Navigating to real life objects in indoor environments using an Augmented Reality headset

Maximilian Bågling

May 22, 2017
Master’s Thesis in Computing Science, 30 HP
Internal Supervisor: Juan Carlos Nieves
External Supervisor: Johan Westerlund
Examiner: Henrik Björklund

UMEÅ UNIVERSITY
DEPARTMENT OF COMPUTING SCIENCE
Abstract

Augmented Reality (AR) headmounts are an rising technology with great chances to be a common gadget used by a lot of people in the near future. With the rise of this technology, new possibilities opens up on how to combine the interaction of the real world together with the virtual world. This thesis meets some of these upcoming interaction questions in terms of indoor navigation. The thesis introduces an approach for designing and implementing an AR-based system that is able to let users navigate around an indoor environment to find various real life objects while using an Augmented Reality headmount. The thesis also discusses how to personalize the navigation to different users in different environments. A proof-of concept was implemented and evaluated with several users inside different indoor environments, e.g., a real food store, where the results showed that users were more effective finding objects while using the AR-based system, compared to when not using the AR-based system.
Acknowledgements

I would like to thank everyone at Codemill for giving me the opportunity and necessary tools to do this thesis. I would also like to thank my supervisor Juan Carlos Nieves Sánchez for our interesting discussions around the subject, and for providing a lot of helpful feedback. Last but not least I want to give a huge thanks to my friends and family who has been supportive throughout the whole study time.
# Contents

1 Introduction .......................... 1
   1.1 Codemill .................................. 1
   1.2 Problem Statement ......................... 1
   1.3 Outline .................................. 2

2 Background ........................... 5
   2.1 Augmented Reality ......................... 5
   2.2 Computer Vision .......................... 6
     2.2.1 Application areas .................. 7
     2.2.2 Common tasks ......................... 7
     2.2.3 Techniques ......................... 8
   2.3 Object tracking .......................... 9
     2.3.1 Object representation ................. 9
     2.3.2 Image features ....................... 10
     2.3.3 Object detection ..................... 10
   2.4 Planning ............................... 11
     2.4.1 Planners and Plans ................... 12
     2.4.2 Pathfinding .......................... 12
       2.4.2.1 Grids .............................. 13
       2.4.2.2 Search algorithms ................. 14
   2.5 Inside-out tracking ..................... 16
   2.6 Personalization ....................... 17
     2.6.1 Context awareness .................... 17
     2.6.2 Proxemics ........................... 18
     2.6.3 Personalizing with proxemic interactions ............. 18

3 Methods and Tools ................... 21
   3.1 Planning the project ..................... 21
     3.1.1 The preparatory Phase ................. 21
     3.1.2 Deciding devices and framework ........... 21
       3.1.2.1 Important features ............... 22
3.1.2.2 Proposed devices .................................................... 22
3.1.2.3 Proposed frameworks ............................................. 22
3.2 Final device, frameworks and tools .................................. 22
  3.2.1 HoloLens ............................................................... 22
    3.2.1.1 Hardware .......................................................... 23
    3.2.1.2 Inside-out tracking ........................................... 23
    3.2.1.3 Spatial mapping ............................................... 24
  3.2.2 Vuforia ................................................................. 24
  3.2.3 HoloToolKit .......................................................... 25
3.3 Workflow .................................................................. 26
  3.3.1 Working with the system .......................................... 26
  3.3.2 Design .................................................................. 26
3.4 User tests .................................................................. 27
  3.4.1 Office .................................................................. 27
  3.4.2 Store .................................................................. 28
4 Implementation ............................................................... 31
  4.1 The flow of the system .................................................. 31
    4.1.1 Setting up the navigation ....................................... 31
    4.1.2 Using the navigation ............................................. 32
  4.2 Planning .................................................................. 34
    4.2.1 Spatial Understanding .......................................... 34
    4.2.2 Graph representation ........................................... 35
    4.2.3 Algorithms .......................................................... 36
      4.2.3.1 A* ................................................................. 36
      4.2.3.2 Blur ............................................................... 38
      4.2.3.3 Bézier path ....................................................... 39
5 Results ......................................................................... 41
  5.1 Prototype .................................................................. 41
  5.2 User study ................................................................. 41
    5.2.1 Time .................................................................. 41
    5.2.2 Questionnaire ....................................................... 46
6 Discussion .................................................................. 49
  6.1 Locating objects .......................................................... 49
  6.2 Design and Personalization ........................................... 50
  6.3 Improvements ............................................................. 50
  6.4 Practical useful solutions ............................................. 51
7 Conclusion ..................................................................... 53
A Questionnaire: Navigating in a store using HoloLens
## List of Figures

