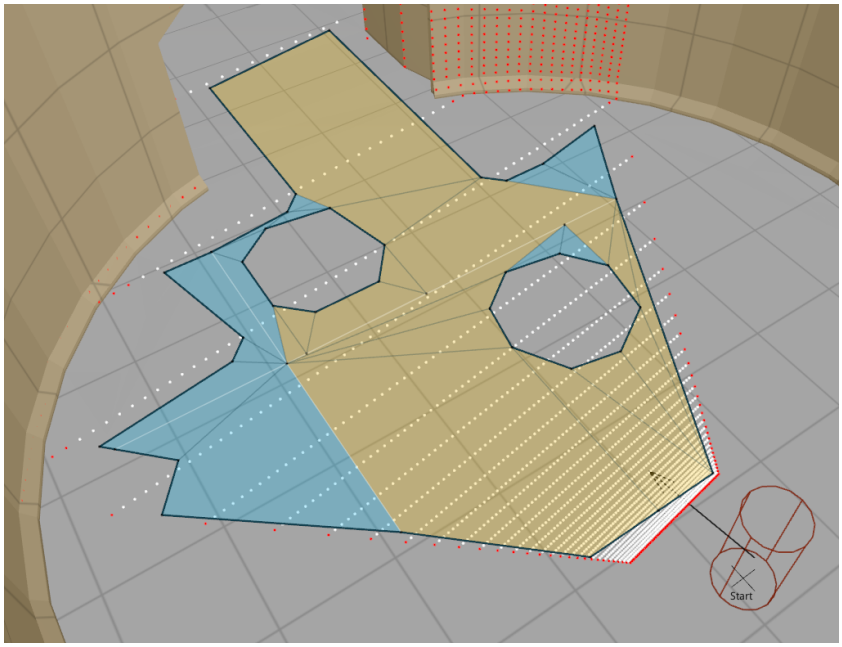


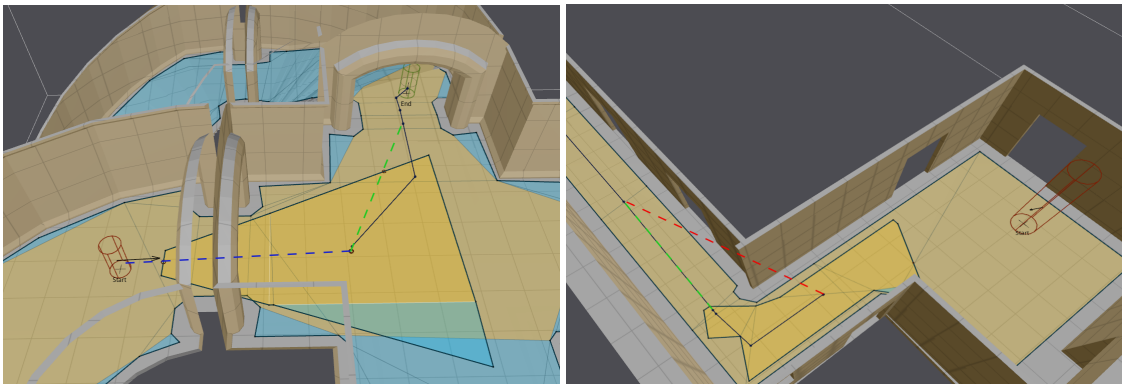
Figure 4.15: FOV navigation mesh with dynamic obstacles

represents a support point for the calculation of the other two points. The entry and exit point illustrate the start and end point of the path inside the FOV navigation mesh. Figure 4.16a displays the FOV navigation mesh with the three points of interest (red circles in Figure 4.16a).
At first, the centroid of the FOV navigation mesh is calculated. The PolyMeshDetail information calculated in Section 4.2.4 is used to calculate the centroid. The centroid of the navigation mesh is the sum of all the triangle centroids of the PolyMeshDetail representation provided by the navigation mesh. The entry point should be centered on the near side of the FOV to provide an intuitive path. The entry point is the intersection of the navigation mesh contour with the ray connecting the user's current position and the centroid of the navigation mesh (red dashed line in Figure 4.16a).
The calculation of the exit point is more complex. The exit point should provide a seamless connection between the FOV path and the middle path towards the destination. The exit point calculation starts with a middle path calculation from the centroid of the FOV navigation mesh to the destination (blue path in Figure 4.16a). The vertices of this middle path are then matched against the FOV navigation mesh and the first vertex outside the FOV navigation mesh is flagged for further processing. The calculation of the exit point is then separated between two cases:

- **Case A:** The exit point is the intersection of the navigation mesh contour with the line between the centroid and the flagged vertex outside the FOV navigation mesh (green dashed line in Figure 4.16a). Figure 4.16a shows the interest points

(red circles) and the intersections (red, green dashed line) used to calculate them.

- **Case B:** If the line between the centroid and the flagged outside vertex intersects a wall or obstacle of the 3D model of the indoor environment (see dashed red line in Figure 4.16b), the exit point is calculated with a different logic. In this case the exit point is the intersection of the line connecting the flagged outside point with the last point inside the FOV navigation mesh and the contour of the FOV navigation mesh (dashed green line in Figure 4.16b).



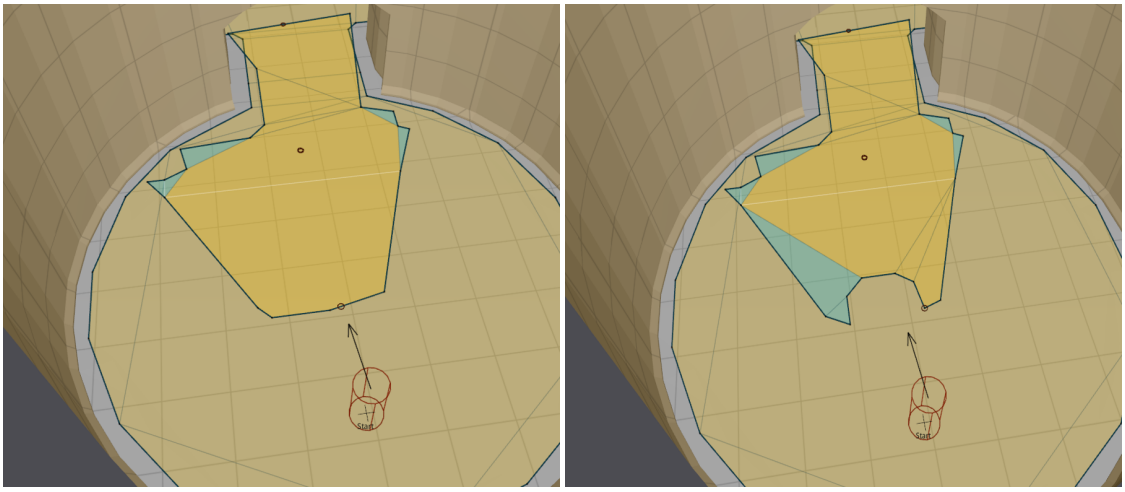
(a) FOV navigation mesh with points of interest. Exit point is calculated with case A

(b) FOV navigation mesh with exit point calculation for case B

Figure 4.16: FOV navigation mesh with points of interest

For dynamic obstacles an additional special case for the entry and exit point calculation is implemented. Dynamic obstacles are included in the FOV navigation mesh. The special case is invoked if the dynamic obstacle is on the border of the navigation mesh and overlaps with the position of the entry or exit points. Then the respective point has to be recalculated. Figure 4.17 shows a FOV navigation mesh without the dynamic obstacle(4.17a) and with a dynamic obstacle at the position of the entry point 4.17b). In this case the entry point is recalculated.

The entry and exit point is calculated twice to detect this special case. At first, the points are calculated for the FOV navigation mesh without dynamic obstacles and then again including dynamic obstacles. If the position of one of the points changed, that entry or exit point is recalculated. The Euclidean distance from a predefined point to every vertex of the FOV navigation mesh is calculated. The vertex with the lowest distance to the predefined point is the new entry or exit point. For the entry point the user's current position is the predefined point. For the exit point the first point of the middle path outside the FOV navigation mesh is the selected point. The vertex with the lowest distance to the user's position represents the new entry point. The vertex with the lowest distance to the point outside the FOV navigation mesh represents the new exit point.
Once the entry and exit points for the FOV navigation mesh are calculated, the middle path calculation (Section 4.3.2) is invoked to calculate the middle path inside the FOV



(a) FOV navigation mesh without dynamical obstacle

(b) FOV navigation mesh with dynamical obstacle at entry point

Figure 4.17: Navigation meshes with and without dynamical obstacles including respective points of interest

navigation mesh. Figure 4.18 shows the FOV assisted path (orange path) from the user's current position to the destination. The highlighted part of the path represents the FOV path inside the FOV area presented in this section. The concatenation of the FOV path with the path outside the FOV area is presented in the next section.

4.3.5 FOV Assisted Path

In this section the FOV assisted path from the user's current position to the destination is concatenated. Figure 3.8 shows all the different concatenation scenarios. The standard concatenation process results in a FOV assisted path consisting of the FOV path and the path from the exit point of the FOV area to the destination. The standard process is invoked when the view direction of the mobile device heads towards the same direction as the shortest path. The orange path in Figure 4.19 represents a visualization of the standard process of the FOV assisted path. The first part of the path is the FOV path (inside the green rectangle in Figure 4.19). The second part is the middle path from the exit point of the FOV to the destination (inside the blue rectangle in Figure 4.19). A special case for the FOV assisted path is invoked when the path destination is inside the FOV navigation mesh. In this case no exit point is necessary. A middle path from the entry point of the FOV area to the destination is calculated immediately.

Figure 3.7b displays the user's view direction heading towards the opposing direction than the shortest path hence the path corridor. For this special usecase a separate

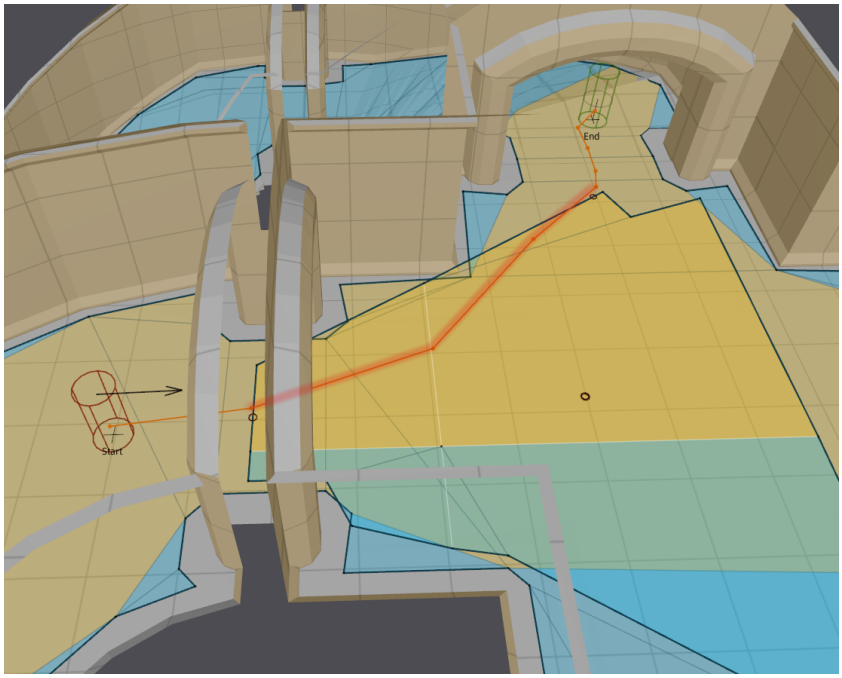


Figure 4.18: FOV assisted path (orange) with highlighted FOV path (red shaded)

path concatenation logic was implemented. The special concatenation is triggered once the angle between the user's view direction and the shortest path exceeds a predefined threshold. The threshold is an input value of the DLL and is provided by the user. Once the angle exceeds the threshold, the starting point for the path calculation changes from the user's current position to the centroid of the FOV navigation mesh. The path calculation is then split up into three parts:

- Middle path from the centroid of the FOV navigation mesh to the entry point of the FOV navigation mesh. The calculation of the entry point is described in Section 4.3.4
- Middle path from the entry point to the user's current position
- Middle path from the user's current position to the destination

The three sub paths are concatenated to form the FOV assisted path. Figure 4.20 shows a path calculated with the special concatenation logic (orange path). The gray path is the smoothed version of FOV assisted path. The smoothing algorithm is presented in the next section. The final special case of Figure 3.8 called wall path is introduced in Section 4.3.7.

