computers

MDPI

Article

DARGS: Dynamic AR Guiding System for Indoor Environments

Georg Gerstweiler *, Karl Platzer and Hannes Kaufmann

Institute of Software Technology and Interactive Systems, Vienna University of Technology, Favoritenstrasse 9-11-188/2, 1040 Vienna, Austria; platzer.k@gmail.com (K.P.); kaufmann@ims.tuwien.ac.at (H.K.)

* Correspondence: gerstweiler@ims.tuwien.ac.at; Tel.: +43-(1)58801-18823

Received: 21 November 2017; Accepted: 24 December 2017; Published: 28 December 2017

Abstract: Complex public buildings, such as airports, use various systems to guide people to a certain destination. Such approaches are usually implemented by showing a floor plan that has guiding signs or color coded lines on the floor. With a technology that supports six degrees of freedom (6DoF) tracking in indoor environments, it is possible to guide people individually, thereby considering obstacles, path lengths, or pathways for handicapped people. With an augmented reality (AR) device, such as a smart phone or AR glasses, the path can be presented on top of the real environment. In this paper, we present DARGS, an algorithm, which calculates a path through a complex building in real time. Usual path planning algorithms use either shortest paths or dynamic paths for robot interaction. The human factor in a real environment is not considered. The main advantage of DARGS is the incorporation of the current field of view (FOV) of the used device to visualize a more dynamic presentation. Rather than searching for the AR content with a small FOV, with the presented approach the user always gets a meaningful three-dimensional overlay of the path independent of the viewing direction. A detailed user study is performed to prove the applicability of the system. The results indicate that the presented system is especially helpful in the first few important seconds of the guiding process, when the user is still disoriented.

Keywords: augmented reality; indoor navigation; path planning; path visualization

1. Introduction

Guiding people to specific locations in indoor environments is a challenging task. Especially in complex buildings, such as airports, hospitals, or other public buildings, operators are struggling with the problem of guiding visitors through their building in an optimized way. Visitors often have to go through different paths in order to reach their individual goals. Therefore, it is necessary to tailor a guiding system to the need of a visiting person and the specific task that has to be fulfilled. This can be a customer, but also a person who needs to fulfill maintenance tasks or has to deliver a package. Nowadays, guiding systems such as overhead signposts, maps, or digital stationary terminals try to help a person find their way. Individual solutions can only be provided with digital content, where the current position of the user in the building needs to be estimated. Many technologies have already been published [1] proving a reasonable tracking technique in indoor environments with an accuracy of a few meters down to a few centimeters in certain regions. Technologies using signal strength triangulation methods allow us at least to estimate a rough position. Other technologies, such as vision based methods [2], provide higher accuracy in positional tracking and also estimate the orientation of a device in real time. However, there are still limitations concerning the tracking volume and reliability over time.

While walking through the environment and using a camera and a display in a guiding setup, it is possible to present individual augmented information to a user. Display devices, such as smartphones,

or more sophisticated devices, such as augmented reality (AR) glasses such as the HoloLens, are able to localize the person while presenting visual content.

A number of research projects focus on implementing a novel tracking technology, but neglect the human factor. Pedestrians have much more freedom in motion in contrast to people driving cars or bicycles. Therefore, the presentation of the guiding information has to be more flexible and capable of reacting to sudden changes in motion and especially to fast head movements. A prerequisite is to have a reliable, valid, and human understandable calculated path starting from the current position of the user and leading towards a certain target. For that reason, the paper at hand presents a novel dynamic AR guiding system (DARGS). In this paper, the focus lies on the structure of the path within a usually small AR field of view (FOV) area in which virtual content can always be observed by the user. In order to prevent the user from searching for relevant AR content (yellow path in Figure 1) with such a tiny window into the virtual world, DARGS creates a new appearance of a path (green path in Figure 1). The constructed FOVPath is dependent on a number of hardware, human, and environmental constraints, while leading a person to any three-dimensional (3D) position within the indoor environment.

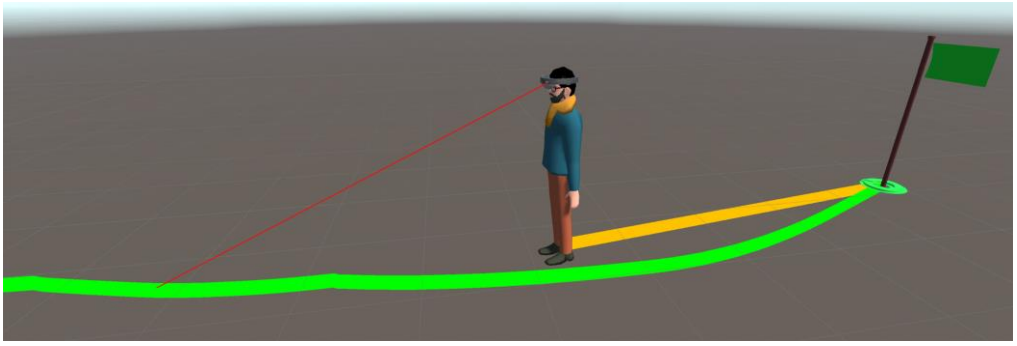


Figure 1. The green path is reacting to the viewing direction of the user in contrast to conventional paths (yellow), which usually try to calculate a direct path.

The work at hand is structured into four sections. Right after discussing other approaches in this area, the proposed system is presented. It contains the design of the system and implementation details. A detailed description of the path calculation is given. Furthermore, solutions are presented to enable an almost constant visible path independent of the viewing direction of the user. Before presenting technical results, a real-world AR guiding application will be described. Finally, a user study proves the performance of the implemented algorithm with 16 participants, who were asked to perform two different AR indoor guiding tasks.

2. Motivation and Contribution

While the authors of this work were dealing with various tracking solutions, including an implementation of a vision-based tracking system, tested in an airport scenario, the lack of available and customizable path generation algorithms became noticeable. During various tests, the visualization of the path for AR was not satisfying. Also, other approaches in this direction only used fundamental graph traversal algorithms, such as A* [3,4] or Dijkstra [5], mostly in order to place static arrows along the way. These basic approaches are not suitable and not flexible enough to create a credible AR guiding path. Using clearances to walls is far not enough to provide the user with constant guiding information. Previous research has shown [2] that some people do not need guiding information all the time, but only at certain spots. For that reason, re-initializations are likely to be observed along the path. The user therefore always has to search for the virtual content with the rather small available window into the virtual world. The FOV of state of the art AR glasses are usually around 40° horizontal and even less in the vertical dimension. Because of these reasons, we believe that a novel path planning

algorithm, which is specifically trimmed to AR guiding tasks, has to be developed and would enhance the guiding experience.

In this paper, our main contribution consists of a well-thought-through path planning algorithm trimmed for usage in AR guiding applications. We provide a guiding path to an arbitrary 3D position in an indoor environment in real time on mobile devices depending on the size of the field of view. In addition to that, we point out specific situations that appear when integrating the viewing direction and the current virtual field of view into the path calculation process. In contrast to other work in this area, this allows us to present visual information about the direction all the time, even if the user is looking at an arbitrary viewing angle. Furthermore, we also present a decision-making process based on a path recalculation model for triggering the update of the path. The presented approach is parameterizable, and can be therefore be adjusted to the available hardware performance. On top of the implemented FOVPath, we designed a visualization indicating direction and speed. Nevertheless, it is also possible to test other visualizations with this path, such as using arrows or even walking avatars. Occlusions are also considered at all times. The presented approach is designed to be independent of the tracking system and can be integrated into any application where the position and orientation of the user is known.

The implementation of the path will be included into DARGS, which contains the FOVPath calculation, a visualization of the path, and a guiding application on the Microsoft HoloLens. Finally, we show that our suggested approach can successfully be used in order to guide pedestrians in an indoor environment. A user study was conducted in order to observe how people behave when being guided with AR glasses. We observed parameters such as walking speeds, duration, and the amount of information needed in order to reach the desired target.

3. Related Work

We split related work into two sections. First, projects dealing with the implementation of an indoor tracking are presented. These selected projects do not only present a tracking solution, but also use a custom implementation of a guiding task in order to evaluate the quality of their work. Different approaches were used to implement such a virtual guide in AR. Another area of research concentrates on the design of a user interface to guide people in AR, and deals with step-by-step descriptions or audio instructions. In the second part, relevant work that has been done in the area of path calculation for crowd simulation and animation is presented, although these approaches have never been adapted to an AR guiding task and its special requirements.