2.1 A person wearing a HMD (HoloLens) and learning about the human body.  
2.2 A football game where AR is applied to show current important lines, the yellow and blue lines are added to the original image.  
2.3 Example of tracking a pedestrian with occlusion.  
2.4 A 2D square grid with 3 obstacles.  
2.5 A hexagon grid with 3 obstacles.  
2.6 Example of a finished Dijkstra’s run on a square grid.  
2.7 Example of a finished A* run on a square grid.  
3.1 How to do the tap-gesture.  
3.2 Representation of a mesh in the HoloLens.  
3.3 How it looks when running a simulation of the scan in the computer.  
4.1 Flowchart of how the system is initiated.  
4.2 Flowchart of how the system is used.  
4.3 An example of a processed 10x10m room, the green represents the mesh, and blue squares represent the floor positions of the room.  
4.4 A grid representation of a 10x10m room.  
4.5 Example of A* run on a 10x10m room with the node size 15cm. The blue dot represents the start position.  
4.6 Horizontal and vertical blur.  
4.7 A cubic Bézier curve.  
5.1 The view when starting the system.  
5.2 The view when a scan is big enough to start processing.  
5.3 The menu, showing the available options in the setup menu.  
5.4 The menu, showing the available options in the user menu.  
5.5 The blue line, representing the path.  
5.6 The green line, representing the path when close to the location.  
5.7 The red arrow, provides feedback to the user when facing the wrong direction.
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.8</td>
<td>The item to be found (The dark color within the square is not visible in the AR HMD)</td>
<td>44</td>
</tr>
<tr>
<td>5.9</td>
<td>The item marked as found (The dark color within the square is not visible in the AR HMD)</td>
<td>44</td>
</tr>
</tbody>
</table>
List of Tables

3.1 Microsoft HoloLens specifications[5] ......................................................... 23

5.1 User test Group 1, Set 1, no HMD. Time presented in seconds = s. ........ 45
5.2 User test Group 2, Set 2, no HMD. Time presented in seconds = s. ........ 45
5.3 User test Group 1, Set 2, with HMD. Time presented in seconds = s. ....... 45
5.4 User test Group 2, Set 1, with HMD. Time presented in seconds = s. ....... 46
5.5 Overview information, how easy test user’s thought different parts of the
    system was. Data is presented as number = n. ........................................... 46
Nomenclature

$AI$  Artificial intelligence

$AR$  Augmented Reality

$DTAM$  Dense Tracking and Mapping

$HMD$  Head-mounted display

$HPU$  Holographic Processing Unit

$SDK$  Software development kit

$SLAM$  Simultaneous localization and mapping
Chapter 1

Introduction

Augmented Reality is a fast growing technology that has a great opportunity to become a viral product with the support of companies like Microsoft, Google and Samsung, which are putting large amounts of money into this technology [6]. AR is on the rise and is predicted to be used by a lot of people in the near future, both in industries and for personal use. AR is an advanced technology based on computer vision to integrate augmentations with the real world. The aim is to mix objects, people and places from both the physical and virtual world together. AR aims to transform how people explore their environments and how they will communicate, create and collaborate with each other.

1.1 Codemill

This master thesis project was conducted with support from Codemill, a company working with video technology and digital product development. Codemill is located in Umeå and has experts in technical video solutions, they are working with many international companies, mostly in the media and fashion industries.

1.2 Problem Statement

Everyday, billions of people are walking around in various indoor environments doing different kind of routines. When doing these routines, whether it being to buy food for the day, lending a book at the library or trying to find the correct gate at the airport, there can be difficulties to find the way. Trouble of finding the location of interest can cause delays and confusion which can affect the people in a negative manner. A number of solutions to this problem have been tested using, e.g., wifi, bluetooth or visible light communication together with smartphones to guide the user indoors [7]. However, the use of these technologies have side effects, e.g., low location accuracy can cause problems when finding specific objects, and having the user hold and look at a smartphone may cause loss of awareness and mobility.

The aim of this master thesis project is to explore if AR Head-mounted displays (HMD) together with computer vision and Artificial Intelligence (AI) algorithms are ready to be used for helping people navigate and show information about real life objects in indoor environments via an AR HMD. Studies show that AR is a technology which is on the rise and has a great chance to be a part of many people’s lives in the near future [6]. Microsoft HoloLens
is a top candidate when it comes to AR HMDs, it is currently the only HMD that can provide a representation of real-world surfaces in the environment and build a 3D mesh out of it.