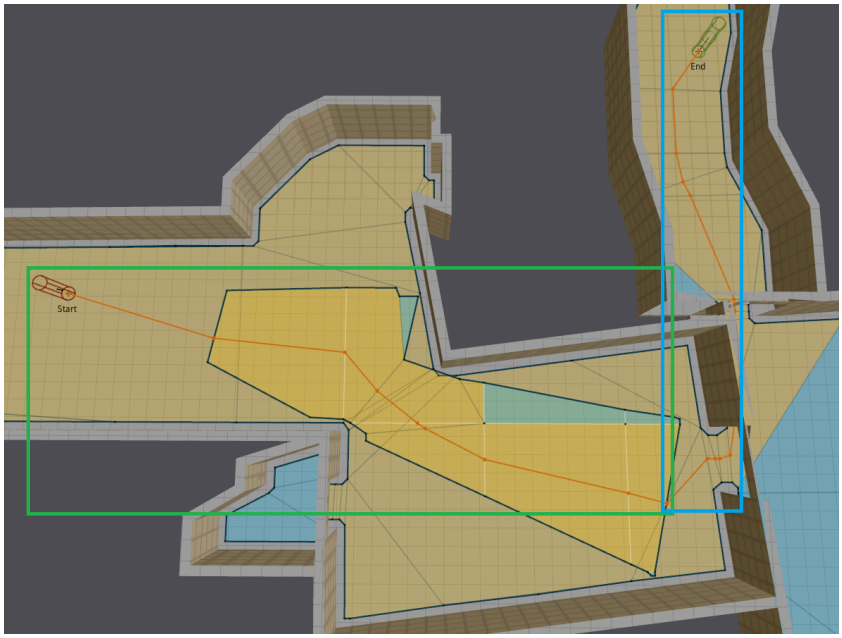


Figure 4.19: Two sub paths (green rectangle highlights the FOV path, blue rectangle highlights the finish path) of the FOV assisted path.

4.3.6 Path Smoothing

The final stage of the FOV assisted path planning algorithm represents a path smoothing stage. The FOV assisted path is smoothed to provide a more intuitive looking path to the user. The implemented smoothing algorithm is related to an algorithm used for local movement presented by Geraerts in his corridor map method paper [15].
The path smoothing is based on an attraction point which proceeds along a path towards the destination. This attraction point forces the user towards the destination. The backbone path, on which the attraction point proceeds, corresponds to the FOV assisted path with waypoints added at a constant distance. For every waypoint on the backbone path the distance to the nearest obstacle or wall is calculated. The attraction point $\alpha(x)$ is a waypoint along the backbone path. The attraction point represents the point with the highest possible distance to the user's current position in relation to the closest obstacle of the attraction point. In other words, the distance between the attraction point and the user has to be smaller than the distance between the attraction point and the nearest obstacle including the user radius:

$$euclideandistance(x, B[t]) = R[t] - r, \tag{4.1}$$

where x is the current position of the user, $B[t]$ a candidate attraction point on the backbone path, $R[t]$ the distance of that point to the nearest obstacle or wall and r is the radius of the user. Figure 4.21 visualizes an attraction point (white dot). The brown path is the backbone path with the waypoints(brown dots). The brown circles

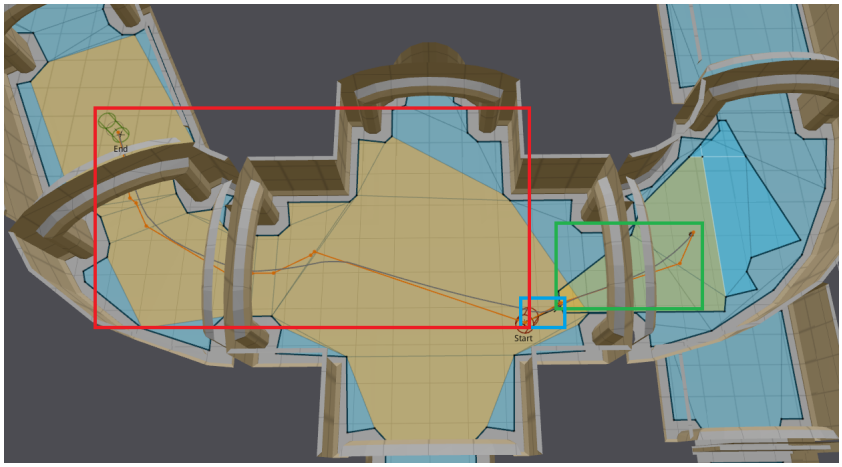


Figure 4.20: Unsmoothed (orange) and smoothed (grey) FOV assisted path calculated with the special concatenation logic.

visualize the radii of circles to the nearest obstacles. In Figure 4.21 only the first five radii are visualized for a clearer visualization. In a loop every waypoint is verified with Equation 4.1. The loop stops at the white dot. The white dot on the backbone path marks the attraction point including the white circle for the distance $R[t]$ to the closest obstacle. The white circle surrounds the gray circle which represents the user's current position including the user's radius. The attraction point is defined as the point on the backbone path, the furthest away from the user, where the white circle still surrounds the gray circle representing the radius of the user.

After the attraction point calculation a force pulling the user towards the attraction point is calculated:

$$F = f\frac{\alpha(x) - x}{\|\alpha(x) - x\|}, \tag{4.2}$$

where $\alpha(x)$ represents the position of the attraction point and x expresses the position of the user. f describes a parameter related to the Euclidean distance d between the attraction point and the user. f is small when the user is close to the attraction point and large when the user is nearly $R[t]$ away from the attraction point:

$$f = \frac{1}{R[t] - r - d} - \frac{1}{R[t] - r}, \tag{4.3}$$

where $R[t]$ represents the distance to the nearest obstacle of the attraction point, r expresses the radius of the user and d describes the Euclidean distance between attraction point and user.
Integrating the force F over time leads to the new position of the user. In every iteration a new attraction point, a new velocity and a new position for the user is calculated. A

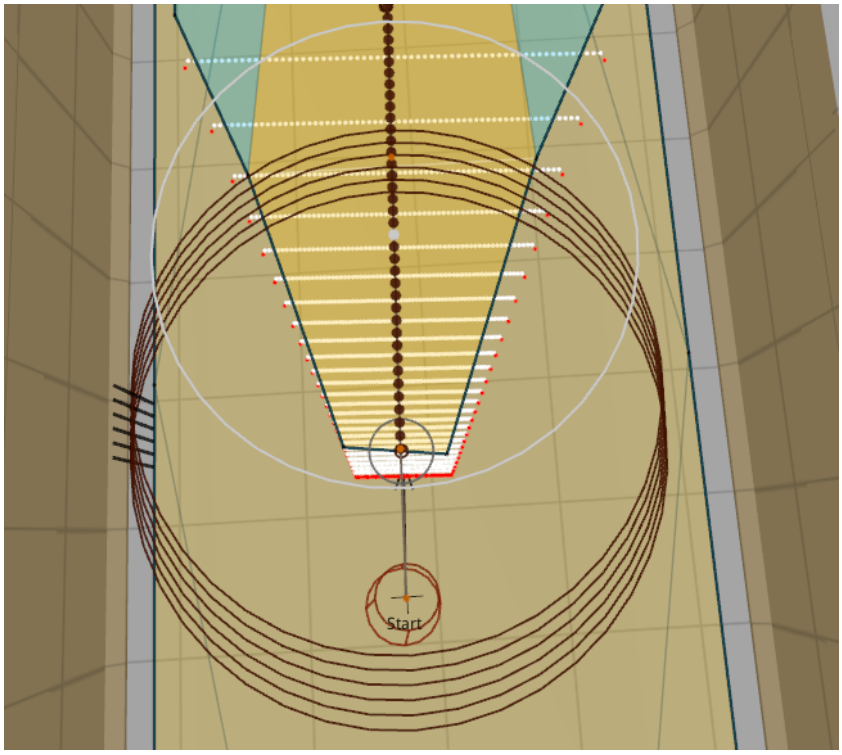


Figure 4.21: Visualizes the calculation of an attraction point

velocity form [43] of the Verlet integration method [51] is used to compute the smooth path:

$$x_{t+\vec{\Delta t}} = \vec{x_t} + \vec{v_t}\Delta t + \frac{1}{2}\vec{a_t}\Delta t^2, \tag{4.4}$$

$$v_{t+\vec{\Delta t}} = \vec{v_t} + \frac{\vec{a_t} + a_{t+\vec{\Delta t}}}{2}\Delta t, \tag{4.5}$$

where x_t is the position of the user, v_t is the velocity of the user and due to Newton's laws of motion where $F = m * a$ with a mass $m = 1$, a_t corresponds with the force F. The maximum velocity of the user is capped at $1.2m/s$. This speed represents the average walking speed of a human [30]. The path smoothing ends when the user reaches the perimeter of the destination. This perimeter is defined by the user (see Section 4.1). The following listing summarizes the smoothing process:

1. Calculate backbone path B with waypoints $B[t]$ and distance to nearest obstacle $R[t]$ for every waypoint
2. Calculate new position using velocity verlet (Equation 4.4)
3. Calculate attraction point for the new position (Equation 4.1)
4. Calculate force to attract the user (Equation 4.2)

5. Calculate velocity (Equation 4.5)

6. Go to 2 until reaching the destination

Figure 4.22 visualizes the final smooth FOV assisted path of the FOV assisted path planning algorithm. The orange path is the FOV assisted path before the path smoothing algorithm and the gray path is the smoothed FOV assisted path. The smoothed FOV assisted path represents the result of the implemented algorithm and fulfills the requirements defined as part of this thesis. The smoothed FOV assisted path is the final path returned by the DLL.

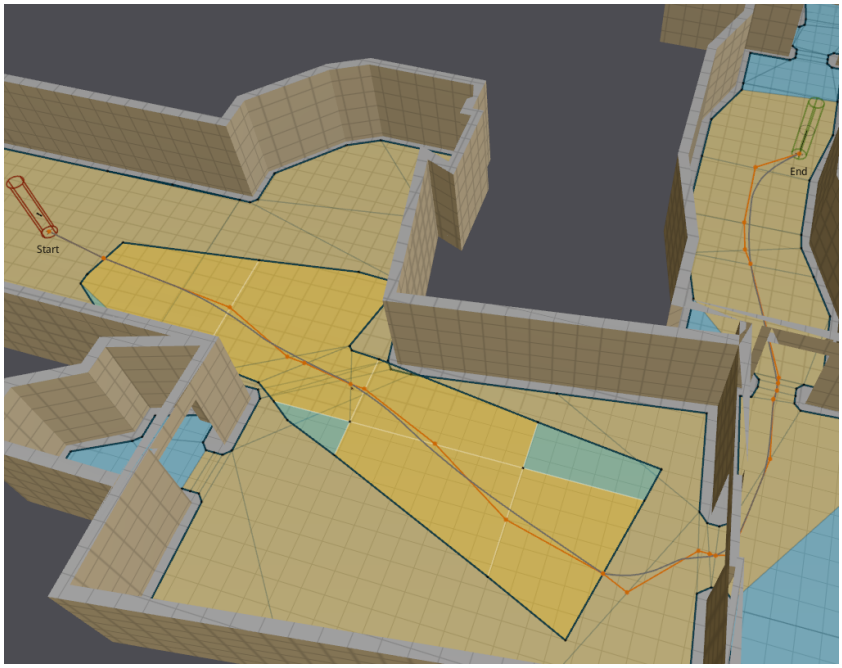
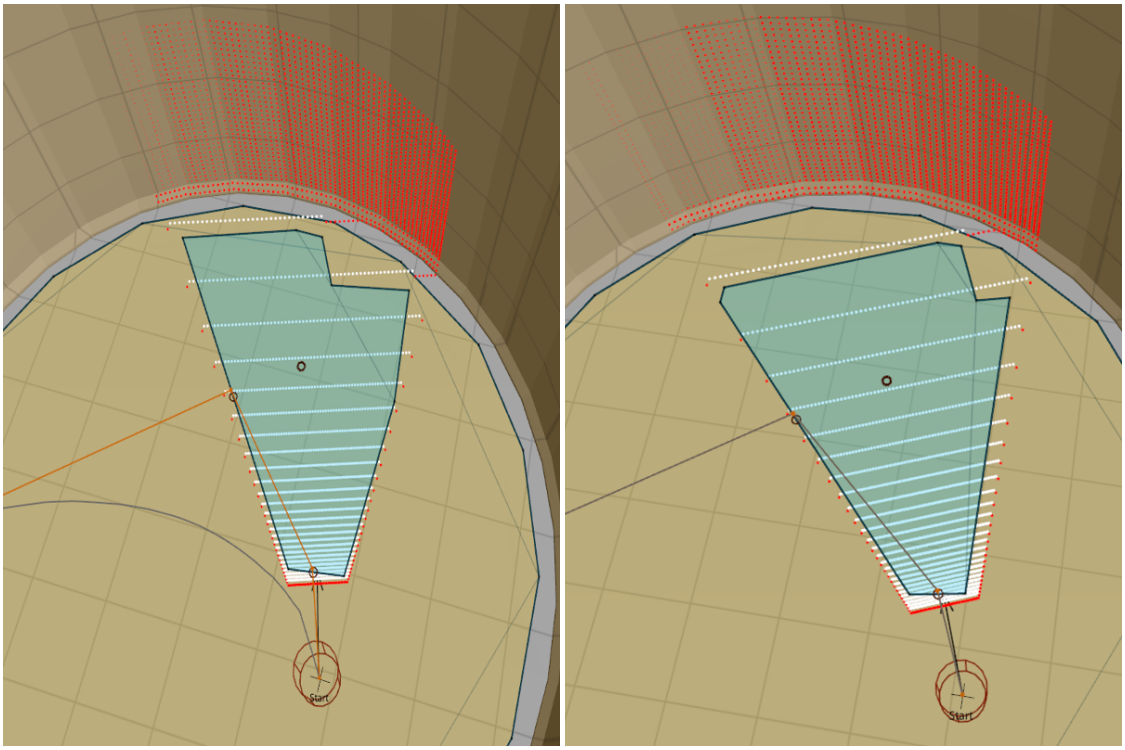