3.1. Tracking for Indoor Navigation and Path Visualization

For providing up-to-date guiding information to a user, it is essential to know the user's current position within the building. Different types of technologies can be used to estimate the pose of the user in the room. In the area of AR indoor navigation, two approaches are commonly used. On the one side, there are sparse localization technologies, in which tracking is only available within certain key-points in the real environment. On the other side, there are implementations that allow for continuous tracking, with available position and orientation. This enables the full capabilities of AR visualizations at arbitrary locations. Depending on the used technologies, different guiding possibilities have been presented by different research groups.

With Key-Point Localization, it is possible to define certain points in the room where the user is able to receive tracking information. This gives the user access to AR content on specific predefined locations in the building. Work that has been published in this area, such as [6–9], uses different kinds of markers to provide locational tracking. Early work, such as in [8] or [9], was based on black and white fiducial markers specifically placed in the environment. Later, scientists used libraries such as Vuforia [10], which allows them to use images as markers. This makes it possible to integrate visual features of the environment [6,7] without the need to place fiducial markers on walls or floors. Therefore, features such as company logos, door plates, or paintings can be integrated into the tracking

system. Due to the restricted tracking capabilities, visualizations can only be placed within the immediate vicinity of these markers. Mulloni et al. [10] focused on the design of the user interface and information presented to the user. In addition to arrows, they also present information such as walking instructions, targeting step counting and turn information. Furthermore, the visualization and computation of the guiding path was also not that relevant in these scenarios. In most of the approaches, static arrows are used to communicate the direction towards the target to the user. In order to guide the user, Kasprzak et al. [7] used, for example, small arrows right on top of markers with a known position in the room.

A solution with more accurate tracking and thereby having more guiding flexibility was presented by Alnabhan et al. [1]. The developed tracking system is based on Wi-Fi triangulation and provides a position every 3 s. In combination with this position, a compass is used to detect the orientation of the user in relation to the target point. Based on this development, the authors of that work implemented a user interface which shows a 3D arrow at a fixed point on the screen. This red arrow rotates depending on the relation between the phone's orientation and the target position.

Kim et al. [11] presented a concept with AR and indoor navigation with a self-developed tracking algorithm. Although the tracking is based on markers and so-called image sequence matching, which would allow for a 3D presentation of AR data, users were only guided with a two-dimensional (2D) miniature map in the top left corner of an AR display.

For displaying AR guiding information at arbitrary positions within a building, continuous tracking is beneficial. In this area, most of the already published work contains tracking solutions with less focus on planning the guiding process. Great indoor tracking approaches have been published in [12,13], but unfortunately did not present any application scenarios or guiding approaches at all. The main focus in these publications is on tracking accuracy in buildings, although these setups would be perfect for evaluating various methods in the area of path planning and guiding methods. Other projects with similar tracking possibilities have integrated guiding scenarios [2,14,15]. Miyashita et al. [14], for example, used the Ubisense tracking system to provide high quality position information. Using a tablet device, users were able to explore a museum with guiding information. In this case, a virtual character was used to give spatially related information. In order to bring attention to special points of interest, virtual balloons were placed within the environment, which can be seen as a first step to visually guide the attention (viewing direction) of the user towards a certain target.

Regarding visual continuous tracking in combination with a guiding system, the work of Rehman et al. [15] has to be mentioned. In this work, the Metaio SDK (a framework for SLAM tracking and marker tracking) was used to establish a tracking system based on SLAM (simultaneous localization and mapping) in an indoor environment. This approach was extended with sensor fusion and a pair of Google Glasses were tracked in order to guide the user to a certain target. As a visual guiding object, an arrow integrated into the environment was used for showing the direction. In addition to that, the researchers also integrated audio instructions.

A majority of the already mentioned approaches are more concentrated on the developed tracking system and have less focus on the actual guiding concept. Independent of the tracking capabilities, showing an arrow in front of the user is the most used approach. The arrow usually only shows a direction to a next stopover, consisting of predefined positions. Just a few approaches, such as [11], make use of a simple graph traversal approach, such as the well-known A* or Dijkstra algorithm, which provide at least a continuous path from the position of the user towards the target in order to estimate the direction to the target. None of the mentioned approaches take the narrow FOV of AR devices into consideration. Also, the viewing direction has not been addressed in the guiding scenarios where static virtual objects, such as arrows, were placed in the environment.

3.2. Path Planning

For guidance applications, algorithms have to be considered that on one hand deliver a short or even the shortest path and on the other hand need to find a way to the target destination in case a path exists. Algorithms such as the Dijkstra's [5], the A* [3,4], or modifications of these are the first reference points. These algorithms alone can be used to find a rough path to the target, but are far from optimal since the shortest path between two points usually runs along walls and very close to corners. These are not natural walking paths of pedestrians.

Another solution to compute a path between a start and an end point is a graph-based approach, where the nodes can be user-defined. Using key-points, which are generated out of markers, was done in [10]. Alnabhan et al. [1], for example, use a reference point system where the system stores the direction from each point towards the target. The shortest path is then defined by calculating the number of references between the start and end point. For the definition of the reference points, Alnabhan et al. combined them with the measurement points of the tracking system. This binds the path planning algorithm very tightly to the tracking system and is therefore hard to integrate into other systems.

In contrast to a graph-based solution, the concept of navigation meshes opens up more flexibility in calculating paths. Snook et al. [3] introduced navigation meshes in 2000. Different representations have been developed since then in order to represent a walkable area within a 3D mesh.

The Visibility–Voronoi diagram [16], for example, is a hybrid between the visibility graph and the Voronoi diagram of polygons in a plane. The visibility graph on the one hand represents the shortest path between obstacles in the indoor environment. The Voronoi diagram on the other hand is able to create a path with maximum clearance to obstacles. These properties can be very important in an AR application.

In favor of runtime performance, some approaches separate the two necessary steps of the generation of the navigation and the path planning algorithm. Kallmann et al., for example, use a Delaunay triangulation of the input geometry, thereby considering a clearance value in order to generate a triangulated navigation mesh called Local Clearance Triangulation [17,18]. Geraerts et al. on the other hand presented Explicit Corridor Maps [19] based on a medial axis [20]. The medial axis is related to a Voronoi diagram and represents points that are equidistant to two or more polygons. It represents the maximum clearance to obstacles and walls of the environment. The disadvantage of using the medial axes is that they have to be calculated for each path individually at runtime. These approaches work on planar surfaces. A 3D environmental model is split up into multiple planar layers and a separate navigation mesh is generated for each layer. This allows the calculation of paths within a multistory building.

Similar to the work from Geraerts et al. [21], most of the above-mentioned algorithms are used to either do crowd simulation, robot guidance, or for avatars to walk through a virtual environment. None of the algorithms had the requirement of visualizing the path as a guiding instrument in a real walking AR application. For this reason, the visibility and the human factors were not considered at all.

4. Proposed System

Creating a navigation aid for indoor environments is a challenging task, since multiple components have to work together in order to achieve a seamless and correct visualization for the user. The work at hand concentrates on the fact that AR glasses only provide a small window for displaying AR content. So, the main challenge is how to adapt a path starting from the user's position leading to the target in a way where the user always gets visual guiding information, independent of her/his viewing direction. In addition to that, all components necessary for navigation have to be real time capable, in order to be able to register the virtual content at the correct position in the real world.

For the creation of a custom path, we first need a correct 3D model of the environment in order to calculate walkable areas. Since users may also look directly at a wall, the 3D model must contain

floors, walls, and openings such as doors. To know where in the model walkable areas exist, the scale of the model has to match the real-world scale. This is especially important if, for example, staircases need to be included.

Concerning the final path, which is presented to the user, more information has to be provided. The current position and orientation of the user in the building have to be known in an AR-usable quality. Almost any tracking system capable of doing this, such as a SLAM approach or tracking systems based on RGB-D data (color images with depth data), can be used. Regarding the used AR device, we do not demand any special requirements, such as processing power or FOV area, since the target of the work focuses on both low- and high-quality devices. The main distinctive feature of state of the art AR display devices is the size of the field of view, which should at least be a known value for the presented work in order to be able to integrate it into the calculation. Another requirement that was defined by the authors was the modularity of the developed approach. This makes it possible to exchange individual components, such as the tracking system or the AR display device. The modularity of this solution is especially important, because different application areas also demand different technologies. It is also important to define standards in the area of indoor navigation, for example, as it is done in the OGC Standard for Indoor Spatial Information [22]. A modular system such as DARGS can easily be integrated into such a specification.