During this project, we aim to approach the following research questions:

- How to locate multiple target objects in an indoor environment and guide the user to the objects while using an AR headset?
- How can the navigation be personalized to different users?
- Is AR headsets ready to be used with practical useful solutions?

Expected outcomes of the project will be to increase the knowledge of how AR headsets can be used to guide an user to real objects in indoor environments, and how personalization can be connected to the guiding experience.

The target objects will be unique, opaque and have a surface with low reflection, the reason for these assumptions are that it is harder to find features on transparent and reflective surfaces, and hence harder to recognize and track them. (see Section 2.3). The minimum size of the objects is unclear, it will be researched and tested which size that can be tracked well at an distance of around 70cm which is a bit further than the humans average arm length.\[8\]. The target object will be a pre-learned object imported to the HoloLens from a database. The pre-learning will be made by using images or an external smartphone with vuforia scanner installed, and then uploaded to the database. AR-headsets are very new devices and there are not many studies done using them, this makes it hard to know which practices to use when it comes on how to deliver information to users. In this case we will discuss on how to guide the user, e.g., with help of an arrow pointing to the object, a line on the floor showing the path to the object, showing the object through walls, making sounds from the object and so forth.

1.3 Outline

The rest of the document is structured as follows:

1. Introduction
   This section introduces the project and contains a problem description, a problem statement which includes questions that are of interest and the expected outcomes.

2. Background
   Contains a literature review about the field of computer vision, AR and planning algorithms.

3. Methods
   Contains an investigation about computer vision and AR frameworks to find the ones best suited for the project. Then the section will continue to explain the experiments done by showing how they were conducted.

4. Results
   This section presents the results from the experiments.
5. **Discussion**  
   This section contains a discussion about the results and problems encountered during the project.

6. **Conclusion**  
   Conclusions and possible future work are outlined.
Chapter 2

Background

To solve the problem of locating lost objects with AR and computer vision, we need to investigate more about what AR and computer vision is and how it can be applied to different problems. In this chapter we will explain the detection and tracking areas of computer vision and how they can be used in the settings of AR technologies. Definitions and algorithms will be described in a simple yet detailed way.

2.1 Augmented Reality

AR is a technique used to mix virtual and real life objects together in real-time, which is then turned into a digital interface and shown on an AR device. AR devices can be mobile devices like smartphones and tablets, a personal computer or a television connected with a webcam, and during later years it can even be used with HMDs and glasses[9].

How you interact with the mixed world will vary depending on which device is being used. When using a mobile device at least one of the hands will be occupied by holding the device, a PC with a webcam will have you look at the screen, and with a HMD you will see everything right in front of your eyes with both hands free.

Common application areas of AR are games, education, military, industries, sports and navigation[10]. In Figure 2.1 an example where AR can be used in education is shown, the person in the image walks around the human body while seeing and interacting with the muscles, skeleton, etc, to learn about how everything works. Another application example used in sports is to show important lines that the players aim to reach in football as shown in Figure 2.2 the AR effects can be seen as one yellow and one blue line.

To integrate these kinds of augmentations with the real world, the software must derive real world coordinates from camera images. This is a process called image registration which uses different methods of computer vision, the data used may come from multiple photographs or from different sensors[11].
2.2 Computer Vision

Computer vision aims to build autonomous systems that could perform tasks, which the human visual system can perform. Many tasks within computer vision are related to the extraction of 3D information from time-varying 2D data such as footage from one or more cameras, and how to understand such dynamic scenes.

In the late 1960s[12], studies about computer vision began at universities that were pioneering in AI. Computer vision was meant to mimic the human visual system to provide intelligent behavior to robots; hence, it aimed to recover the three-dimensional structure of the world from images as a step to full scene understanding. A lot of the computer
algorithms that exists today were formed by studies made in the 1970s, including edge extraction, line labeling, non-polyhedral and polyhedral modeling, optical flow and motion estimation. Time went on and new techniques were discovered, such as camera calibration, photogrammetry, image segmentation and feature-based methods.

Computer vision is difficult and the human visual system are still superior to the computers in the most tasks within this area. Despite all kinds of variations in illumination, viewpoints, expressions, quantity, etc, the human can still recognize and remember faces or objects for future recognition. A vast amount of computation is required to calculate things like this in real time. However, a system with computer vision together with convolutional neural network could classify objects into fine-grained classes, such as the particular species of a bird or a turtle, which humans tend to have problems with.

The following sections will present several popular application areas and which techniques were used to build these systems.