Figure 4.22: The final smooth path of the FOV assisted path planning algorithm

As mentioned in Section 4.3.4 the path smoothing algorithm is applied on two sub paths. Keeping the path inside the user's FOV is a main requirement for the implemented FOV assisted path planning algorithm. Therefore, the FOV path and the path outside the FOV area are smoothed in two separate steps. Figure 4.23 visualizes the result of the path smoothing algorithm, when the smoothing is applied on a single FOV assisted path. The smoothed path avoids the FOV area of the mobile device and thus the user is unable to see the path without adjusting the view direction.

4.3.7 Wall Path

The definition of a navigation mesh states that a navigation mesh represents only the walkable surfaces of the 3D model of an indoor environment. Inside of buildings users



(a) smoothing for entireFOV assisted path (b) smoothing split up in two sub parts

Figure 4.23: Comparison of the smooth FOV assisted path with and without two part calculation

might look towards a wall or an obstacle and have no walkable surface inside their FOV. In this special case the path is concatenated from two different paths. At first, a connection from the wall or obstacle to the global navigation mesh of the indoor environment is calculated. Inside the navigation mesh a middle path towards the destination is calculated. This middle path is smoothed and returned as a smoothed FOV assisted path. For the remainder of this section this path is called wall path.
The wall path calculation is activated when the FOV navigation mesh generation process returns no valid FOV navigation mesh. When there are not at least 3 contour pixels or no floor pixels at all inside the FOV depth texture (see Section 4.3.3) the FOV navigation mesh generation fails. Three points represent the connection to the navigation mesh. These three points also describe the first three points of the wall path:

1. When no FOV navigation mesh is calculated, the middle pixel of the depth texture represents the first point of the wall path. This point represents the center of the FOV of the mobile device.

2. The second point of the wall path illustrates the intersection of the ray between the user's current position and the middle pixel of the depth texture with the

navigation mesh boundary. This guides the path from the center of the user's FOV to the floor hence the walkable surface of the 3D model of the indoor environment.

3. The third point represents a point, at least the user's radius away from the second point and the navigation mesh border. The third point is necessary to prevent errors in the smoothing stage 4.3.6 of the path calculation.

The third point represents the starting point for the middle path calculation and the resulting middle path is smoothed to get the final wall path. Figure 4.24 visualizes the wall path (gray line). The first 3 points are marked with numbers. The wall path logic is activated because all depth texture pixels are on an unwalkable surface(red pixels).

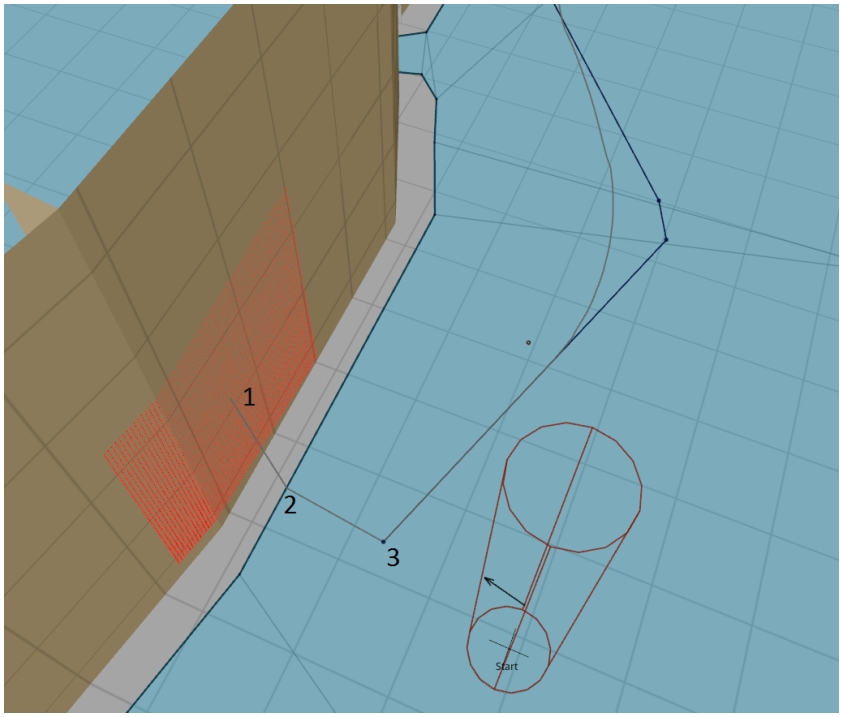


Figure 4.24: Visualization of the wall path logic with the first 3 wall path points and the remaining part of the wall path (gray)

4.4 Unity3D Implementation

This section presents a special path recalculation logic implemented in Unity3D and the visualization of the results in Unity3D. The recalculation logic controls how often the path calculation of the FOV assisted path planning library is invoked. For the user study conducted as part of this thesis (see Section 5.2), calculating a new path for every frame is not required. The recalculation logic is based on the visible path points of the FOV assisted path inside the FOV of the mobile device in relation to the sum of all path points

in a predefined area around the user. Figure 4.25 visualizes the path recalculation logic. The idea is to invoke the path calculation when the path visible inside the mobile device's FOV is below a user defined threshold. Turning away from the path results in less path points inside of the FOV of the mobile device. The implemented logic counts all path points inside a predefined radius (yellow and cyan lines in Figure 4.25) and separates points inside the FOV (yellow lines in Figure 4.25) from points outside the FOV (cyan lines in Figure 4.25). The borders of the FOV of a mobile device are visualized with blue and green lines in Figure 4.25. In addition to the path points outside the FOV, path points inside the FOV which are occluded by walls or obstacles are flagged as outside the FOV. The ratio of points inside the mobile device's FOV to all points inside the predefined radius is used to decide whether the path is recalculated or not. If the ratio is above a predefined threshold the path is recalculated. When the ratio is below the threshold the old path is still valid. The threshold is defined in the Unity3D project.

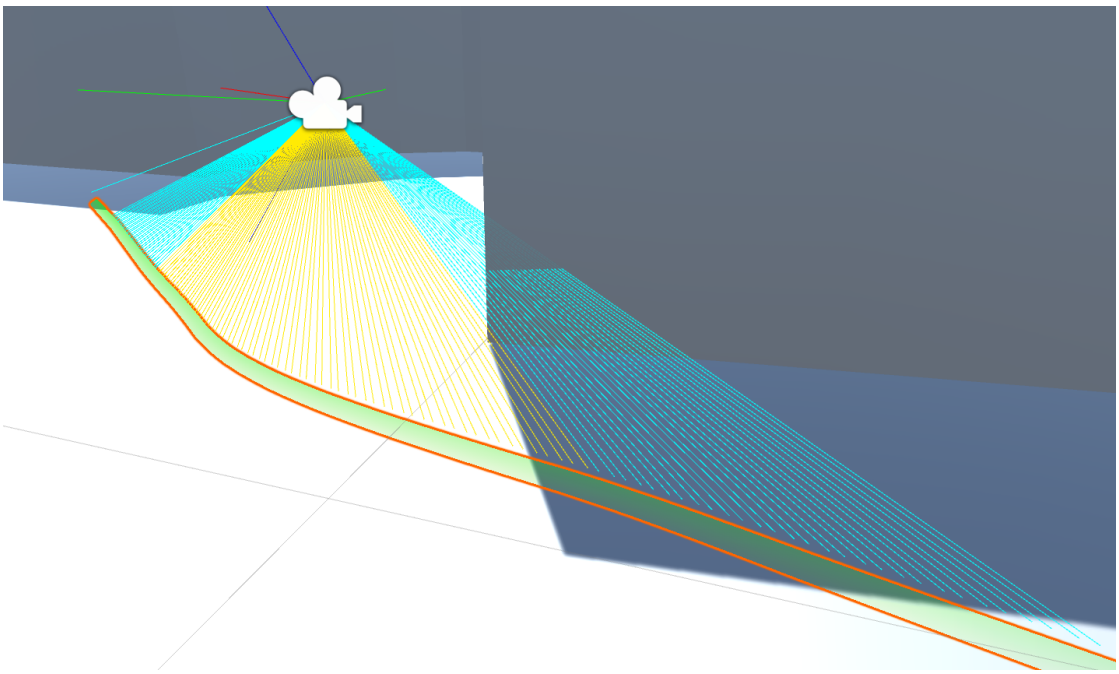


Figure 4.25: Path recalculation logic in Unity. The image shows the user's view direction (red line) with the borders of the FOV (blue, thin green line), the path (thick green line) and the path point rays (yellow and cyan lines) which are used for the path recalculation logic

The Unity3D project also includes a visualization of the FOV assisted path. The Unity3D project applies permanent lines from the user's position to the destination. The path is visualized slightly above the floor of the indoor environment. The implemented path planning library returns three different paths, the smoothed FOV assisted path, an

unsmoothed middle path and an unsmoothed shortest path. If the algorithm is unable to calculate the smoothed FOV assisted path a smoothed middle path is returned in addition to the two unsmoothed paths. Figure 4.26 visualizes the three different paths returned from the FOV assisted path planning algorithm. The green path represents the smoothed FOV assisted path, the blue path represents the middle path and the yellow path represents the shortest path. At the transitions between the FOV area and the rest of the environment the path is not smoothed. This emerges due to the smoothing problem described in Section 4.3.6. The path at the entry point into the FOV (Point 1 in Figure 4.26) and at the exit point out of the FOV (Point 2 in Figure 4.26) is not smoothed. The path inside the FOV area and from the exit point to the destination is smoothed. In addition to the path visualization, methods to display the navigation mesh and the FOV area are provided in the Unity3D project.