4.1. System Design

In this chapter, the framework of the presented system is described in detail. The concept of DARGS can be divided into three major modules that work together during the guiding task in order to react to real-time data. As mentioned before, DARGS was designed to be able to be used with different tracking technologies and AR display devices. For that reason, a server-client architecture was chosen, thereby keeping the load and the requirements of the display device low.

Figure 2 gives a rough overview of the proposed system. The server is in charge of two modules responsible for calculating the path and preparing the data for the user side. Module one, the "Navigation Toolbox", holds the core functionalities of the system. It prepares the 3D model of the environment for calculating paths and contains the main functionality of the FOVPath calculation. The "Navigation Toolbox" also reacts to special circumstances as a result of the motion of the user. To achieve an FOV-dependent path, the algorithm renders the 3D model of the environment in a depth texture and projects the FOV of the user into this model. As a result, a smaller area of the navigation mesh is used in order to prepare a position- and orientation-specific representation of the path. This approach is compiled to a library, which is intended to be used in various applications, and is based on game engines such as Unity 3D or Unreal.

For this work, the authors decided to use the Unity 3D game engine for integrating the "Navigation Toolbox" on the server side and to control the client module ("Guiding Application"). The client runs wirelessly on an AR device such as the Microsoft HoloLens. The second and central module is the servicing point for loading the 3D model, and it handles all parameters and the wireless communication via Wi-Fi to the AR device. It is also responsible for receiving tracking data and decision-making regarding the visualization.

The third module runs exclusively on the client AR device. It is a Unity 3D application responsible for receiving and converting the FOVPath information into an understandable visual representation. In the presented scenario with inside out tracking, the "Guiding Application" is also responsible for sending the tracking information to the server.

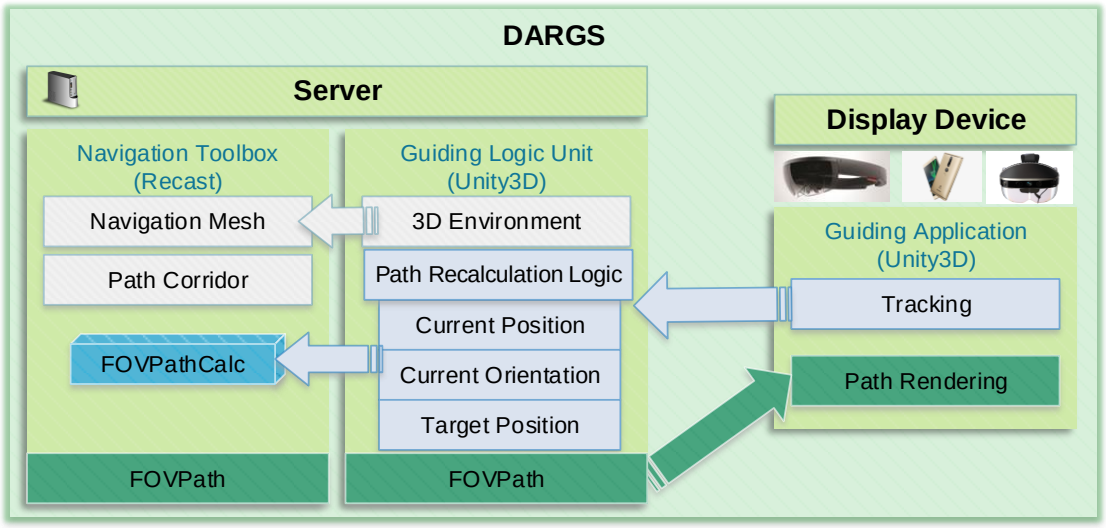


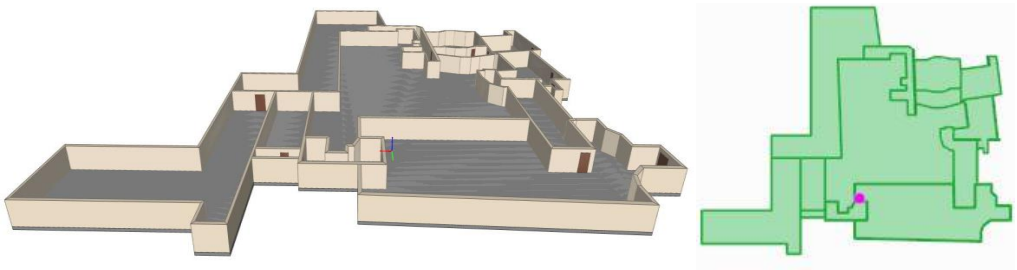
Figure 2. System Overview of the Dynamic Augmented Reality (AR) Guiding System (DARGS). The Server part on the left side is responsible for the path calculation. The client consists of a light weight implementation for an AR device mainly responsible for providing tracking data and path rendering. 3D: three-dimensional. FOV: field of view.

4.2. Implementation

The following chapter describes the functionality of the developed algorithm, starting from the 3D model and ending with the implementation of the path visualization.

4.2.1. Navigation Toolbox

The Navigation Toolbox, including the FOVPath, is implemented in C++ within the Recast library [23]. It is a toolbox for path planning tasks and contains fundamental tools for mesh generation and path planning methods amongst other things. The first step in the algorithm is to prepare the 3D model of the environment for the path calculation. As input for the 3D model, a triangulated mesh (see Figure 3a) is required. With the help of the Recast library, a rasterization step creates a voxel height field, where the size of each voxel has to be chosen accordingly. A denser voxel field has a big impact on the calculation time of all following steps. After that, individual voxels are connected to spans, which are categorized into walkable areas and non-walkable areas. These areas are then converted into regions (convex polygons), which consider static spaces to static obstacles such as walls. As a result, one connected or multiple separated regions can be detected. In the next step, the borders of the regions are simplified to speed up further processing steps. Finally, a navigation mesh is created by triangulating the results of the last step, which represents the walkable areas of the whole 3D model. This can be achieved with the Recast library.



(a) **(b)**

Figure 3. (**a**) 3D Model of an indoor environment with approximately 2400 m^2 used as a reference input model to the system; (**b**) shows the walkable area in this environment.

The proposed FOVPath is implemented on top of the navigation mesh. The final calculated FOVPath is composed of three sub paths (see Figure 4). The first path leads from the user to the FOV area. Next, the path inside the FOV area is calculated, and finally the area between the FOV and the target is used to finalize the path. Each path calculation is based on a path corridor, which is a sequence of mesh polygons connecting the starting position with the target position. To obtain this path corridor in a first step, the graph searching algorithm A* is used with the entire navigation mesh as input. The graph itself consists of all edge midpoints of the navigation mesh.

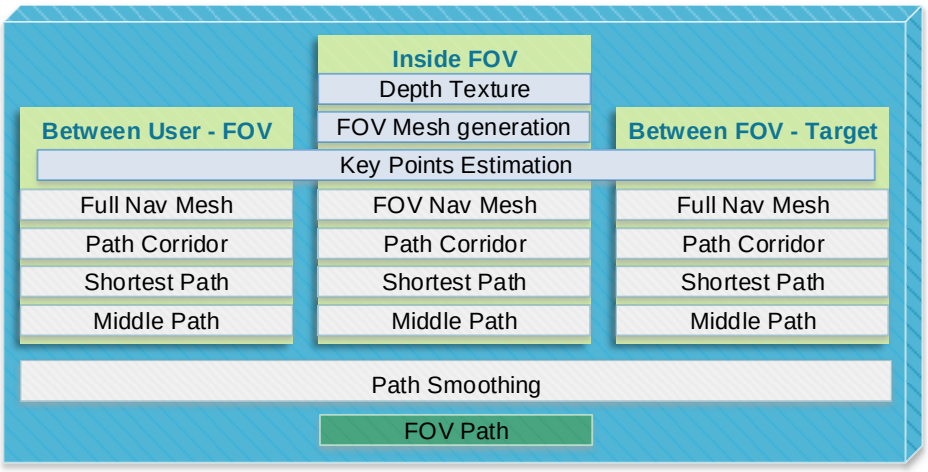


Figure 4. FOVPath calculation process inside DARGS. The FOVPath is composed of three sub paths covering the area between the user and the FOV - inside the FOV area - between the border of the FOV and the target.