2.2.1 Application areas

There are a wide area of applications for computer vision, which ranges from industrial machine vision systems where objects get inspected on a production line, to integration with robots that can comprehend the world around them. Examples of applications of computer vision include systems for:

- Automatic inspection, e.g., inspect production of an item to find defects.
- Assisting humans in identification tasks, e.g., a species identification system.
- Detecting events, e.g., surveillance.
- Interaction, e.g., track hands and use signs as input to a device.
- Navigation, e.g., for a car or a robot.

These different applications were built by using different computer vision tasks.

2.2.2 Common tasks

Common tasks used to build the applications in computer vision are recognition, motion analysis, scene reconstruction and image restoration.

The recognition task can be divided into how the object shall be detected, the objects can be pre-specified or learned, an individual instance of an object can be recognized, e.g., fingerprint, or scanning an image for a specific condition to find unusual regions in the image. The best algorithms for this are based on convolutional neural networks which is a method used to compute the probability that a given thing is shown in an image, and where it is shown. Several more specialized tasks based on recognition exists, but these will not be discussed in this document, for further details see.

The motion analysis can consist on: how to determine the ego-motion, which is the 3D rigid motion of the camera from an image sequence, how to track and follow the movements of a set of interest points or objects in an image sequence, or how to determine the
optical flow which for each point in the image, calculates how that point is moving relative to the image plane. The motion calculated in optical flow is a result of how the corresponding 3D points is moving in a scene and how the camera is moving relative to that scene.

Scene reconstruction is mainly how to compute a 3D model of a scene given a set of images of that scene. This can be done by using algorithms to stitch the images together into a point cloud (a set of 3D points) or to produce a 3D surface model.

The image restoration task aims to remove different kind of noise from images, such as sensor noise and motion blur. The noise removal can be done by either using filters or methods that assumes how a model of the local image structure looks like.

### 2.2.3 Techniques

Every task can be very different and solve a lot of specific problems only needed for exactly that task. However, there exists typical methods that are used in a lot of these task to build the various systems[14]. Some of these methods are:

- **Image acquisition**
  Almost every task needs to have some kind of input, and as input data there is a lot of different sensors and cameras that can be used. Examples: image sensors produce digital images, range sensors measuring the distance, radar and ultrasonic cameras. These different ways of acquire data can result into 2D images, 3D volumes or image sequences.

- **Pre-processing**
  Before extracting information from image data, it is usually good to pre-process the data, by reducing different kind of noise to decrease the amount of false information, enhancing the contrast to make certain parts clearer, or using a scale-space representation to handle different object scales in images.

- **Feature extraction**
  Features is extracted from the image data with an aim of find reliable detectable and distinguishable locations in different images. These features can consist of lines, edges, corners, points, texture, shape or motion.

- **Segmentation**
  Segmentation is used to find areas in an image to process further, those areas can exist of a set of interest points or image regions which contain objects of interest. These areas are used to find relevant information that is of interest in the image.

- **High-level processing**
  When coming this far, the input is a small set data, which can be a set of points or an image region containing some assumed objects. This input can be processed to recognize the structure of the points or what is seen within the image region, recognizing and matching the feature points to combine multiple images into a coordinate system, or estimate the application specific parameters as object pose and scale.
2.3 Object tracking

Object tracking is the problem of predicting the location of an object in the next frame as the object moves around a scene. This is a task that can vary in complexity depending on several parameters\cite{17}, such as:

- noise in images,
- complex object motion,
- nonrigid objects,
- occlusion of object,
- complex object shape,
- illumination changed in the scene,
- real-time processing requirements.

In Figure 2.3 it is shown how tracking can be used to predict where a pedestrian are even though he is not visible. How tracking works: for every frame in a video a new location is predicted based on an estimation of the objects velocity, which can be used to predict the next frame, then estimate velocity again, then predict location and so forth\cite{18}.

![Figure 2.3: Example of tracking a pedestrian with occlusion.](image)

2.3.1 Object representation

The object to be tracked can vary, it can actually be anything that is of interest for further analysis. For instance: pedestrians, faces, cats, cars, balls, everyday objects like a wallet, well it can be almost anything. All these objects can be represented by their shapes and appearances with help of representations such as:

- Points.
- Primitive geometric shapes such as rectangles, circles, etc.
- Object silhouette and contour.
- Articulated shape models such as body parts connected by joints.
- Skeletal models.

There is often a strong relationship between the object representations and the algorithm
used to track the object. The representations are usually chosen depending on the application domain, e.g., to track objects who appear very small in an image, it can be good to use a point representation, while it could be better tracking humans using silhouettes\cite{17}.