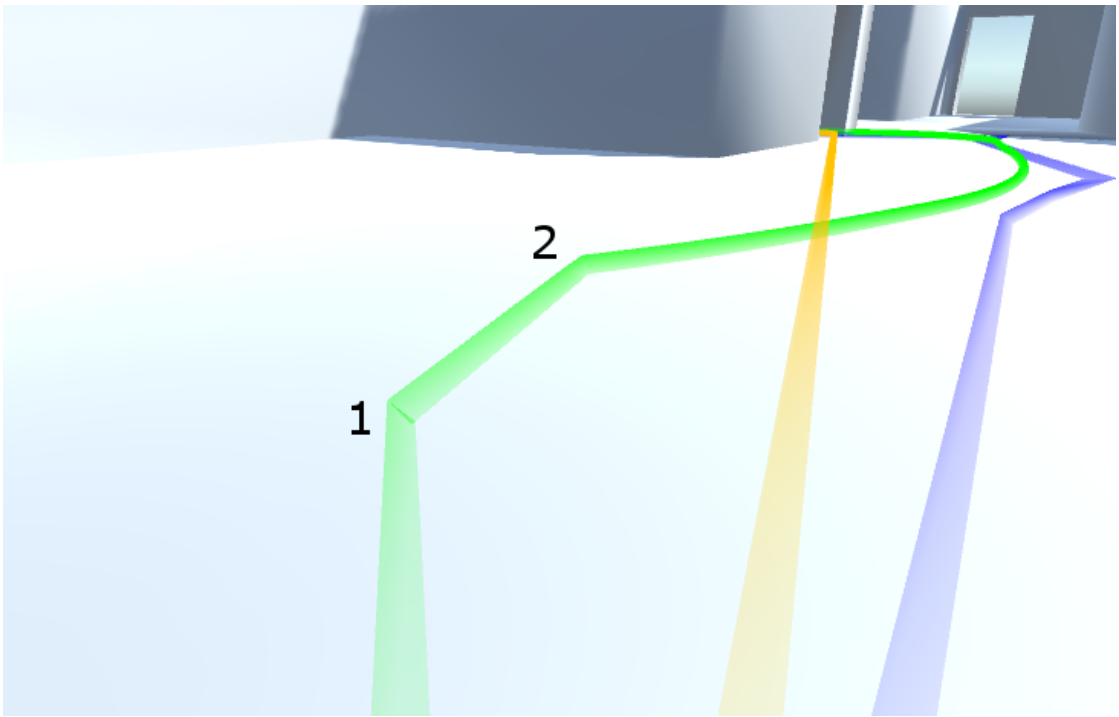


Figure 4.26: Visualization of the three paths returned from the FOV assisted path planning algorithm. Shortest path (yellow), middle path (blue) and FOV assisted path(green)

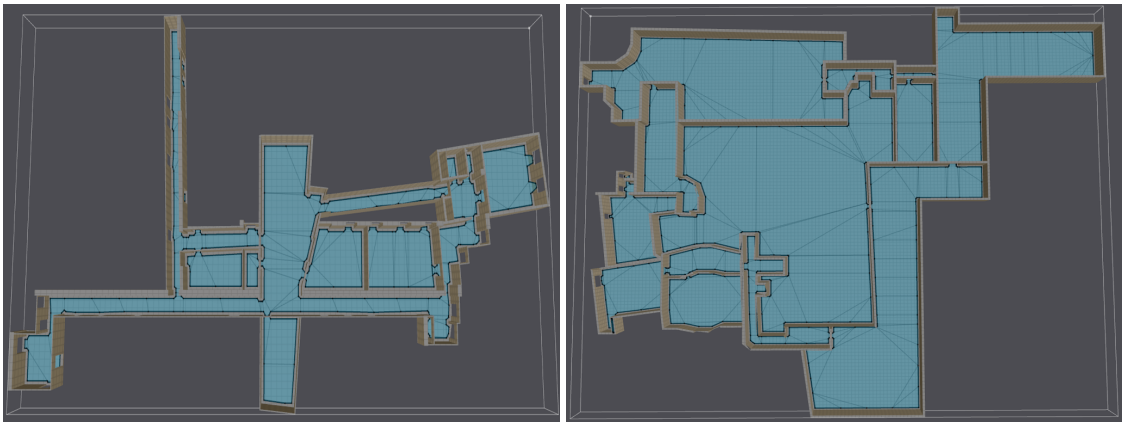
CHAPTER 5

Results

This chapter describes the evaluation of the implemented FOV assisted path planning algorithm. In Section 5.1 a detailed performance analysis is conducted. The effects of different indoor environments and cell sizes (see Section 4.2.1) on the generation of the navigation mesh and the path planning algorithm are discussed. Furthermore, the performance evaluation presents the impact of the FOV area size and the path length on the path calculation duration. At last, the runtimes of the different implementation stages presented in Section 4.3 are compared. The usability of the implemented algorithm was evaluated in a user study conducted as part of this thesis. The results of the user study are presented and discussed in Section 5.2.

5.1 Performance Analysis

The performance of the FOV assisted path planning library is evaluated in performance tests presented in this section. Performance requirements vary between different areas of application of an algorithm. The minimal performance requirement for the implemented FOV assisted path planning algorithm depends on how often the path needs to be recalculated and how fast the path reacts to changes of the view direction. For different areas of application the amount of recalculations can change between every frame, only once per second or the recalculation is only triggered on special events like the recalculation logic presented in Section 4.4. The path recalculation has to provide a correct registration of the path in relation to the real world.
The following runtime evaluations are performed on the C++ FOV assisted path planning library as it provides the key functionality of the algorithm. The algorithm is evaluated on a notebook with an Intel $i7-4940MX 3.30GHz$ Quadcore CPU with 16.0GB RAM and a Geforce GTX $880M$ GPU with 8GB RAM. For the evaluations two 3D models of an indoor environment at the Technical University of Vienna are selected (see Figure 5.1).



(a) 3D model Institut 3rd floor TU Vienna (b) 3D model Institut ground floor TU Vienna

Figure 5.1: 3D models used for runtime analysis

In Table 5.1 the duration of the navigation mesh generation process is compared with the duration of the path planning process. For both models the FOV area and the path length are constant at $8m^2$ and $45m$.
Table 5.1 shows that the generation of the navigation mesh for the entire 3D model of the environment is much slower than the path planning algorithm. The duration of the navigation mesh generation process depends on the cell size and cell height of a voxel used to generate the navigation mesh(see Section 4.2.1) and the size of the walkable area of the 3D model. The smaller the cell size the longer it takes to generate the navigation mesh due to a higher amount of processed voxels. For the second model it takes up to 60 seconds to generate the navigation mesh for a cell size and height of $20cmx20cm$. All of the navigation mesh generation times in Table 5.1 show that the navigation mesh generation is too slow for real-time path planning. To assure real-time performance the navigation mesh generation is independent from the path planning algorithm. The navigation mesh is only generated at the beginning of the implemented algorithm and forms the basis for the continuous calculation of the FOV assisted path. For the two evaluated 3D models the path calculation times are stable for cell sizes between $50cm$ and $80cm$ and slow down for very small cell sizes.

The path length and the FOV area are the main focus of the path planning algorithm analysis. Table 5.2 and Table 5.3 show results for variable path length with constant FOV area and vice versa. Both tests are conducted on the 3D model in Figure 5.1b. The cell size and cell height are constant at $50cm$.
Table 5.2 shows the results for constant FOV area at $10m^2$ and a variable path length between $10 - 160m$. The path length represents a minor factor in the calculation times for the FOV assisted path planning algorithm. For a path length of up to $80m$ the FOV assisted path planning algorithm computes 10 new paths per second. 10 path recalculations per second are enough to provide a correct registration of the path in relation

Map	#verts	#tris	#cell size/ height (cm)	navmesh generation(ms)	path calculation(ms)
Institut 3rd floor	11.0k	9.7k	80x80	391.6	50.4
Institut 3rd floor	11.0k	9.7k	50x50	974.3	61.8
Institut 3rd floor	11.0k	9.7k	20x20	6317.8	177.5
Institut EG	1.0k	2.2k	80x80	2988.3	56.8
Institut EG	1.0k	2.2k	50x50	7872.4	61.8
Institut EG	1.0k	2.2k	20x20	58829.5	145.9

Table 5.1: Runtime comparison between navigation mesh generation process and path calculation

to the real world. This means that when the mobile device is rotated and, therefore, the view direction changes, the calculated path is visualized on the AR visualization device withouth any noticeable delay. For long path lengths of over $100m$ the path calculation slows down. For such long distances a limitation of the path smoothing is a possible solution. The algorithm could calculate the path corridor up to the destination to calculate a traversable path but restrict the path smoothing up to a predefined value.

Map	FOV area (m^2)	path length(m)	path calculation(ms)
Institut EG	10.0	10.8	59.0
Institut EG	10.0	32.3	67.6
Institut EG	10.0	46.3	75.2
Institut EG	10.0	79.1	99.1
Institut EG	10.0	113.1	142.1
Institut EG	10.0	159.0	219.0

Table 5.2: Runtime comparison with variable path lengths and fixed FOV area

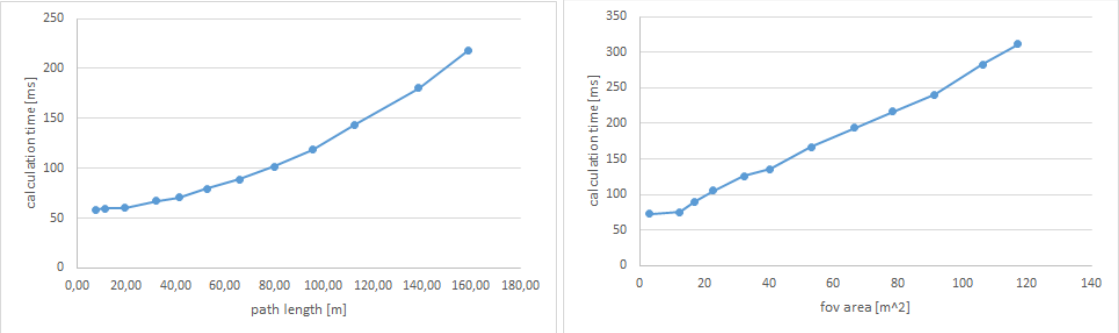
Table 5.3 shows the results for constant path length of $50m$. The simulated FOV has a horizontal and vertical FOV of 45 degrees and a far view plane of $20m$. The results show that for a FOV area of around $20m^2$ the calculation time is already at $100ms$. The size of the FOV area is the main factor for the calculation time of the FOV assisted path planning algorithm. In large open areas, and thus a large FOV, the calculation time is up to $300ms$. The solution to speed up the navigation mesh generation for large open areas is to restrict the size of the FOV area.

Figure 5.2 shows two graphs with variable path lengths and a constant FOV area and vice versa. For continuous path recalculation at least 10 recalculations per second, hence a calculation time of up to $100ms$ are enough to visualize the path without visible artifacts. In Figure 5.2a path lengths of up to $80m$ are calculated in up to $100ms$. Path calculations in large single-story buildings or path calculations on multistory levels are possible in relation to the path length.
The size of the FOV area is the bottleneck of the algorithm. For buildings with narrow

Map	FOV area (m^2)	path length(m)	path calculation(ms)
Institut EG	3.1	50.0	72.8
Institut EG	12.4	50.0	75.1
Institut EG	22.8	50.0	105.5
Institut EG	53.2	50.0	166.9
Institut EG	75.3	50.0	216.3
Institut EG	92.6	50.0	249.4
Institut EG	117.3	50.0	311.2

Table 5.3: Runtime comparison with variable FOV area and fixed path lengths

hallways or rooms of up to $50m^2$ FOV the path calculation times are under $150ms$ leading to around 7 path recalculations per second. For large open areas the path recalculations drop significantly.



(a) Runtime analysis for constant FOV area and variable path length

(b) Runtime analysis for constant path length and variable FOV area

Figure 5.2: Effects of path length and FOV area on runtime of FOV assisted path planning algorithm

At the end of the performance analysis the individual stages presented in Section 4.3 are evaluated. Table 5.4 visualizes the individual stages and the respective runtimes. The test is conducted on the 3D model in Figure 5.1b. The cell size and cell height are constant at $50cm$. For Run1 and Run2 the FOV area is constant at $10m^2$ and a variable path length of $50m$ and $160m$ is selected. For Run3 and Run4 the path length is constant at $50m$ and the FOV area size is variable at $20m^2$ and $120m^2$.
The results in Table 5.4 show that the generation of the FOV navigation mesh and the path smoothing for long paths are the bottlenecks of the algorithm. In Run2 the path length of $160m$ results in a long path outside the FOV area. A solution for the long runtime is the restriction of the path smoothing outside the FOV area. This is a plausible solution as the user does not see the path outside the FOV and, therefore, no benefit is gained from a smooth path. Due to the generation of the path outside the FOV, the path is still calculated until the end.

For larger FOV areas the generation of the FOV navigation mesh is the bottleneck of the algorithm. The generation time is speeded up through filtering the floor pixels before starting the FOV navigation mesh generation. The generation of the separate FOV navigation mesh is still beneficial in relation to recalculating the navigation mesh for the entire 3D model for every path recalculation. Generation of the FOV navigation mesh in Run4 takes 262.8ms whereas the generation of the global navigation mesh takes 7.8s.

algorithm stage	Run1 (ms)	Run2 (ms)	Run3 (ms)	Run4 (ms)
Calculate path corridor	0.06	0.20	0.05	0.05
Calculate middle path	0.03	0.05	0.02	0.02
Render FOV	22.1	22.2	22.2	23.8
Filter floor pixels	0.12	0.10	0.27	0.09
Generate FOV navmesh	25.0	26.6	44.8	262.8
Calculate FOV path	0.03	0.04	0.07	0.05
Smooth FOV path	0.4	0.5	0.24	2.1
Calculate outside path	0.12	0.4	0.10	0.08
Smooth outside path	17.9	159.9	19.4	14.6
Calculate FOV assisted path	70.5	216.1	92.0	308.3

Table 5.4: Runtime analysis of the individual stages of the algorithm

Using the Recast library by Mononen as basis for the FOV assisted path planning algorithm proved to be a good decision. The separation of the path planning algorithm from the navigation mesh is easy to implement and the navigation mesh eases the calculation of the FOV assisted path. One drawback of calculating a Recast navigation mesh is the amount of variables it takes to generate a navigation mesh. When generating a navigation mesh for a new 3D model, it takes some time to calibrate the navigation mesh generation process with suitable parameters.