A* provides a very short and direct path to the target; therefore, it can be used to define the path corridor. However, the resulting path of A* almost cuts edges and often runs right next to a wall (see red path in Figure 5), and can therefore not be used for the guiding application. Clearance to walls and other obstacles is already given due to the region calculation, which means that the A* path runs parallel to wall at a certain distance. This distance to walls, however, must not be too wide, since this space in between is defined as a non-walkable area. If this area is chosen to be too wide it has two negative effects in the application area at hand. For one, it is not possible to calculate a path if the starting point is outside of the walkable mesh. So, if a person is standing next to a wall, the path planning is interrupted. The second issue could accrue at small passages such as doorways. If the clearance is chosen to be too wide, a doorway could separate the two sides into two different not-connected regions and no navigation is possible. For that reason, this clearance cannot be used to shape the path; therefore, it was necessary to implement another way to guide the path towards the middle of the room. To achieve a more centered path, the authors designed the Middle Path concept.

Starting from the corridor path, the Middle Path algorithm processes each polygon towards the target successively. In each step, the edge between two neighboring polygons is used to add a new point to the middle path. The point on this edge can be chosen in any desired relation. This relation influences on which side of the room the path is generated. For the application area at hand, an equal relation was chosen. It was also necessary to make sure the path is still situated within the boundaries of the walking area. Figure 5 illustrates the yellow path based on A* and blue path as result of the Middle Path calculation. Since this step has to be performed in each run, it was necessary to design a simple but fast method to tweak the path towards a certain area along the route. As the figure shows, the path passes the door at the center and is then drawn towards the center of the room. The concept of the Middle Path is used in all three sub paths as described before, but not using only the starting and ending points of the global path. Instead, several key points along the route and inside the current FOV area are estimated, which are used to apply the Middle Path multiple times.

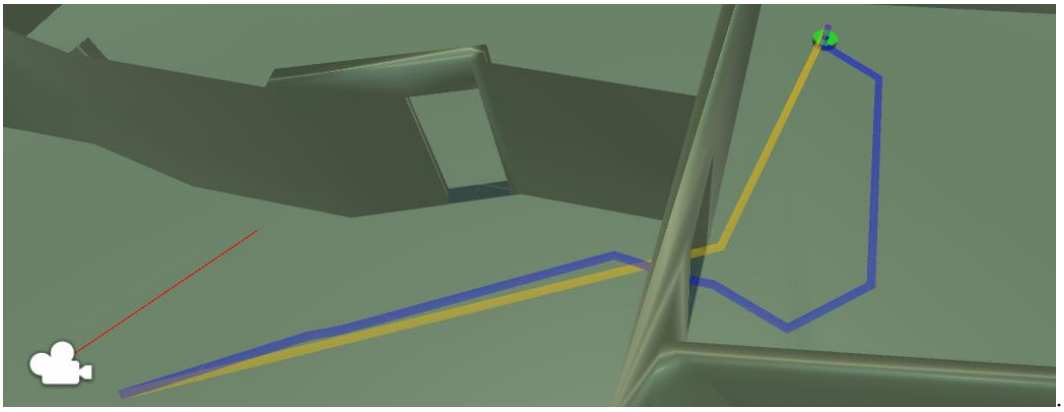


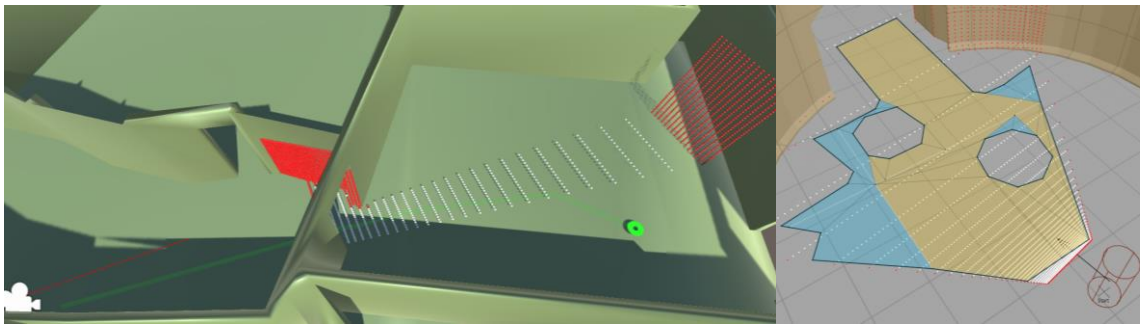
Figure 5. Yellow line on top shows a straight path based on the A* algorithm. The blue line in below shows the Middle Path dragging the path towards the center of the corridor. The red line indicates the viewing direction of the user.

The tracking system delivers a position and a viewing direction to the algorithm. In addition to that, the horizontal and vertical FOV of the used AR display have to be known. The aim of the FOVPath is to optimize only the part of the path that can be seen by the user. To achieve this, the area in the 3D model, which the user is looking at, has to be estimated. To handle this area in a separate way, it is necessary to construct a new navigation mesh, the FOVNavMesh. Since objects in the environment such as walls or other obstacles can occlude parts of the floor area in front of the user, it is necessary to shoot rays starting from the virtual position towards the viewing direction. Therefore, the 3D model is rendered into a depth texture. The rays are then projected to the environment according to the horizontal and vertical FOV of the AR device. A grid of 64 by 64 rays with a common starting position (position of the camera) creates several hit points on the 3D model. These hit points are then clustered into three categories: wall points, far points, and floor points (see Figure 6a). Only floor points are used for further processing at this stage. In order to generate the FOVNavMesh, the contour of all floor points is calculated with the help of the Moore–Neighbor tracing algorithm [24]. The result of this is a sum of pixels describing the boundary around the viewed area on the floor. As a next step, it is necessary to reduce these points. The Douglas–Peucker algorithm [25,26] is used to reduce the floor pixels to a minimum, since it considers the whole polyline to find a similar curve depending on a maximum distance between the original and the final polyline. As a result, only corner points remain, and they describe the contours of the FOV. This data structure is then treated like a new input into the navigation mesh generation and results in the FOVNavMesh. From now on, two navigation meshes are available, but only the FOVNavNesh dynamically changes.

The main advantage of dividing the path planning into these two navigation meshes is that the calculation of the FOVNavMesh is much faster due to the small size of the FOV in contrast to the whole building. In the worst case of fast head movements, every frame could trigger a recalculation of the FOVNavMesh. The advantage of speed is that in this area dynamic obstacles can also be included, which would be impossible to use for the whole navigation mesh. Dynamic obstacles are all real-world objects that are not modeled within the 3D environment, such as people, furnishings, or other moving objects. These can be detected with an RGBD camera, for example, and fed into the algorithm. The Recast library uses circular dynamic obstacles (see Figure 6b), which can now be included into the FOVNavMesh without losing much performance.

At this point in the algorithm, a full path is available based on the whole navigation mesh. In addition to that, a by far smaller navigation mesh is ready for integration of the proposed FOVPath. Within the FOV area, a number of interest points have to be estimated in order to make sure that it is possible to seamlessly connect the FOVPath with the full path and to make sure the path is built up in the right direction. In addition to that, it was also necessary to make sure enough understandable

content is generated inside the FOV area, so that the user is able to make sense of the visualized information. To achieve this, first a center point of the FOV area is calculated. Therefore, the centroid of each triangle of the FOVNavMesh is summarized to a point in the center of the mesh. However, a test is necessary before continuing, since any obstacle or concave shape of the FOVNavMesh could lead to a center point in a non-walkable area. In that case, the center point is moved over the closed border of the mesh.



(**a**) (**b**)

Figure 6. (**a**) Illustration of the FOV calculation process. White points are hit points with the global navigation mesh. Red points represent the area outside a walkable area. The point-shaped plane in the distance represents the max length of the FOV. The green line represents the FOVpath. (**b**) Final FOVNavMesh including two round obstacles.