### 2.3.2 Image features

When selecting which features to track, it is important to choose the ones fitting best depending on the application domain. The most desirable property of a feature is how well it distinguishes between the object region and the background. The selection of features is closely related to the object representation, e.g., color is used as a feature for histogram representations and object edges are often used for contour-based representations. Some common features are\cite{17,19}:

- **Color**
  The perceived color of an object is influenced both by the light source and the reflective properties of the object surface. Different color spaces is used when processing images, e.g., RGB, HSV, L*u*v and L*a*b. However, all of these color spaces are sensitive to noise, and which is the most efficient is up to the application. Hereby a lot of different color spaces has ended up being used in tracking.

- **Edges**
  Object boundaries usually creates strong changes in image intensities, and edge detection is used to identify these changes. Edges are less sensitive to illumination changes compared to color features, and are usually used in algorithms to track the boundary of objects.

- **Optical flow**
  Optical flow is the pattern of apparent motion of image objects between two consecutive frames caused by the movement of an object or a camera. It is a vector field where each vector is a displacement vector showing the points from one frame to another.

- **Texture**
  Texture measures the intensity variation of a surface, which specifies properties like smoothness and regularity. These features are also less sensitive to illumination changes, similar to edge features.

Features are mostly chosen manually depending on which domain the features will be used. However, there are methods for automatic feature selection too\cite{17}. These can be categorized as either filter or wrapper methods. The filter methods tries to make a feature selection based on general features, e.g., the features should be uncorrelated. The wrapper methods select the features based on how useful they will be in a particular problem domain.

### 2.3.3 Object detection

Every tracking method requires a method to detect what to track. This is where object detection is applied. Object detection is often used to detect things in a single frame at the time, but some methods are using multiple frames in a sequence to reduce the number of false detections\cite{17}. This information is usually in the form of frame differencing, where the difference between two frames is checked to see where the pixels have changed. The tracked will then establish correspondence between detected objects across frames. Some common techniques to detect objects are:
• Point detectors
Point detectors are used in images to find points of interest, which have expressive texture close to the points. These points are used a lot with stereo images and tracking problems. The interest points aims to be points that will not change even though the illumination or viewpoint is altered.

• Background subtraction
Background subtraction is a technique used to detect moving objects in videos from static cameras. The idea is to detect deviations from the background for each incoming frame, a change in an image region from the background signifies a moving object.

• Segmentation
Segmentation tries to divide the image into multiple regions, e.g., an image containing a person standing on a beach could segment to receive regions of the sky, the beach and the person. The reason to do this is to simplify and change the representation into something that is easier and more meaningful to analyze.

• Supervised learning
Supervised learning is a machine learning task which can be used to detect objects based on learning different views of the object from a set of learning examples. Supervised learning methods take a set of learning examples and maps inputs to desired outputs via a function, e.g., an image set where each image contains either a dog or a cat, the input is the image which is mapped to the output which tells if there is a dog or a cat in the corresponding image. The images can be interesting regions of a bigger image which contains a lot of different objects.

2.4 Planning

There exist many different kinds of planning problems; and, there are several components that defines what is needed in basically all planning. These components are:

• State
Planning problems involve a state space that captures all situations that could arise. The state could, for example, represent the position and orientation of an item in space.

• Time
Planning problems involve a sequence of decisions that must be applied over time. Time may be implicit, by reflecting the fact that actions must follow in succession. The particular time is unimportant, but the proper sequence must be maintained.

• Actions
A plan generates actions that manipulate a state. When an action is performed, it must be specified how a given state changes.

• Initial and goal states
A planning problem usually involves a starting initial state, which aims to arrive at a specified goal state or any state in a set of goal states. All actions are selected in a way that tries to make this happen.
2.4. Planning

A criterion
A criterion is the desired outcome of a plan in terms of the state and actions that are executed. There are two different kinds of planning concerns based on the type of criterion:

1. Feasibility
   Find a plan that arrives at a goal state, regardless of its efficiency.
2. Optimality
   Find a feasible plan that also have optimized performance, in addition to arriving in a goal state.

If a desirable criterion can be formulated, it may be impossible to obtain a good algorithm that computes the optimal plans. In cases like this, feasible solutions are still preferable to not having any solutions at all.

A plan
A plan enforces a specific strategy or behavior on a decision maker. A plan may be simple and only specify a sequence of actions to be taken, but can also be more complicated if it is impossible to predict future states. In this case, the appropriate action must be determined from whatever information is know up to the current time.