5.2 User Study

In order to evaluate the usability of the implemented system a user study was conducted. This section provides an overview of the design of the user study before discussing the results of the user study.

5.2.1 User Study Design

The goal of the user study was to guide participants through a real life scenario with the assistance of the implemented FOV assisted path planning algorithm. Therefore, an AR indoor navigation challenge was implemented where the participants had to follow a calculated path to different targets. Figure 5.3 shows the 3D model of the indoor environment applied in the user study. The test environment was the ground floor of a university building of the Technical University of Vienna. The area was a set of rooms and hallways, thus open rooms as well as narrow hallways have been part of the user study.

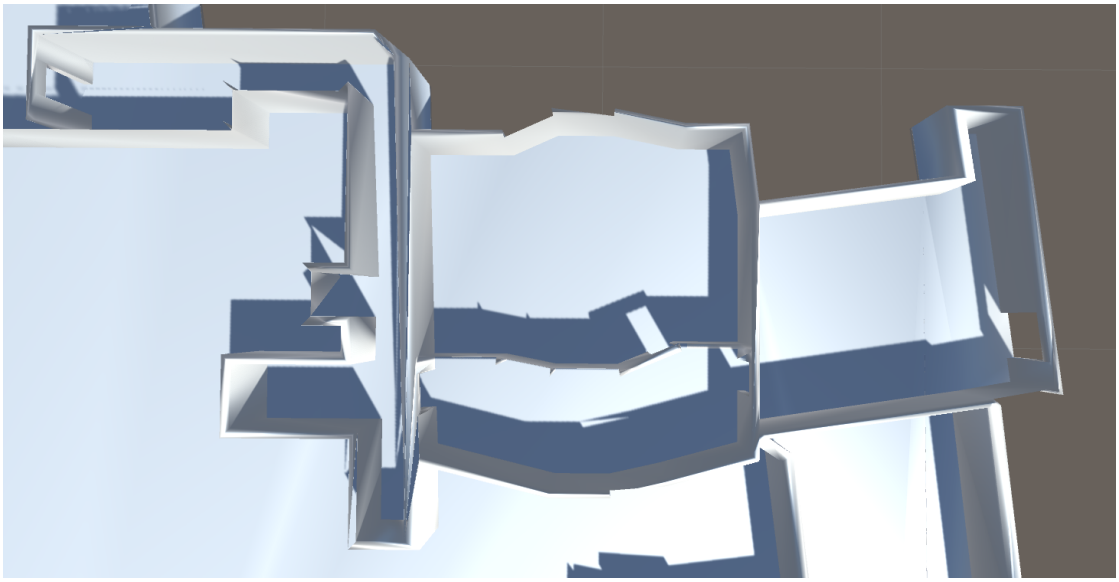


Figure 5.3: 3D model of the indoor environment used in the user study

The main task of the two-part user study was to locate 5 targets distributed around the test environment. The FOV assisted path was calculated to guide the participants towards the targets. The path is calculated from the participant's current position to the next target. Once a participant reached a target, they had to write down a code hidden on the target before moving on to the next target.
While the user is walking around in a real environment, the path information as well as the targets are virtual objects superimposed onto the real environment. Figure 5.4 shows the path visualization and a virtual target with the requested code on it. The path in Figure 5.4a is a particle system composed of floating spheres. The floating spheres float towards the target. Hence the flow direction of the spheres describes the path direction. The path was only visualized for the participants while a remote control button was pressed. The targets are two different virtual objects. A green ring on the floor is displayed for the user continuously, whereas the cylinder with the ghost texture (see Figure 5.4b) is activated when the participant is approaching the target's position.

The user study setup was implemented on a notebook and a Microsoft Hololens see-through HMD. The navigation mesh generation and the path calculation were executed on the notebook. The notebook is equiped with an Intel $i7-4940MX 3.30GHz$ Quadcore CPU with 16.0GB RAM and a Geforce GTX $880M$ GPU with 8GB RAM.
The tracking of the participant's current position and orientation and the visualization of the virtual objects were facilitated with the Microsoft Hololens. A small remote control belonging to the Microsoft Hololens was used to activate the path visualization as described above. The Microsoft Hololens sends the participant's current position and orientation to the notebook. The notebook uses the Hololens data and the precomputed

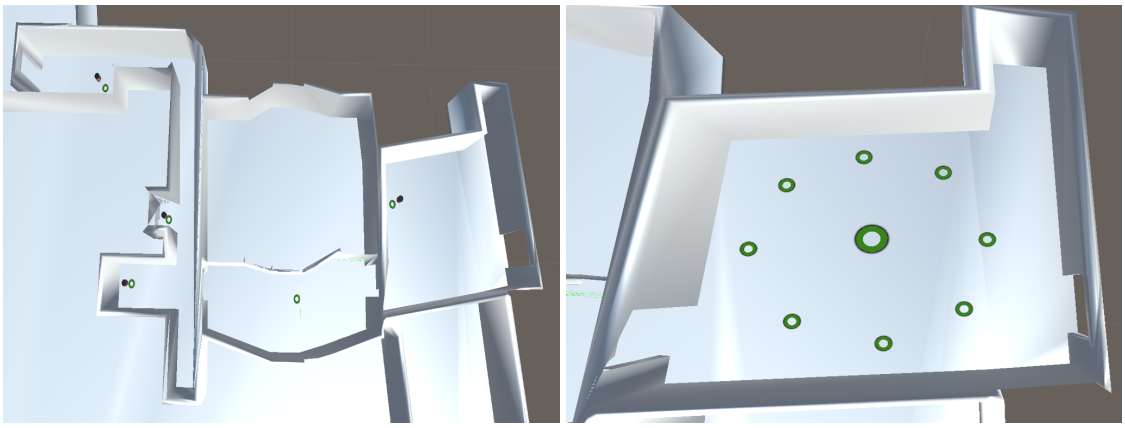
(a) Virtual path superimposed onto real environment

(b) Virtual target with code

Figure 5.4: Virtual objects visualized to the participant on the Microsoft Hololens

navigation mesh to calculate the path towards a target. The path points are then sent back to the Microsoft Hololens and visualized to the participant.
The user study was divided into two separate scenarios and a pre-study to test the scenarios. The pre-study was conducted to evaluate the tracking accuracy of the Hololens and to test the process of the user study. During the first scenario called the navigation challenge, the user had to locate 5 targets, write down a code and then move on to the next target. The visualization was activated with clicking the remote control button. The participants had to accomplish the navigation challenge twice. The difference between the two runs was the visualized path. In one run the FOV assisted path was visualized to the participant, whereas in the other run the shortest path implementation provided from the implemented FOV assisted path planning algorithm was displayed on the Microsoft Hololens. The selected path for the first run was alternated between every participant. Every run had 5 unique positions for the targets. Before the run started the participant was able to get used to the path and target visualization in a small testing area. Once the participant was familiar with the system, the evaluated run started.
The goal of the second scenario referred to as orientation challenge, was to find different targets again. Instead of moving around in the indoor environment, the participant was standing on a predefined location and had to find 8 predefined targets distributed around the participant's location. The participants had to turn around their own axis to find the targets. Once a target was found, the participant had to activate the next target by pressing the button on the Hololens remote control. The orientation challenge was once again split up into two runs. The difference between the two runs was again the visualization of the FOV assisted path in one run and the shortest path in the other run. The activation order of the 8 targets was changed between the two runs.
Figure 5.5 shows the indoor environment and targets of both scenarios. In Figure 5.5a the environment of the navigation challenge is visualized. Five targets (black cylinders

and green rings) are distributed in the indoor environment and the participant had to find one after the other. For the second run five different targets are positioned around the indoor environment.
Figure 5.5b visualizes the environment of the orientation challenge. The predefined location of the participant is the larger ring in the middle and the 8 targets are located around the participant. The targets are activated one after the other with the click of the remote control button. The participants had to remain at the predefined location and find the targets by rotating around their own axis.



(a) 5 Targets of the navigation challenge distributed within the indoor environment

(b) 8 Targets (small green rings) and predefined participant position (large green ring)

Figure 5.5: Environments of both scenarios including targets

In order to evaluate the user study, a questionnaire was handed to the user study participants. The questionnaire starts with a general section on previous experience with AR technology and experience with regard to navigation (see Table 5.5).

Q1	I have experienced Augmented Reality before...
Q2	If yes, in what way (device, applications)?
Q3	I feel uneasy interacting with the Microsoft Hololens...
Q4	I have a good sense of orientation...
Q5	I feel easy navigating through unknown indoor environments...

Table 5.5: General questions of the user study

The questions listed in Table 5.5 are rated on a 5-part Likert scale ranging from ‘Not at all‘ to ‘Very much‘ except for Q2. Q2 is an open text field where participants write down their previous experiences with AR devices and applications.
After each run of the navigation challenge the experience was evaluated with three different questionnaire parts. The first part were individual questions about the experience

during the corresponding run (see Table 5.6).

Q6	How good did the virtual path lead you towards the targets?
Q7	Was the visualized path intuitive to follow?
Q8	To what extent did the visualized path irritate or confuse you, so you did not know where to go?
Q9	At which point in time did you get used to the guiding path?
Q10	Did you feel save when moving around during the task?
Q11	Have you perceived the environment while being navigated?

Table 5.6: Individual questions for every run in the navigation challenge

The questions listed in Table 5.6 are rated on a 7-part Likert scale ranging from ‘Not at all (Q6,Q7,Q10,Q11) / At no time (Q8) / Not until the end (Q9)‘ to ‘Very much (Q6,Q7,Q10,Q11) / All the time (Q8) / Immediately (Q9)‘.
The next part of the questionnaire consists of standardized tests to evaluate usability and workload of each run are in the questionnaire. The System Usability Scale (SUS) is included to evaluate the usability of the test setup [6]. The SUS persists of ten questions rated on a 5-part Likert scale ranging from ‘Strongly disagree‘ to ‘Strongly agree‘. The final SUS score is a value between $0 - 100$, where a higher value indicates a higher usability of the system.
The NASA Task Load Index (TLX) is a tool to evaluate the subjective workload of the task at hand [21]. The NASA TLX is a two-phase evaluation tool. The first phase consists of rating 6 individual categories on a 21-part scale. The second part is rating the six categories against each other to evaluate which category the participant indicates as more important for the task. For the analysis in this paper only the first part of the NASA TLX was conducted. After both runs of the navigation challenge the participants had to compare both runs and were able to give positive, negative and general feedback (see Table 5.7).

Q12	Do you find superimposing a virtual path onto the real environment helpful?
Q13	After finishing both scenarios, please rate the scenario again. Would you like to use the path visualization for a way finding task? First run
Q14	After finishing both scenarios, please rate the scenario again. Would you like to use the path visualization for a way finding task? Second run
Q15	Please note the incidents you found most positive during this experience?
Q16	Please note the incidents you found most negative during this experience?
Q17	Please feel free to add any comments, suggestions, criticism....

Table 5.7: Final part of the questionnaire with comparison of the two runs and feedback

The questions Q12 - Q14 in Table 5.7 are rated on a 7-part Likert scale ranging from ‘Not at all‘ to ‘Very much‘. In addition Q13 and Q14 have text fields to state a reason for the answers. Q15 - Q17 are open text fields for comments.

The orientation was evaluated with a different set of scenario specific questions (see Table 5.8).

Q18	How much did it help you that in one scenario the path adapted to your view direction?
Q19	How good could you anticipate the direction to the target? First run
Q20	How good could you anticipate the direction to the target? Second run

Table 5.8: questionnaire for the orientation challenge

The questions Q18 - Q20 in Table 5.8 are rated on a 7-part Likert scale ranging from ‘Not at all‘ to ‘Very much‘. In addition Q19 and Q20 have text fields to state a reason for the answers.

The following research questions were determined before the user study was conducted:

- Is the FOV assisted path suitable for the user study environment?
- What are the advantages and disadvantages of the FOV assisted path with regard to a shortest path implementation?
- Is AR a suitable area of application for path planning?