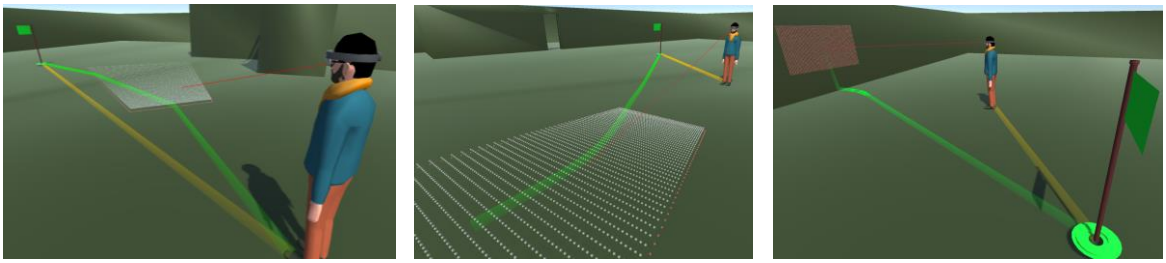
As a next step, the entry and exit points of the FOV area are defined. The entry point on the one hand is set on the intersection between the FOV contour and a line connecting the center point and the position of the user. On the other hand, the exit point is dependent on the arrangement of the FOV area and obstacles. To calculate the exit point, a Middle Path is calculated, starting with the center point and with the target as the final point. The first vertex outside of the FOV area is marked. If there is no obstacle between the marked point and the center point, the intersection between these two and the contour of the FOV is the exit point. In case of an intersection with a wall, for example, the exit point is set between the marked point and the last point of the Middle Path still inside the FOV.

During the design and testing of the algorithm, special conditions were defined. These conditions had to be detected and treated separately during the process of the FOVPath generation and mainly influence the previously described interest points.

1. The most common condition exists if the user is looking in direction of the shortest path (A*), which leads directly to the target, not considering the viewing direction. In this case, the Middle Path is calculated between the entry and exit point of the FOVNavMesh. This path is then combined with a Middle Path calculation between the exit point and the target and the position of the user and the entry point (see Figure 7a).
2. If the user looks away from the shortest path, the path inside the FOV does not start at the entry point near the user, but at the center point of the FOV area towards the exit point. This thereby prevents the creation of a path in the shape of a loop (see Figure 7b). This calculation is triggered due to the angle between the viewing direction and the average direction of the first five points of the shortest path.
3. In case the user is looking at a wall, where no navigation mesh is available, the authors designed the concept of a Wall Path. Figure 7c illustrates the outcome of this principle. It is necessary to generate a connection from the point the user is looking at towards a valid global path. Three points are defined before the Wall Path can be continued to the target. At first the point, where the center ray of the depth texture, representing the center of the FOV, hits an obstacle is calculated. This point is set as the first point of the new path. When projecting this point to the floor, the line

between the projected point and the user's position is calculated. The intersection point between this line and the edge of the navigation mesh is added as a second point of the path. A third point is added with a clearance distance to the wall and is also used as a starting point for a global path calculation.

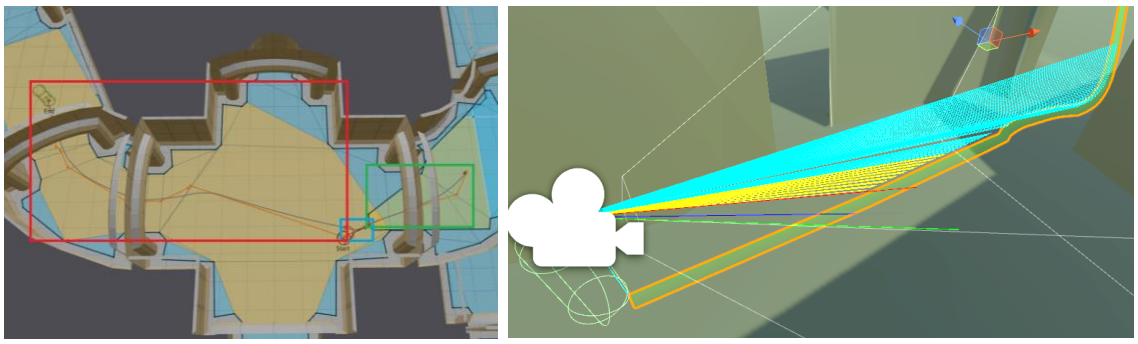
4. If the user is already close to the target, the destination could be inside the FOV area. In this case, the path is only calculated between the entry point and the target.
5. The final situation appears when the user is walking towards the target and the FOV is already behind the target position. If this situation is detected by the algorithm, the center point of the FOV area is used as a starting point of the path. Additionally, the path shows the way towards the user.



(**a**) (**b**) (**c**)

Figure 7. (**a**) Representation of the most common situation without path smoothing; (**b**) Situation in which the user is looking away from the shortest path (yellow); the green FOVPath is calculated. (**c**) Illustration of the Wall Path concept. The first point is set to the most center point of the FOV hitting a wall. Second point: closest point on the navigation mesh. Third point: set to save distance from the navigation mesh for the smoothing process.

In a usual situation, the final path is merged out of three Middle Paths (see Figure 8a). The path before the FOV, the path inside the FOV, and the path after the FOV. The whole path is always calculated, although the user only sees content in the FOV area. The reason for that is that the path can stay valid over a longer period of time. As long as enough information is present inside the FOV area, no recalculation is necessary and the old path stays valid.



(**a**) (**b**)

Figure 8. (**a**) Illustrating three areas of the path calculation process. The small rectangle in the middle describes the area between the user and the FOV. The green rectangle on the right represents the FOV area (the user is looking away from the target). The big red rectangle shows the path area between the FOV and the target. (**b**) The path recalculation process considers the number of lines of sight to the path points (yellow lines) in relation to the number of not seen path points (outside FOV or occluded by obstacles) within a certain area.

After fusing all sub paths, a final smoothing step is performed. We used an algorithm related to the corridor map method of Geraerts et al. [19]. As a first step, a backbone path is calculated, which consists of the three sub paths put together. Additional waypoints are added at constant distances along the path. For every point, the distance to the closest obstacle is calculated. An attraction point is used, which is at the same distance away. By calculating a force and velocity, a new smoothed point is added to the now final FOVPath. Figure 9 shows the overall procedure executed at runtime explaining the concept of the FOVPath calculation as well as the WallPath calculation.

```
procedure FOVPathCalculation
    const SizeFOV = [horizontal, vertical]
    CurrPose = UserPosition, UserViewingDirection
    while <FOVPath outside FOVArea(CurrPose)> DO
        render FOV inside depth texture(currPose)
        FOVAre = calc FOVmesh
        IF <FOVArea outside walkable area>
            //calc WallPath
            wall points = raycast (CurrPose, wall mesh)
            path = combine (middle path, wall points)
            FOVPath = path.smooth
        ELSE <FOVArea part of walkable area>
            //calc FOVPath
            case user-target relation
                user-target: destination in viewing direction
                    set keypoints[3]
                user-target: destination behind the user
                    set keypoints[3]
                user-target: destination inside FOVarea
                    set keypoints[3]
                user-target: target between user and FOVarea
                    set keypoints[3]
            END
            FOR <subpath = 1 to 3> DO
                calc MiddlePath (keypoints[3])
            END
            path = combine(SubPaths[3])
            FOVPath = path.smooth
        END
    END
    show(FOVPath)
END procedure
```

Figure 9. Pseudocode describing the calculation procedure of the WallPath and the FOVPath.

4.2.2. Guiding

All of the above-described logic is handled inside the library, which is meant to be connected to any game engine for further visualization. The guiding part of the presented approach is implemented in the Unity 3D game engine, which thereby acts as a server unit on a stationary personal computer (PC) or notebook and as a client application running on an AR device. In the course of this work, we made use of the Microsoft HoloLens as a client AR device.

By using a global initializing point of the client device in the 3D model for the HoloLens, it was possible to align the real-world coordinate system (HoloLens) with the coordinate system of the 3D model environment running on the server. From that initialization point, the HoloLens tracks its own pose through the environment. In periodic intervals, which can be set between 5 and 30 updates per second, the new pose of the HoloLens is sent to the server. In this case, the real environment can be simulated on the server part with the 3D model of the environment without the need for transferring the reconstructed mesh of the HoloLens.

The server part is responsible for decision-making if a new path has to be calculated or the old path is still visible for the user. Therefore, a path recalculation model was implemented in Unity 3D, which monitors the length of the path still located in the current FOV area. To achieve this, a round area is defined around the current user position independent of the current FOV. The amount of path

points within this area is counted. This number has to be compared to the number of path points currently seen by the user. An area according to the horizontal and vertical FOV of the device is spanned in front of the user. By creating vectors between the device position and each path point, it is possible to decide whether a point is within the FOV or not by calculating the dot product. In addition to that, it is necessary to do a raycast towards every path point, since obstacles in the room could easily occlude high numbers of points. Figure 8b illustrates this concept. The ratio between all points within a radius and the visible points decide whether a recalculation is necessary or not. In this way, it could be that only one path has to be created for the entire navigation task, but only if the user is not losing the path out of sight. The Unity 3D application running on the AR device receives a new path as soon as it changed on the server side.