2.4.1 Planners and Plans
A planner may be a machine or a human that constructs a plan. If the planner is a machine, it will generally be considered as a planning algorithm. A human can be a planner by producing a plan that may be executed by the machine, alternatively by designing the entire machine itself. Once a plan is decided, there are three ways to use it:

1. Execution
   A plan is usually executed by a machine, but could also be executed by a human. In our case, the planner is a machine and produces a plan, which is then interpreted and executed by the human.
2. Refinement
   A plan used for refinement determines a new plan based on the previous plan. The new plan may take more problem aspects into account or be more efficient.
3. Hierarchical Inclusion
   Package the plan as an action in a higher level plan, such as having multiple machines scanning different part of an area and then "connecting" these smaller parts together.

2.4.2 Pathfinding
Pathfinding is used to find the optimal path between two points and is a fundamental component in many important applications in the fields of video games, GPS and robotics. Generally, pathfinding consists of two main steps [21]:

1. Graph generation
   Generate some kind of graph fitting for the problem.
2. Pathfinding algorithm
   A pathfinding search algorithm aims to return the optimal path in the graph to users in an efficient manner.
2.4.2.1 Grids

A grid can be composed of either vertices/nodes or points that are connected to each other by edges to represent a graph. In most pathfinding algorithms, the performance depends on the attributes of the graphs representation. A common representation of grids are regular grids, which is one of the most well known graph type and are used in a lot of computer games and robotics. These grids describe tessellations of regular polygons such as triangles, squares or hexagons\[21\]. Examples of regular grids:

- **2D Square Grid**
  A 2D square grid is shown in Figure 2.4. The grid have 3 obstacles where red areas are tiles occupied by the obstacles and the green tiles are unoccupied. Numerous algorithms have been proposed to solve the pathfinding problem for this type of grid, such as djikstras, A* and jump point search.

- **Hexagonal Grid**
  Hexagonal grids as shown in Figure 2.5 have many of the desirable properties of square grids. They produce better paths and have smaller search time and memory complexities than square grids. However, this is only for small and medium sized grids, when used on larger maps the pathfinding algorithms require more time.

![Figure 2.4: a 2D square grid with 3 obstacles](image_url)

![Figure 2.5: a hexagon grid with 3 obstacles](image_url)
2.4.2.2 Search algorithms

Search algorithms are used to systematically search through a graph to find a solution to the problem at hand. If the graph is finite, the algorithm will visit every reachable state, which enables the algorithm to find whether a solution exist or not within finite time. Provided that the algorithms keep track of already visited states to avoid that the search will run forever.

A general search algorithm could be expressed with three kinds of states:

1. Unvisited
   An unvisited state is a state that have not been visited yet. At the start this will be every state except the starting state.

2. Dead
   A dead state is a state that has been visited, and for which every possible next state has also been visited. These states cannot contribute anything more to the search.

3. Alive
   An alive state is a state that have been visited, but have unvisited next states. All alive states will be stored in a priority queue for which a priority function must be specified. The only significant difference between many search algorithms is which kind of function that is used to sort the priority queue.

If we assume that the priority queue is a First-In-First-Out queue, the state that has been waited the longest will be chosen. To begin, the queue contains the start state. Then a while loop is executed, which is only terminated when the queue is empty. This termination will only occur when the whole graph has been explored without finding any goal states, this results in a failure to find a path. For each iteration in the while loop, the element with highest rank in the queue will be removed. If that element lies in the goal state it managed to find a path and results in a success and terminates. The algorithm will otherwise try to apply every possible action. For each next state, the algorithm must decide whether the state is being encountered for the first time. If it is unvisited, it is inserted into the queue; otherwise there will be no need to consider the state since it must already be dead or in the queue.

A lot of graph traversal algorithms exist to find the shortest path between two nodes in a graph, however, the algorithms differ in how good they fit to use at various problems. When choosing what algorithm to use, there are a number of situations one must think of such as: performance (different performance in small or big graphs), completeness (if the algorithms always find a path to the goal if such a path exists), it is partially observable (if only a subset of all possible nodes and edges within the graph is known) or if the graph changes over time. Some common search algorithms used in pathfinding are:

- Dijkstra’s algorithm
  This algorithm was originally used to find the shortest path between two nodes, but a more common variant today is to generate a shortest-path tree by finding the shortest path from the start node to all other nodes in the graph. We can see a "finished" djkstra run in Figure 2.6. In this grid each node is connected with its 8 neighbors using bidirectional edges. The costs of the edges are the same as their euclidean lengths. The gray area represents an obstacle, and nodes within the obstacle are inaccessible.
The white nodes with blue boundary represent the set of unvisited nodes. The filled circles are visited nodes, and the color in these nodes represents the distance of a node from the start node: red is lower distance while green is higher distance. The nodes is expanded almost uniformly in all directions. Djikstra’s is a special case of the more generalized A* search algorithm when the heuristics is identically 0.