5.2.2 User Study Results and Analysis

The navigation and orientation challenge described in the last section have been conducted in separate user studies, with some participants taking part in both user studies. 16 participants entered the user study for the navigation challenge. The visualized path for run FOV path and run shortest path was alternated between every participant, so that half of the participants started with the FOV assisted path first and the other half with the shortest path.
The participants were 4 females and 12 males between the ages of 24 and 50. For 8 of the 16 participants it was the first AR experience at all. Two participants had used the Microsoft Hololens before, whereas other participants used mobile phone or tablet AR applications. Despite most of the participants not being familiar with the Microsoft Hololens or other AR applications, the participants had no reservations about using the Microsoft Hololens. Q3 had an average score of 1.375 with a standard deviation of $\sigma = 0.99$. The participants of the user study had an above average sense of orientation

(Q4: average score of 3.56 with $\sigma = 1.06$) and above average sense of navigating through an indoor environment (Q5: average score of 3.31 with $\sigma = 1.10$).
Every participant was able to find all 5 targets in both runs of the navigation challenge. No participant hit any walls or obstacles and no test run failed due to hardware failure or participant related reasons like dizziness or nausea. For the FOV assisted path run the participants walked on average $90.6m$ and it took them $152sec$. For the shortest path run the participants walked on average $92.9m$ and it took them $148.8sec$. The participants had no time limits to finish the runs so that no evaluation of times and speed were conducted during the user study.
The FOV assisted path was the main contribution of this thesis. Figure 5.6 visualizes the participants' ratings for Q6 - Q9 for the FOV assisted path planning algorithm.
Overall the results indicate that the FOV assisted path is helpful to find targets in an indoor environment. 13 of the 16 participants stated that the FOV assisted path leads towards the targets very much. The adaption of the path with regard to the participant's view direction keeps the participant heading towards the target at all time. This is reiterated as the participants stated that they hardly ever did not know where to go. The path is recalculated when the view direction of the participant changes. The constant path changes are not a problem for a majority of the participants as they did get used to the path immediately and indicate that it is intuitive to follow. Some participants stated that they had to get used to the changing of the path or even thought that two different paths were visualized to them. Nevertheless the participants also stated that once they figured out how the path adaption worked, they got used to it. A fading animation between two different path visualizations might solve problems, the participants had with the allegedly double visualized paths.

The second research goal of the user study was to compare the FOV assisted path with the shortest path implementation. The shortest path does not adjust to the participant's view direction, thus the user has to adjust to the path. The FOV assisted path adjusts the path to the view direction of the participant. The comparison between the shortest path and the FOV assisted path indicate that the participants consider both implementations as useful. Figure 5.7 shows a comparison of both evaluated paths. Only one participant gave a significantly lower rating for the FOV assisted path without explaining his rating. In talks after the participants finished the navigation challenge, a large amount of participants reported that the difference between the two runs is minor and was hardly recognized. This explains the equal results in Figure 5.7. The results of Q13 and Q14 indicate that there was no difference for most of the participants between them adjusting to the path or the path adjusting to their view direction for the navigation challenge. Despite equal ratings for Q13 and Q14 some participants stated in the open text fields of the two questions that they did not like searching for the path after they found a target. Searching for the path after the target was found is necessary, when the path leads towards the opposite direction of the participant's view direction. Exactly that case is prevented with the FOV assisted path planning algorithm. One participant stated that the lack of clearance to the wall for the shortest path is a problem. This problem is

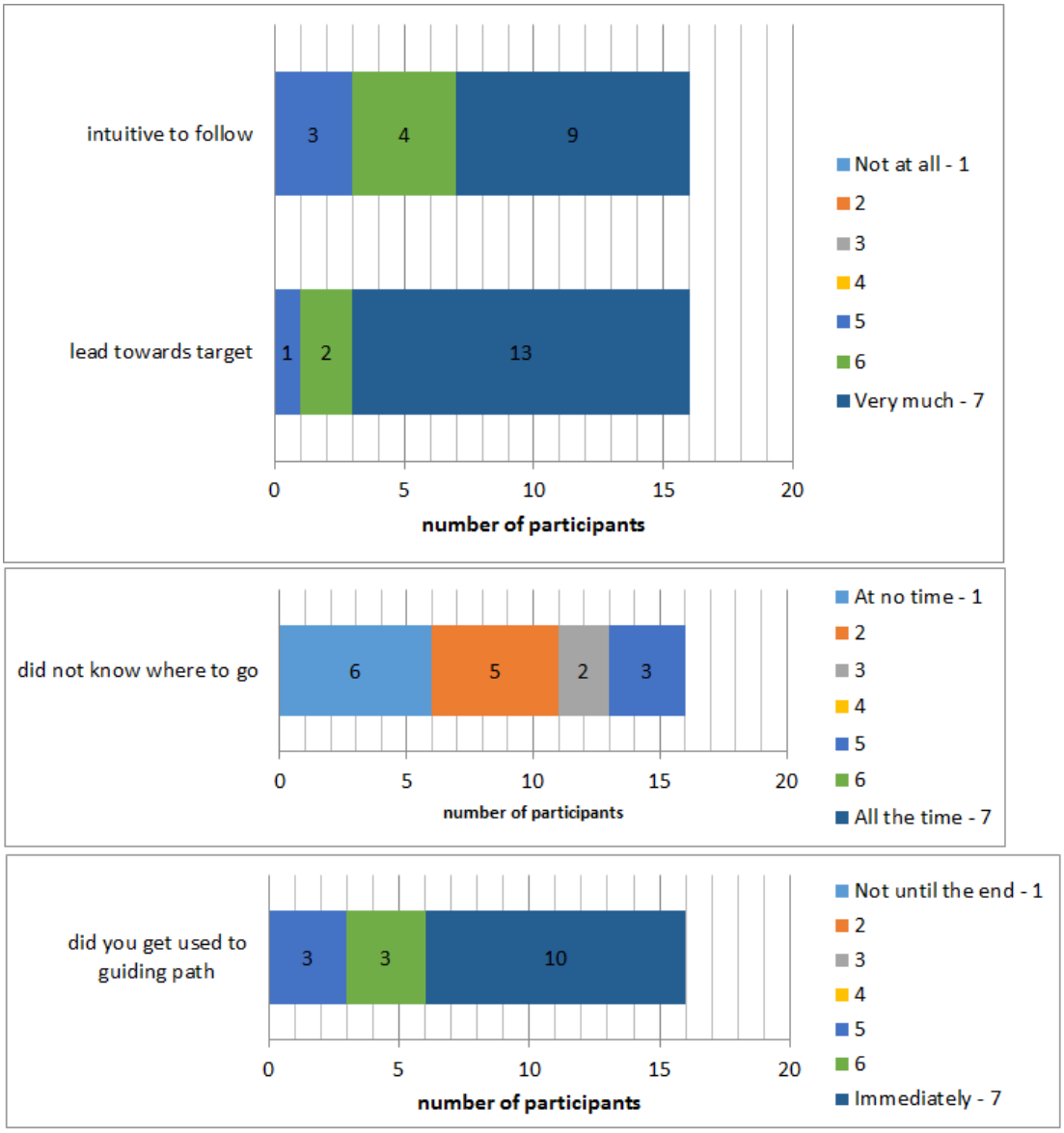
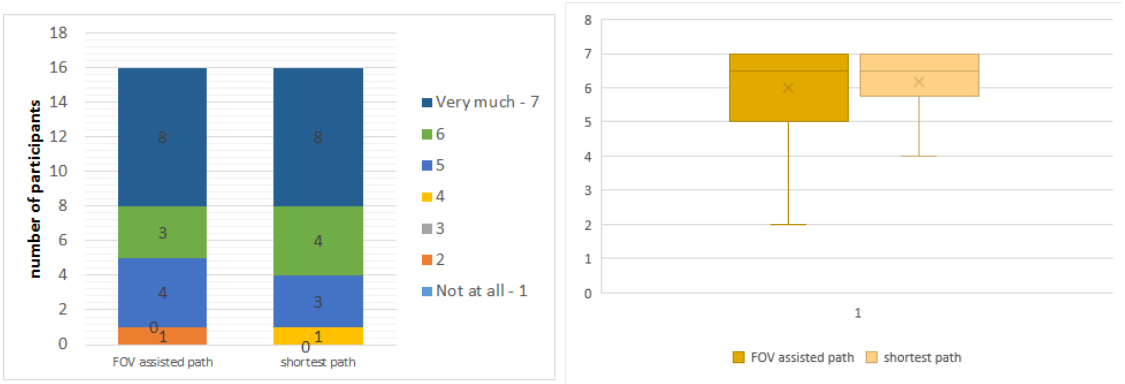


Figure 5.6: Ratings for Q6 - Q9 for the FOV assisted path

also solved with the FOV assisted path as it adds clearance to walls to the path.

The third research question of the user study focuses on the AR aspect of the FOV assisted algorithm. A Microsoft Hololens see-through HMD was used in the user study to visualize the paths. Figure 5.8 visualizes the results of the user study with regard to AR. The Microsoft Hololens superimposes the virtual paths onto the real environment. Therefore, the participants perceive the real environment around them. The participants



(a) Ratings for Q13 and Q14. Comparison between the shortest and FOV assisted path.

(b) Boxplot for Q13 and Q14. Comparison between the shortest and FOV assisted path.

Figure 5.7: Comparison between the shortest and FOV assisted path indicating if the participants would like to use the visualized path.

indicated an above average rating for the perception of the environment. Moreover, that resulted in a likewise high score for feeling save moving around inside the test environment. One participant indicated that wearing the Microsoft Hololens let him perceive the environment as very dark, thus he did not perceive the environment very well.
All three scores in Figure 5.8 indicate that using AR technologies and superimposing virtual path information onto real environments is helpful and an important area of application for AR. In the general feedback of the user study many of the participants criticized the small FOV of the Microsoft Hololens and some of them indicated that the Hololens is too heavy.

The usability of the implemented path planning algorithm was evaluated with the standardized usability test SUS. Figure 5.9 shows a comparison of all the SUS categories between the two path implementations. The results indicate that the implemented system for both paths is easy and quick to learn and that the participants felt confident in using the system.
It is interesting that the participants gave a higher inconsistency level to the shortest path implementation, although the path changes less often in comparison to the FOV assisted path. This might be due to the fact that the participants do not know where to go immediately after they find a target, when the path leads towards the opposite direction.
The official webpage of the SUS test states that an algorithm with a SUS score above 68 is considered an algorithm with above average usability [1]. With an average SUS score of 87.66 (standard deviation $\sigma = 10.06$) for the FOV assisted path and 84.84 (standard deviation $\sigma = 12.58$) for the shortest path both runs score above average usability according to the SUS test.

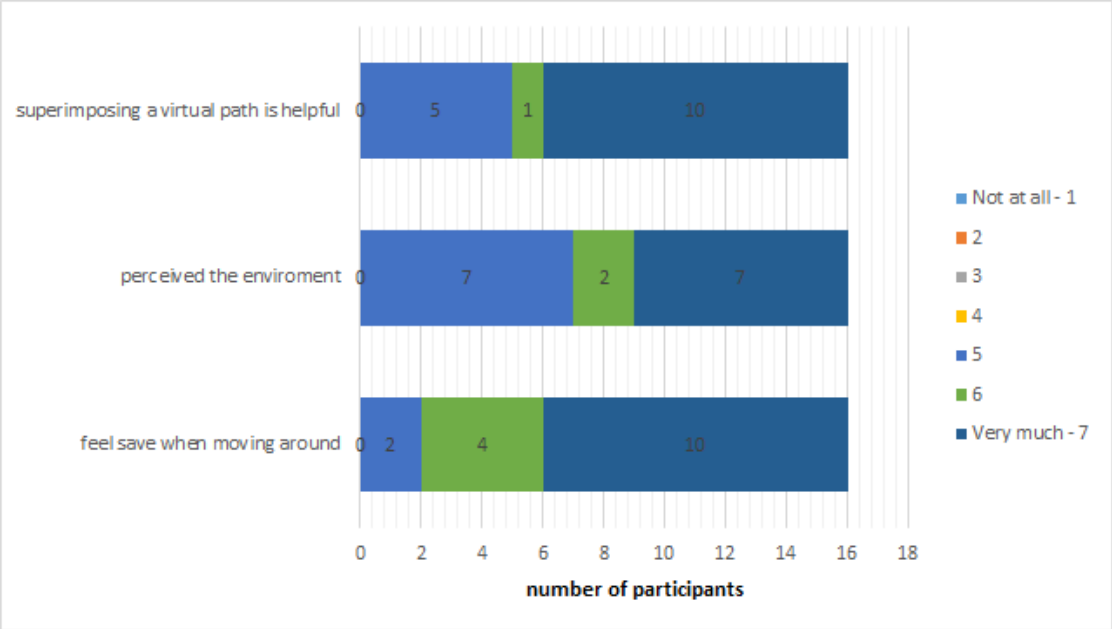


Figure 5.8: Ratings for Q10 - Q12 for the FOV assisted path

The subjective workload of the navigation challenge is assessed with the NASA TLX test [21]. Figure 5.10 shows a visualization of the 6 main categories of the NASA TLX test for the FOV assisted path and the shortest path. The difference between the ratings for the two path implementations are once again minor. The ratings show that the participants felt that they accomplished the given task to a high degree. The low mental demand and effort score indicates that the algorithm was easy to learn for the participants and the visualization helped them to accomplish the target search. The biggest difference between the two paths persists in the physical demand category. The difference might once again be due to the participants not knowing where to go after they found a target, when the path direction is in the opposite direction as the participant's view direction. The slightly higher mental demand for the FOV assisted path might be due to the fact that the path changes more often for the FOV assisted path.