On top of this FOVPath, a visualization was implemented. In a navigation task the direction is essential, which has to be communicated to the user. In the course of this work, we focus on a pure visual concept without textual information or audio support. In addition to that, the requirement of a low calculation performance and a good understandability is necessary.

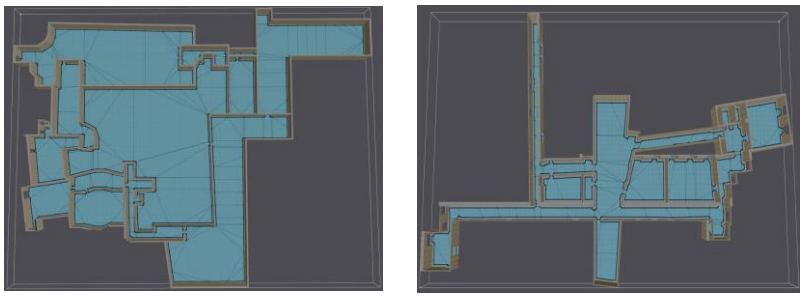
For that reason, a particle path was implemented, which consists of colored points flowing from the start point towards the target dependent on the speed of the user. Since it is possible to recalculate the path several times, at each waypoint the particles are created immediately. Figure 12c shows a screenshot of a user observing the visualization through the HoloLens.

5. Results

The proposed system was evaluated in two stages. The first part describes a pure technical evaluation targeting runtime performance in various situations, thereby identifying parameters with great impact on the performance. In a second stage, DARGS was set up in a real environment for a user evaluation, where each user was given two different navigation tasks through a 200 m^2 environment. The technical evaluations were performed on a Schenker XMG notebook with an Intel i7-4940 CPU, 16.0 GB RAM, and a GeForece GTX880M. The user study used in addition the Microsoft Hololens, a Wi-Fi router, and the HoloLens clicker input device for simple user interaction.

5.1. Technical Evaluation

For the technical evaluation, two different 3D models of indoor environments were used. Figure 10a shows a 2500 m^2 environment with about 11.000 vertices, whereas Figure 10b illustrates a 700 m^2 environment with only 1.000 vertices. The calculation of the whole navigation mesh takes several seconds up to several minutes for the whole building, and is dependent on the cell size used. Since this has only to be done once in an offline process, it is not discussed in detail.



(**a**) (**b**)

Figure 10. Two possible outcomes (**a**,**b**) of a navigation mesh used in the technical evaluation. (**a**) Environment used in the user study. (**b**) Environment with multiple corridors.

In the following, we focus on the algorithm executed during runtime. Especially, two factors in the process of the FOVPath calculation can influence the runtime. First, the size of the FOV area, and

second, the length of the path Table 1 shows all of the necessary steps that have to be performed during runtime that are dependent on FOV size and path length. The left side of the table uses a fixed path length and a variable FOV size of 20 m^2 and 120 m^2. The FOVNavMesh generation is a limiting factor in the algorithm. With a calculation time of 262 ms at 120 m^2, it is far from real-time performance. A reasonable FOV size in terms of calculation time and visualization can be seen in Figure 10a. With about a 20 m^2 FOV size, a reasonable calculation time of 100 ms per path can be guaranteed. To reduce the size of the FOV during runtime, the algorithm is able to limit the maximum distance of the hit points in the depth texture, thereby decreasing the size of the used FOVarea.

Table 1. A performance comparison of all parts within the FOVPath calculation that are running during the guiding process. The two variables varying during this process are analyzed: the path length and the size of the FOV.

Stage of the Algorithm	Path Length 50 m		FOV Size 10 m^2	
	FOV Size 20 m^2	FOV Size 120 m^2	Path Length 50 m	Path Length 160 m
Path Corridor	0.05 ms	0.05 ms	0.06 ms	0.20 ms
Middle Path	0.02 ms	0.02 ms	0.03 ms	0.05 ms
Render FOV	**22.2** ms	**23.8** ms	**22.1** ms	**22.2** ms
Filter Floor Pixels	0.27 ms	0.09 ms	0.12 ms	0.1 ms
FOV NavMesh	**44.8** ms	**262.8** ms	**25.0** ms	**26.6** ms
FOV Path	0.07 ms	0.05 ms	0.03 ms	0.04 ms
Smooth FOV	0.24 ms	2.1 ms	0.4 ms	0.5 ms
Global Path	0.10 ms	0.08 ms	0.12 ms	0.4 ms
Global Smoothing	**19.4** ms	**14.6** ms	**17.9** ms	**159.9** ms
Full Runtime	92.0 ms	308.3 ms	70.5 ms	216.1 ms

The second variable influencing the speed of the FOVPath calculation is the path length. Therefore, a path length of 50 and 160 m is compared in Table 1. It is obvious that the calculation time of the smoothing process is highly influenced by the path length. Depending on the desired calculation effort, it is possible to limit the smoothing process to a certain distance since the user is not able to see the whole path at once. Figure 10b plots the overall calculation time depending on the length of the path. With about an 80 m path length, the calculation time is measured with 100 ms.

Within the working region of about a 10 m^2 to 40 m^2 FOV and a reasonable global path length of 20 m to 100 m, the calculation time increases in a linear way for both factors. In order to limit the whole calculation time, a balance between the size of the FOV area and the global path length smoothing process can be found. In a usual situation with an FOV size of 10 m^2 and a smoothed path length of 50 m, about 14 paths per second can be provided for the display device. Within the FOVPath algorithm, the parameters maxPathLength and maxFOVsize can be adjusted depending the available performance of the system. How often it is necessary to recalculate the FOVPath was tested amongst other things in the real-world scenario with 16 participants.

5.2. User Study

The main goal of the user study was to find out if a user can be guided with the developed dynamic AR path through a real environment. The hypothesis—if an FOV-dependent calculation facilitates the way finding process in indoor environments—has to be answered. Furthermore, the study should show if a user needs navigational information all the time or not. Within the user study, first the users had to accomplish multiple guiding tasks while wearing the Microsoft HoloLens. After that, a detailed questionnaire was filled in, containing questions concerning the tasks and general question about the concept. The questionnaire is based on the seven-point Likert scale [27]. Finally, the standardized system usability scale (SUS) [28] is used to rate the whole system.

The user study consisted of two guiding tasks. Within the first task, the users were placed in a 200 m^2 environment. Each participant had to follow two different path types through this environment (see Figure 11a). Half of the users started with the FOVPath followed by the A* path and the other

half of the participants performed the task the other way round. In each run, five different targets (see Figure 11b) with the same overall distance had to be found after a short introduction. To guarantee that the user saw the virtual target in the real environment, they had to write down a code written on the target. In addition to that, each user was holding the HoloLens Clicker and was instructed to press the button if the path should be shown. The path visualization for both sub tasks is illustrated in Figure 12c. This task was performed by 16 participants, 4 females and 12 males, between the age of 24 and 50. For eight participants, it was their first time using an AR device.

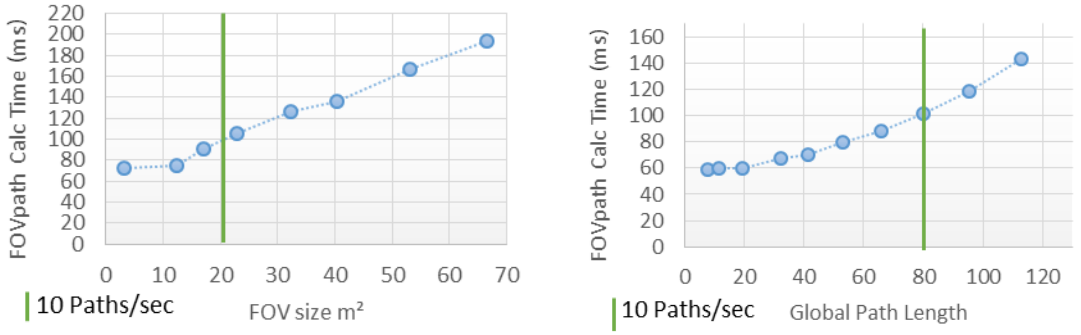


Figure 11. (**a**) Graph showing the calculation time of the FOVPath in relation to the size of the FOV area. The path length is fixed to 50 m. (**b**) Illustration of the FOVPath calculation in relation to the size of the global path length. The FOV area is fixed to 10 m^2.