- **A***
  This is an informed search algorithm that is widely used in pathfinding and graph traversal. It searches among all possible paths to the goal and to find the path with smallest cost (shortest distance travelled, shortest time, etc.). It is used a lot due to its performance and accuracy. Figure 2.7 shows a finished run of A*. The graph setup is the same as in the djikstra example above and the difference is that A* is using heuristics.

To navigate the graph the algorithm calculates the F-cost and examines the node \( n \) with the lowest F-Cost every iteration. F-cost are simply G-cost added with H-cost, \( f(n) = g(n) + h(n) \). \( g(n) \) represents the exact cost of the path from the starting point to the node \( n \). \( h(n) \) represents the heuristic estimated cost from \( n \) to the goal.
2.5 Inside-out tracking

Inside-out positional tracking is a term which refers to a method of tracking objects in three dimensional spaces. The tracking sensors (for example a Camera) is placed within the item being tracked (in our case an HMD), from which viewpoint it look out at the world around it. It uses its changing perspective on the outside world to note changes in its own position, and aims to precisely determine where it is in the space using only the sensors mounted on the device itself. The inside-out tracking can be done using markers to help the software look for already known patterns, which makes computation simpler and faster. However, marker-less inside-out tracking (a system that is robust enough to not need the markers aid) seems like the way to go if the device is supposed to be used in a less controlled environment[25]. From now on, when inside-out tracking is mentioned it is also expected to be marker-less.

A number of companies is currently working to improve the inside-out tracking, some of them have already started to sell their development kits in forms of hardware with some kind of inside out tracking implemented, including Microsoft HoloLens, and Google using a tablet with Project Tango[26]. While other companies like Oculus are still developing their solution[27] and Eonite which have not yet released their solution but is claiming to have “solved” inside-out positional tracking with the “world’s most accurate, lowest latency, lowest power consuming software to democratize inside-out positional tracking for VR and AR”[28].

There exist multiple approaches to inside-out tracking, but they all have something to do with the algorithm Simultaneous Localization And Mapping (SLAM) which is an old problem from robotics and computer vision. The problem in SLAM is to have a robot in an unknown environment which uses some sensors (Cameras, ultrasound, LiDAR, etc.) to create a map of its environment, and at the same time track its location inside a map that is continuously evolving[29][30].
Another approach is to use Dense Tracking and Mapping DTAM which is a real-time camera tracking and reconstruction system \cite{31}. This system is likely to be used by Oculus for their inside-out tracking prototype since they hired creators and refiners of all sorts and variants of SLAM and DTAM \cite{32}.

## 2.6 Personalization

Personalization is used to deliver content and functionality that matches the needs or interests of a specific user, without any effort from the user. A system using personalization could take information from an user profile to adjust various settings in the system with regards to that profile. This may be used to show particular information to different users, e.g., grant or remove access to a tool intended for certain users, or letting the system remember earlier choices an user has made that can be of interest again. There are two types of personalization \cite{33}:

- **Role-based personalization**
  Here the users are grouped together based on certain characteristics that are known in advance, e.g., only the management team in a company can view information about clients and staff due to safety concerns, while rest of the employees cannot.

- **Individualized personalization**
  Here the computer can create a model of each individual user and present different things to everyone, e.g., the computer may guess that an user likes ecologic food based on recent website searches or purchase history.

Even maps can be personalized, as seen in Google Maps where users can get different route suggestions based on if the user usually walks, ride on a bike or drives a car. It can also take current traffic in the users area into account, suggest restaurants or stores in the nearby area that you might like and so forth \cite{34}.

### 2.6.1 Context awareness

Context awareness refers to the idea that computers such as mobile devices can sense and react based on their environment. It is an term which originated from ubiquitous computing, where the thought is to let computing occur to any device, in any location and in any format. These context-aware devices may also be used to make assumptions about the user’s situation.

Context awareness have been proposed in a number of applications areas such as how to adapt interfaces, make the user interaction implicit, and building smart environments \cite{35}, e.g., a context-aware mobile phone may know that it is currently in a driving car, and that the user is the driver. The phone may then decide that the user is currently in a dangerous position to take unimportant calls and hence reject them.

These systems have to gather information to perceive a situation, then understand the context and perform some kind of behavior based on the recognized context. This can be done using various sensors, then match the sensor data to a context and trigger actions based
on this context. Context awareness is used to design innovative user interfaces, ubiquitous computing and wearable computing\[36\].