The orientation challenge was evaluated in a separate user study. 12 participants (3 female and 9 male) between the ages of 24 and 50 participated in the second user study. Instead of moving around inside the test environment to find the targets, the participants stood on a predefined position and had to find the targets located around the participants.
Figure 5.11 visualizes the comparison of the shortest path and FOV assisted path implementation in the orientation challenge. The results indicate that the FOV assisted path helps the participants in locating the targets very much, whereas the shortest path has a significant lower score. The reasoning of the participants for the lower score is

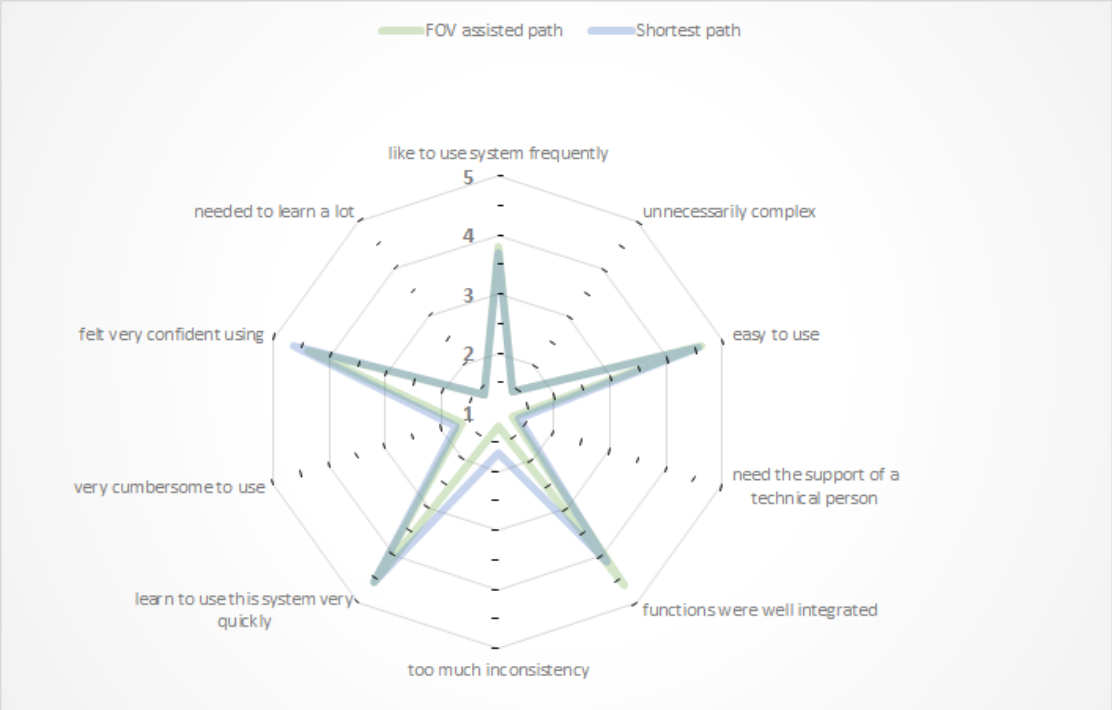


Figure 5.9: Comparison of the raw SUS data for FOV assisted path and shortest path

that the FOV assisted path guides them towards the target immediately, whereas the participant has to find the direction, and thus the target, on their own with the shortest path. This is in accordance with the definition of the FOV assisted path, as the path calculation adopts to the view direction of the participant; as such, a valid path is visible to the user at all time. The participant who rated the FOV assisted path as not helpful at all, stated that he ignored the path to solve the task. Therefore, for the participant both runs result in the same experience.

The results of Figure 5.12 confirm the positive effects of the FOV assisted path for the orientation challenge. Contrary to the low score of two participants for Q18, they still stated that the adaption of the view direction was helpful.

Comparing the results between the navigation and orientation challenge it stands out that the participants do not indicate much of a difference between the shortest and FOV assisted path for the navigation challenge. In contrast, for the orientation challenge, there is a significant difference between the scores of the two paths. The orientation challenge focuses on the uncertainty of the participants when they do not know which direction to go after the target position changed. During the navigation challenge the time of orientating towards a new target in respect to following the path towards the target is small. In the orientation challenge the target's location and thus the participant not knowing the target location changes quickly.

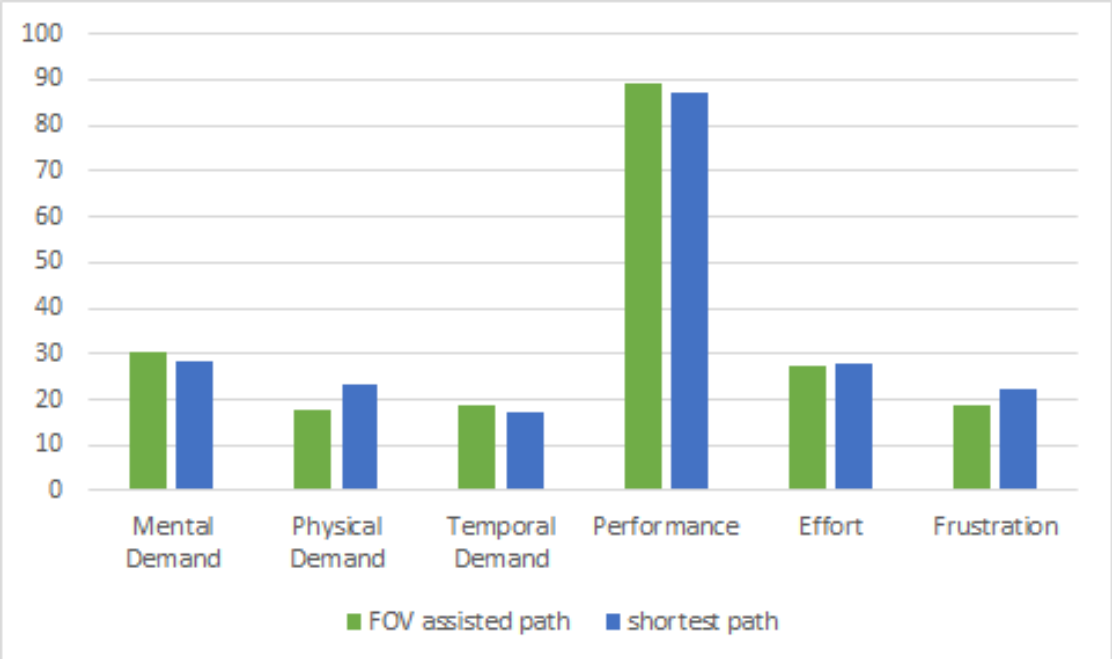
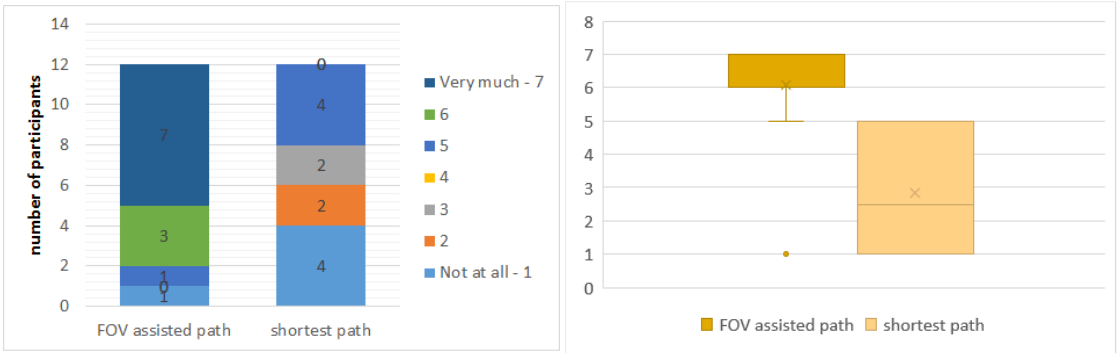


Figure 5.10: Comparison of the NASA TLX data for FOV assisted path and shortest path

Therefore, the FOV assisted path is most helpful as long as the direction towards a target is not clear to the participant. When the direction towards the target is allegedly clear to the participants, both path implementations are perceived as helpful.

Another interesting perception of the user study led to an expansion of the algorithm. During a run in the navigation challenge the participant is looking towards both walkable and unwalkable areas. Walkable surfaces of the indoor environment consist of areas where the participant is able to walk, such as the floor of the building. Unwalkable areas, in contrast, consist of walls or obstacles. When the participants activate the path visualization, they focus on the ground and, therefore, look towards walkable areas of the environment. As long as the algorithm detects a walkable area, the FOV assisted path is calculated. When a participant is not looking towards a walkable surface, a fallback path without the FOV information was visualized to the participant during the user study. The fallback path visualization is invoked when participants loook towards a wall or an obstacle. This is often invoked when the participant knows where to go and neglects looking towards the floor or in narrow hallways. The expansion of the FOV assisted path implemented after the user study was conducted, calculates a FOV assisted path even when no walkable area is inside the FOV of the user. The so-called wall path is presented in Section 4.3.7.



(a) Ratings for the orientation challenge. Comparison between the shortest and FOV assisted path.

(b) Boxplot for the orientation challenge. Comparison between the shortest and FOV assisted path.

Figure 5.11: Comparison between the shortest and FOV assisted path indicating how good the participants could anticipate the direction to the target.

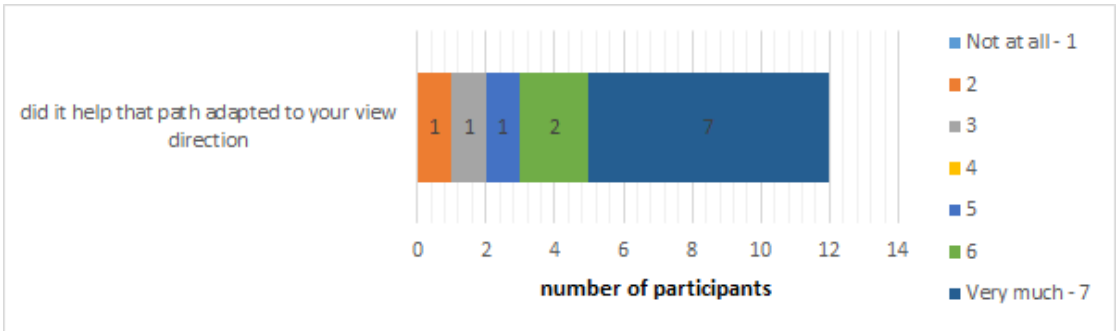


Figure 5.12: Ratings for Q18

CHAPTER 6

Summary and future work

In this thesis, a path planning algorithm for an AR indoor environment was designed and implemented. The algorithm is called FOV assisted path planning algorithm. The novel approach of this thesis is to include FOV information of a mobile device into the path planning process. The FOV area is the area of the indoor environment captured with the mobile device's camera. The final path should be inside the FOV of the mobile device at all times. The algorithm is implemented for the usage with AR visualization devices such as smartphones, tablets or a see-through HMD. Allowing for collision-free movement through the indoor environment the FOV assisted path is calculated with clearance to walls and obstacles. In addition to avoiding walls and static obstacles within the indoor environment, the FOV assisted path planning algorithm avoids collisions with dynamic obstacles.
The implemented algorithm uses the 3D model of the indoor environment to calculate the walkable areas of the indoor environment. Walkable areas represent all areas of the indoor environment the user is able to walk on. The walkable areas are combined in a data structure called navigation meshes. The Recast library implemented by Mononen [32] is applied to generate the navigation mesh. The navigation mesh combined with the tracking data of the user and the destination information are the input parameters for the path planning algorithm. The tracking data represent the current position of the user, hence the mobile device and the view direction of the mobile device. At first, the shortest corridor from the current position to the destination is calculated. Then a path with clearance to static obstacles and walls is calculated. This path with clearance is then extended to include the FOV information of the mobile device. Therefore a separate navigation mesh of the FOV area including dynamic obstacles is generated. Based on this FOV navigation mesh a path through the FOV of the mobile device is calculated and concatenated with the path towards the destination. The work at hand presents several special cases where the paths are calculated and concatenated in a particular way. Special cases depend on the view direction of the mobile device or on a lack of walkable

areas. The final FOV assisted path is smoothed to provide a more intuitive looking path to the user.
To evaluate the performance and usability of the FOV assisted path planning algorithm a user study was conducted. A navigation challenge where participants had to find different targets within an indoor environment was conducted. The see-through HMD Microsoft Hololens was used to track the participants inside the building and the AR path was visualized on the Microsoft Hololens. During the user study the FOV assisted path itself, the FOV assisted path planning algorithm in comparison to a shortest path implementation and the overall AR experience was evaluated. A majority of the participants liked using the FOV assisted path for the navigation task and the algorithm was awarded with high usability ratings. The FOV assisted path was preferred to the shortest path implementation in situations where the user did not know the direction towards the destination. The main advantage of the FOV assisted path is the availability of a path inside the user's FOV at all times. Therefore, the user has permanent feedback on the direction towards the target. The participants also gave positive reviews about the AR setting of the navigation challenge and confirmed path planning as an area of application for AR visualization devices.