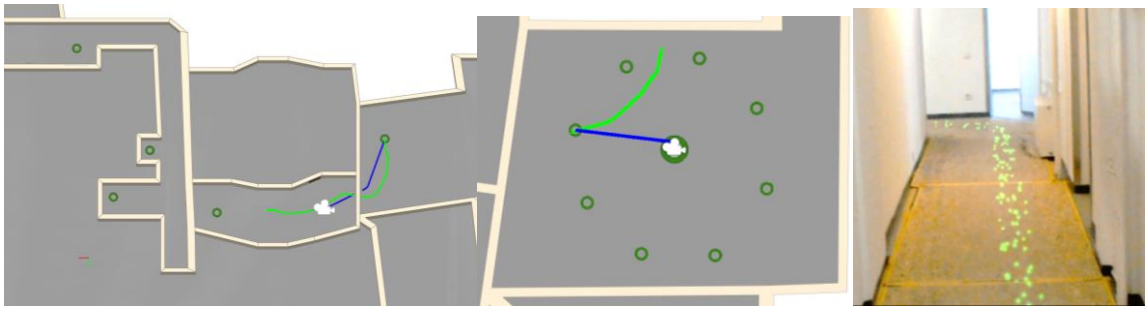


Figure 12. (**a**) Area of 160 m^2 used in scenario one containing five green targets the user had to reach. (**b**) Area of 65 m^2 used for the second scenario. The big green circle represents the position the user is standing. The small circles are the targets in the room; (**c**) A view through the camera of the HoloLens with a visualized path.

All participants were able to find all targets placed in the environment with the FOVPath as well as with the A* path. The question about if the visualized FOVPath was intuitive to follow was answered with a mean of 6.37 (with $\sigma = 0.80$), an almost perfect result. The constant path changes, due to head rotations, were no problem for a majority of the participants, as they got used to the path immediately and indicated that it is intuitive to follow.

Since every user could decide with a click of a button at which time to have the path visualized, it was possible to ask if people would rather see the path all the time, or just at certain points. Four participants out of 16 would prefer to see the path only on demand. Overall, this answer resulted in a mean value of 4.8 (with $\sigma = 2.28$) (see Figure 12a), which tells us that the majority of users would like to see the path all the time. This would mean that a user has multiple starting points between the start position and the target, each time not knowing where the path would be shown.

After the users finished the first scenario, a large number of participants reported that the difference between the two runs is minor and was hardly recognized, which was also reflected in

the results, for example in the question if the participant would use one or the other approach. The difference of the mean value between these two questions was only 0.16 points.

The duration of the first task was about 160 s with no measurable difference between the two path calculations. During this time, the FOVPath was calculated 377 times (median value), which results in one path every 2.52 s. The average walking speed, including the stopovers for writing down the code on the target, was also very similar with 1.98 km/h at the A* path and 2.22 km/h with the FOVPath.

Since the results of the first task proved that the FOVPath can lead people to arbitrary targets, the second guiding task was designed to study the advantage of the FOVPath. This mainly arises at the beginning of a navigation task or when the user is again asking for guidance information after a while. Twelve participants (3 female and 9 male) between the ages of 24 and 50 participated in this study. For this task, the user had to stay in the middle of a room (see Figure 11b) looking at a certain direction. Without walking around, the user had to identify eight targets in the room located around the user. Each user was testing the FOVPath and the A* straight path. Whenever a target was visible to the user, a click had to be performed, which activated the new target in the room until the final target was reached.

In this second scenario, the results indicate that in contrast to the straight path the FOVPath was rated significantly better. This can be seen in Figure 12b showing the result of the following question: How good could you anticipate the direction to the target? The FOVPath was rated with a score of 6.08 (with $\sigma = 1.73$) in contrast to the straight path with a score of 2.83 (with $\sigma = 1.74$). Also, the time for completing the task (see Figure 13) differs significantly between the A*-based straight path and the FOVPath. A median of 33.88 s was measured for completing the task with the straight path in contrast to 23.32 s for the FOVPath.

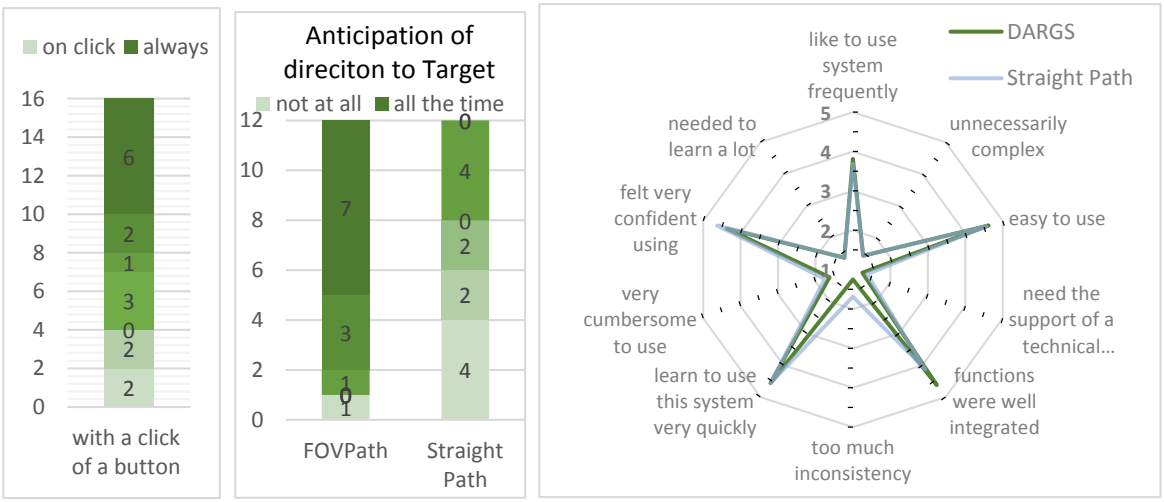


Figure 13. (**a**) Bar chart representing the results of the question if the user wants to see the path all the time or with click of a button (Scenario 1). (**b**) Results of the question of how good the target direction could be anticipated with the path. (**c**) A breakdown of the results of the system usability scale (SUS) for the individual questions and comparing both path calculation concepts.

In this scenario, it could be observed how users adapted their behavior to the visualization of the path. People guided by the A* path used in general two strategies to find the next target. Either they looked at their feet and turned to find the path, or they ignored the visualization and immediately rotated around their own axes until the target was in the FOV. Most people with the FOVPath immediately knew in which direction to turn and did not look down at their feet, but rather at the distance.

The first and the second scenario show that the participants did not notice a big difference between the A* algorithm and our FOVPath algorithm when being guided through the environment. The calculated FOVPath is able to guide the user to several targets without showing a negative effect on the results. The second scenario, however, shows a significant difference between the scores of the two paths. This scenario focuses on the uncertainty of the participants at the starting point when the direction towards the target is not clear yet. Therefore, the FOVpath can especially be helpful as long as the direction towards a target is not clear.

The usability of the implementation was tested with the standardized system usability scale. Figure 12c illustrates 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 the participants felt confident in using the guidance system.

It is interesting that the participants gave a higher inconsistency level to the shortest path (A*) implementation, although the path changes less often in comparison to the implemented FOVPath. According the SUS standard, a score value of 68 is considered above average usability [28]. Within the user study, an average SUS score of 87.66 (with $\sigma = 10.06$) for the FOVPath and 84.84 (standard deviation $\sigma = 12.58$) for the shortest path was achieved (see Figure 14).

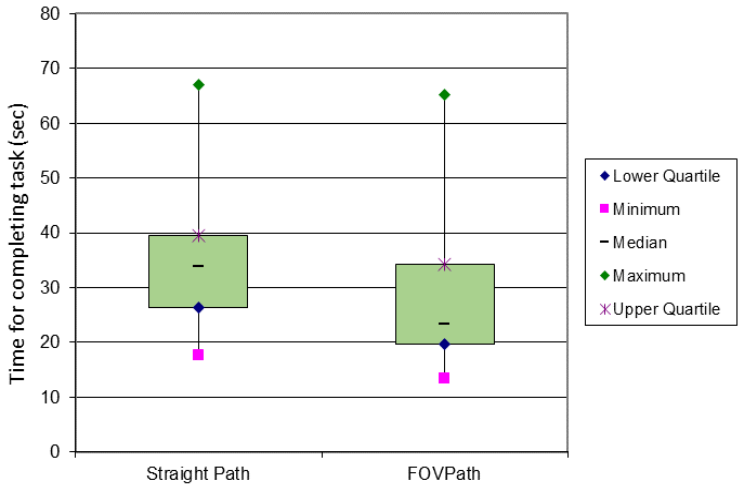


Figure 14. Comparison between duration of the second task with straight path in contrast to the FOVPath.

6. Conclusions

We presented a complete system DARGS for guiding people in indoor environments with AR. The system provides a dynamic path calculation, which is dependent on the viewing direction of the user, thereby guaranteeing that the path is always within the FOV of the used device. DARGS is designed to be integrated as a modular system into any application and is not dependent on the type of tracking system. It is also possible to extend the FOVPath with various guiding methods. In this work, we implemented a particle system as visualization, but it can also be replaced with a guiding avatar, any kind of animated object, or just with arrows along the path.

The first evaluation presented in this work showed that it was possible to guide people through an indoor environment with the FOVPath without any further instructions. It also showed that it is hard to measure the actual performance in a real scenario. Measuring time is not sufficient enough to evaluate the system, since people looking in a wrong direction always have to turn around. The only difference is if they are just searching for information or if they are provided with turning information. When talking to participants, especially after the second task, it was clear that they felt much more confident and the results also showed a better performance in time for completing this task. In addition

to that, they were able to better anticipate the direction to turn to in order to find the correct way. This gives a much better feeling when being guided.

As a next step, it would be interesting to use different devices, such as a smart phone, in order to compare the applicability in application areas where no AR glasses are available. In addition to that, we also noticed that, during the study, the visualization of the path influences the behavior of the users. Depending on the complexity, a visualization containing only arrows, floating objects, or even a human avatar could influence the learning curves of the users in a different way.

Acknowledgments: This study was supported by research funds from the Vienna Science and Technology Fund (WWTF—ICT15-015). In addition to that, we would like to thank all participants of the user studies for their time.

Author Contributions: Georg Gerstweiler and Karl Platzer performed the manuscript's preparation, implemented the methodology and the main algorithm, and conducted the user studies. Hannes Kaufmann supervised the work. All authors discussed the basic structure of the manuscript and approved the final version.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Alnabhan, A.; Tomaszewski, B. INSAR. In Proceedings of the Sixth ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness (ISA '14), Fort Worth, TX, USA, 4–7 November 2014; ACM: New York, NY, USA, 2014; pp. 36–43.
2. Gerstweiler, G.; Vonach, E.; Kaufmann, H. HyMoTrack: A mobile AR navigation system for complex indoor environments. *Sensors* **2015**, *16*. [CrossRef] [PubMed]
3. Snook, G. Simplified 3D Movement and Pathfinding Using Navigation Meshes. In *Game Programming Gems*; DeLoura, M., Ed.; Charles River Media: Newton Centre, MA, USA, 2000; pp. 288–304.
4. Hart, P.E.; Nilsson, N.J.; Raphael, B. Correction to "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". *ACM SIGART Bull.* **1972**, 28–29. [CrossRef]
5. Dijkstra, E.W. A note on two problems in connexion with graphs. *Numer. Math.* **1959**, *1*, 269–271. [CrossRef]
6. Al Delail, B.; Weruaga, L.; Zemerly, M.J. 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), Macau, China, 4–7 December 2012; pp. 286–290.
7. Kasprzak, S.; Komninos, A.; Barrie, P. Feature-based indoor navigation using augmented reality. In Proceedings of the 9th International Conference on Intelligent Environments, Athens, Greece, 16–17 July 2013; pp. 100–107. [CrossRef]
8. Huey, L.C.; Sebastian, P.; Drieberg, M. Augmented reality based indoor positioning navigation tool. In Proceedings of the 2011 IEEE Conference on Open Systems (ICOS), Langkawi, Malaysia, 5–28 September 2011; pp. 256–260. [CrossRef]
9. Reitmayr, G.; Schmalstieg, D. Location based applications for mobile augmented reality. In *Proceedings of the Fourth Australasian User Interface Conference on User Interfaces 2003*; Australian Computer Society, Inc.: Darlinghurst, Australia, 2003; pp. 65–73.
10. Mulloni, A.; Seichter, H.; Schmalstieg, D. 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), Stockholm, Sweden, 30 August–2 September 2011; p. 211.
11. Kim, J.; Jun, H. Vision-based location positioning using augmented reality for indoor navigation. *IEEE Trans. Consum. Electron.* **2008**, *54*, 954–962. [CrossRef]
12. Möller, A.; Kranz, M.; Diewald, S.; Roalter, L.; Huitl, R.; Stockinger, T.; Koelle, M.; Lindemann, P.A. Experimental evaluation of user interfaces for visual indoor navigation. In Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems (CHI '14), Toronto, ON, Canada, 26 April–1 May 2014; ACM Press: New York, NY, USA, 2014; pp. 3607–3616.
13. Möller, A.; Kranz, M.; Huitl, R.; Diewald, S.; Roalter, L. A Mobile Indoor Navigation System Interface Adapted to Vision-based Localization. In Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia (MUM '12), Ulm, Germany, 4–6 December 2012; ACM: New York, NY, USA, 2012; pp. 4:1–4:10.

14. Miyashita, T.; Meier, P.; Tachikawa, T.; Orlic, S.; Eble, T.; Scholz, V.; Gapel, A.; Gerl, O.; Arnaudov, S.; Lieberknecht, S. An augmented reality museum guide. In Proceedings of the 7th IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2008), Cambridge, UK, 15–18 September 2008; pp. 103–106.
15. Rehman, U.; Cao, S. Augmented Reality-Based Indoor Navigation Using Google Glass as a Wearable Head-Mounted Display. In Proceedings of the 2015 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Kowloon, China, 9–12 October 2015; pp. 1452–1457.
16. Wein, R.; Van Den Berg, J.P.; Halperin, D. The visibility-Voronoi complex and its applications. *Comput. Geom. Theory Appl.* **2007**, *36*, 66–87. [CrossRef]
17. Kallmann, M. Shortest Paths with Arbitrary Clearance from Navigation Meshes. In Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Madrid, Spain, 2–4 July 2010; pp. 159–168.
18. Kallmann, M. Dynamic and Robust Local Clearance Triangulations. *ACM Trans. Graph.* **2014**, *33*, 1–17. [CrossRef]
19. Geraerts, R.; Overmars, M.H. The corridor map method: Real-time high-quality path planning. In Proceedings of the 2007 IEEE International Conference on Robotics and Automation, Roma, Italy, 10–14 April 2007; pp. 1023–1028.
20. Preparata, F.P. The Medial Axis of a Simple Polygon. In *MFCS 1977: Mathematical Foundations of Computer Science 1977*; Gruska, J., Ed.; Lecture Notes in Computer Science; Springer: New York, NY, USA, 1977; Volume 53, pp. 443–450.
21. Van Toll, W.; Cook IV, A.F.; Geraerts, R. Navigation meshes for realistic multi-layered environments. In Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2011), San Francisco, CA, USA, 25–30 September 2011; pp. 3526–3532.
22. Lee, J.; Ki-Joune, L.; Zlatanova, S.; Kolbe, T.H.; Nagel, C.; Becker, T. *OGC IndoorGML. Draft Specification OGC*; v.0.8.2; Open Geospatial Consortium Inc.: Wayland, MA, USA, 2014.
23. Mikko Mononen Recast Library. Available online: https://github.com/recastnavigation/recastnavigation (accessed on 24 December 2017).
24. Pavlidis, T. Algorithms for graphics and image processing. *Proc. IEEE* **1982**, *301*. [CrossRef]
25. Ramer, U. An iterative procedure for the polygonal approximation of plane curves. *Comput. Graph. Image Process.* **1972**, *1*, 244–256. [CrossRef]
26. Douglas, D.H.; Peucker, T.K. 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*; John Wiley & Sons: New York, NY, USA, 2011; pp. 15–28, ISBN 9780470681749.
27. Likert, R. A technique for the measurement of attitudes. *Arch. Psychol.* **1932**, *22*, 55.
28. Brooke, J. SUS—A quick and dirty usability scale. *Usability Evaluation in Industy* **1996**, *189*, 4–7. [CrossRef]