In future work all the stages of the FOV assisted path planning algorithm need further testing in real life scenarios. During the user study certain positions and orientations led to subpar FOV assisted paths.
The implementation of the wall path functionality presented in Section 4.3.7 displays the problem of no available walkable surfaces. The wall path implementation is a very basic method to combine unwalkable surfaces with walkable areas within the indoor environment. In future work a data structure similar to navigation meshes for walkable areas could be implemented for unwalkable areas to plan paths along walls or other obstacles.
The Recast library is limited to circular dynamic obstacles. Due to the fact that the Recast library is open source, the library could be extended to support other shapes of dynamic obstacles.
The user study was conducted without dynamic obstacles and multistory buildings. The goal of the user study was to evaluate the core functionality of the FOV assisted path planning algorithm. Dynamic obstacles and multistory buildings add an extra level of difficulty for the participants of the user study. Future evaluation should be expanded to incorporate dynamic obstacles and multistory buildings.
Experiencing different visualization techniques for the FOV assisted path was not part of this study's scope. Evaluating different visualization techniques to find a visualization technique suitable for an AR visualization device is an interesting topic for future work. Frequent fast changes of the view direction of the mobile device might lead to paths that jump around on the AR visualization device. Fading between jumping paths could result in a clearer visualization for the user. Other techniques to solve the jumping paths are also tasks for future work.

List of Figures

List of Tables

List of Algorithms

Acronyms

AR Augmented Reality. v–viii, 1–3, 5, 6, 8, 10, 12, 15–22, 25, 26, 31, 48, 63, 65, 68, 70, 72, 73, 79–81

DLL Dynamic Link Library. 31–33, 44, 45, 47, 48, 52, 56

DOF degree of freedom. 16–18

ECM Explicit Corridor Maps. 10–13, 81

FOV Field of View. v–viii, 1–3, 6–8, 10–13, 15–18, 20–29, 31–33, 37, 40, 41, 44–54, 56–67, 70–77, 79–83

GPS Global Positioning System. 1

HMD Head Mounted Display. v, 1, 16, 17, 19, 20, 22, 66, 72, 79, 80

HPU holographic processing unit. 19

INSAR Indoor Navigation System Using Augmented Reality. 6, 7

LCT Local Clearance Triangulation. 10–13, 81

NEOGEN Near optimal generator of navigation meshes. 11

SUS System Usability Scale. 69, 73, 75, 82

TLX Task Load Index. 69

VR Virtual Reality. 19

WIM World-In-Miniature. 9, 10

Bibliography

[1] *System Usability Score*, accessed December 2, 2017. `https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html`.

[2] Ahmed Alnabhan and Brian Tomaszewski. INSAR: Indoor Navigation System using Augmented Reality. In *Proceedings of the Sixth ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness*, pages 36–43. ACM, 2014.

[3] Ronald T Azuma. A Survey of Augmented Reality. *Presence: Teleoper. Virtual Environ.*, 6(4):355–385, 1997.

[4] Serge Beucher and Fernand Meyer. The morphological approach to segmentation: the watershed transformation. *Mathematical Morphology in Image Processing*, pages 433–481, 1993.

[5] Gunilla Borgefors. Another comment on "a note on 'distance transformations in digital images'". *CVGIP: Image Understanding*, 54(2):301–306, 1991.

[6] John Brooke. SUS - A quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

[7] Bernard Chazelle. A theorem on polygon cutting with applications. In *Foundations of Computer Science, 1982. SFCS '08. 23rd Annual Symposium on*, SFCS '82, pages 339–349, Washington, DC, USA, 1982. IEEE Computer Society.

[8] Xiao Cui and Hao Shi. A*-based Pathfinding in Modern Computer Games. *International Journal of Computer Science and Network Security*, 11(1):125–130, 2011.

[9] Buti Al Delail, Luis Weruaga, and M. Jamal Zemerly. CAViAR: Context aware visual indoor augmented reality for a University Campus. In *Proceedings of the 2012 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology Workshops, WI-IAT 2012*, volume 3, pages 286–290, 2012.

[10] Mark DeLoura. *Game Programming Gems*. Charles River Media, Inc., Rockland, MA, USA, 2000.

[11] H Digabel and Christian Lantuejoul. Iterative Algorithms. In *Proceedings of the 2nd European Symposium Quantitative Analysis of Microstructures in Material Science, Biology and Medicine*, volume 19, pages 85–89. Stuttgart, West Germany: Riederer Verlag, 1978.

[12] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[13] David H. Douglas and Thomas K. Peucker. Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature. In *Classics in Cartography: Reflections on Influential Articles from Cartographica*, pages 15–28. 2011.

[14] Roland Geraerts. Planning short paths with clearance using explicit corridors. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1997–2004, 2010.

[15] Roland Geraerts and Mark H. Overmars. The corridor map method: Real-time high-quality path planning. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1023–1028, 2007.

[16] Georg Gerstweiler, Emanuel Vonach, and Hannes Kaufmann. HyMoTrack: A mobile AR navigation system for complex indoor environments. *Sensors (Switzerland)*, 16(1), 2015.

[17] Google Inc. *Google ARCore*, accessed December 2, 2017. `https://developers.google.com/ar/`.

[18] Google Inc. *Google Tango*, accessed December 2, 2017. `https://get.google.com/tango/`.

[19] Google Inc. *Google Tango Developers*, accessed December 2, 2017. `https://developers.google.com/tango/overview/concepts`.

[20] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[21] Sandra G. Hart and Lowell E. Staveland. Development of NASA-TLX - Results of empirical and theoretical research. *Human mental workload*, 52:239–250, 1988.

[22] Denis Haumont, Olivier Debeir, and François Sillion. Volumetric cell-and-portal generation. In *Computer Graphics Forum*, volume 22, pages 303–312, 2003.

[23] Bernhard Hofmann-Wellenhof, Herbert Lichtenegger, and James Collins. *Global positioning system : theory and practice*. Springer-Verlag Wien, 2001.

[24] Tobias H. Höllerer, Drexel Hallaway, Navdeep Tinna, and Steven Feiner. Steps toward accommodating variable position tracking accuracy in a mobile augmented reality system. *2nd International Workshop on Artificial Intelligence in Mobile Systems*, (September 2015):31–37, 2001.

[25] Low Chee Huey, Patrick Sebastian, and Micheal Drieberg. Augmented reality based indoor positioning navigation tool. *2011 IEEE Conference on Open Systems*, pages 256–260, 2011.

[26] Marcelo Kallmann. Shortest Paths with Arbitrary Clearance from Navigation Meshes. *Proceedings of the Eurographics SIGGRAPH Symposium on Computer Animation SCA*, pages 159—-168, 2010.

[27] Marcelo Kallmann. Dynamic and Robust Local Clearance Triangulations. *Acm Transactions on Graphics*, 33(5):17, sep 2014.

[28] Sebastian Kasprzak, Andreas Komninos, and Peter Barrie. Feature-based indoor navigation using Augmented Reality. *Proceedings - 9th International Conference on Intelligent Environments, IE 2013*, pages 100–107, 2013.

[29] Oussama Khatib. Real time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics and Research*, 5(1):90–98, 1986.

[30] Richard Knoblauch, Martin Pietrucha, and Marsha Nitzburg. Field Studies of Pedestrian Walking Speed and Start-Up Time. *Transportation Research Record: Journal of the Transportation Research Board*, 1538:27–38, 1996.

[31] C.G. Low and Y.L. Lee. SunMap+: an intelligent location-based virtual indoor navigation system using augmented reality. In *International Conference on Frontiers of Communications, Networks and Applications (ICFCNA 2014 - Malaysia)*, 2014.

[32] Mikko Mononen. *Recast Library*, accessed December 2, 2017. `https://github.com/recastnavigation/recastnavigation`.

[33] Alessandro Mulloni, Hartmut Seichter, and Dieter Schmalstieg. Handheld augmented reality indoor navigation with activity-based instructions. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services - MobileHCI '11*, MobileHCI '11, page 211, New York, NY, USA, 2011. ACM.

[34] Ramon Oliva and Nuria Pelechano. NEOGEN: Near optimal generator of navigation meshes for 3D multi-layered environments. *Computers and Graphics (Pergamon)*, 37(5):403–412, 2013.

[35] Theo Pavlidis. *Algorithms for graphics and image processing*. Springer Science & Business Media, 1982.

[36] John L. Pfaltz. Sequential Operations in Digital Picture Processing. *Journal of the ACM*, 13(4):471–494, 1966.

[37] Franco P. Preparata. The medial axis of a simple polygon. In *Mathematical Foundations of Computer Science 1977*, volume 53 of *Lecture Notes in Computer Science*, pages 443–450. Springer, 1977.

[38] Urs Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1(3):244–256, 1972.

[39] Umair Rehman and Shi Cao. Augmented-Reality-Based Indoor Navigation: A Comparative Analysis of Handheld Devices Versus Google Glass. *IEEE Transactions on Human-Machine Systems*, 47(1):140–151, feb 2017.

[40] Gerhard Reitmayr and Dieter Schmalstieg. Location based applications for mobile augmented reality. In *Proceedings of 4th Australasian user interface conference on User interfaces*, AUIC '03, pages 65–73, Darlinghurst, Australia, Australia, 2003. Australian Computer Society, Inc.

[41] Greg Snook. Simplified 3D movement and pathfinding using navigation meshes. *Game Programming Gems*, 1(1):288–304, 2000.

[42] Robert H Spector. Visual Fields. *Clinical methods: the history, physical, and laboratory examinations 3rd edition*, pages 565–572, 1990.

[43] William C. Swope, Hans C Andersen, Peter H Berens, and Kent R Wilson. A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters. *Journal of Chemical Physics*, 76(1):637–649, 1982.

[44] Allen G Taylor. *Develop Microsoft HoloLens Apps Now*. Apress, Berkely, CA, USA, 1st edition, 2016.

[45] Unity Technologies. *Unity3D - Game Engine*, accessed December 2, 2017. `https://unity3d.com/de`.

[46] Paul Tozour. The Evolution of Game AI. *Ai Game Programming Wisdom*, 1:3–15, 2002.

[47] Wouter Van Toll, Atlas F. Cook IV, and Roland Geraerts. Navigation meshes for realistic multi-layered environments. In *IEEE International Conference on Intelligent Robots and Systems*, pages 3526–3532. IEEE, 2011.

[48] Wouter van Toll, Atlas F. Cook IV, Marc J. van Kreveld, and Roland Geraerts. The Medial Axis of a Multi-Layered Environment and its Application as a Navigation Mesh. pages 1–33, 2017.

[49] Wouter van Toll, Roy Triesscheijn, Marcelo Kallmann, Ramon Oliva, Nuria Pelechano, Julien Pettré, and Roland Geraerts. A comparative study of navigation meshes. In *Proceedings of the 9th International Conference on Motion in Games - MIG '16*, MIG '16, pages 91–100, New York, NY, USA, 2016. ACM.

[50] Wouter G. Van Toll, Atlas F. Cook IV, and Roland Geraerts. A navigation mesh for dynamic environments. *Computer Animation and Virtual Worlds*, 23(6):535–546, 2012.

[51] Loup Verlet. Computer "experiments" on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules. *Physical Review*, 159(1):98–103, 1967.

[52] Ron Wein, Jur P. Van Den Berg, and Dan Halperin. The visibility-Voronoi complex and its applications. *Computational Geometry: Theory and Applications*, 36(1):66–87, 2007.