2.6.2 Proxemics

Proxemics is the study of people’s cultural perception and use of personal space to mediate their interactions with others in various situations\[37\]. However, proxemics can also be utilized in computer science to help address key challenges when designing ubiquitous computing systems and how devices can sense or capture proxemic information via five dimensions\[38\]:

- **Distance**
  The distance surrounding a person form a space, which is then divided into four zones depending on the distance from the person. These spaces starts with the closest intimate space, then connects with personal space, social space and lastly public space is furthest apart from the person. These spaces range from intimate at 0-0.5m up to 4m where public space kicks in. However, these distances can vary a lot depending on factors such as culture, age, gender and personal relationship.

- **Orientation**
  This is generally how people face toward or away from each other or a device. People can have different preferences when it comes to their orientation depending on the situation.

- **Movement**
  The movement describes changes in distance and orientation over time.

- **Identity**
  This uniquely identifies entities in the space, e.g., specific items or people such as "Max and Anna" or can distinguish different entities such as "person 1 and person 2" or categories such as "a person and a phone".

- **Location**
  Location describes the qualitative aspects of the environment where the interaction occurs. These aspects characterizes the location, e.g., knowing the difference between the home and a store. It also describes fixed and semifixed features such as the layout of the room and the position of furniture, and provides meta-information such as social practices and context of use.

All these dimensions can also be applied from a human against a device to exemplify how we can use that knowledge to design interaction techniques. Context awareness is used in proxemics to provide the devices with knowledge about the situation around them, so that they can infer where they are and then act accordingly.

2.6.3 Personalizing with proxemic interactions

Some of the core challenges when designing ubiquitous computing relevant to proxemic interactions can be seen as six different challenges:
1. How to reveal interaction possibilities
   How to integrate technology into the everyday environment such that it "dissapears" when we do not need it, and to fluently grab our attention when needed. The challenge here is to design the interaction to all the way from when the technology is in the background of a person’s attention, the transition, and the foreground.

2. Directing actions
   How to decide whether an action is a directive to the system or just part of a person’s everyday actions. These actions can be gestures, movements and voice commands.

3. Establishing connections
   The devices must somehow control how they connect to each other seamlessly while still keeping the transfer of content private and secure.

4. Providing feedback
   How to provide feedback about the application’s current status, its interpretation of user input, or its errors. It is important here to have the knowledge that people’s attention might switch between the technologies background and foreground.

5. Avoiding and correcting mistakes
   How to correct errors or mistakes. It can be easy to misinterpret the different sensor data acquired from the devices and this need to be avoided somehow.

6. Managing privacy and security
   How can the system protect sensitive private information without getting in the way of challenges 1 to 5.

These challenges can be reduced to more simplified states using the proxemic theories. Although these theories describe the relations between people, we can use them as an approximation with devices. If a system can sense the presence and approach of people, it can use that information to reveal possible interactions, e.g., a simple approach where a light turns on when a person walks by, or a more advanced where a screen is displaying various information depending on its distance to the person. This distance could also be used to limit the interaction with the object if the user is too far away. Also other measures such as attention and orientation can be used to detect the user’s current focus, this can be to let the system automatically pause a video when the user looks away from the screen, and to start it again when facing the screen.

When it comes to provide feedback to the user, the system can adjust output as needed via lights, sounds, speech or by moving objects. This assumed that the system knows the physical orientation and position, e.g., when a person is facing away from a screen, the system uses sound to give the user notifications, and while facing the screen it can show visual output. Example of this used in AR can be a virtual person waving to the user to indicate the user to come closer, when moving closer the virtual person would greet you and then giving you options to let it dance, sing, give you the news, follow you to another location etc. Then to mimic the real world, speech can be used as input to tell the virtual person what to do next.

When correcting possible mistakes made by the system the actions can be inverted, such that the system reverted to the prior state, e.g., The example with the virtual person greeting an user, if the user do not want to greet the virtual person he could just step back or
face away to make the virtual person stop the greeting. Mistakes can also be corrected by explicitly undoing the actions, e.g., telling the virtual person to stop talking with speech or hand gestures. However, when handling important actions with a high impact such as deletion of something, more strict proxemics should be used, e.g., being very close to the object of deletion could pop up a menu making it possible to delete the object.

When managing the privacy and security many different approaches can be taken. This could be to recognize a person’s face, body or gaze and only show information on a screen when that user is looking at it, and by considering the identity dimension hiding personal information if someone else is looking at the screen. The location could also be used to adjust the security settings in different environments, by having lower security while at home and higher security when at a public location.
Chapter 3

Methods and Tools

This chapter will present how the project was planned, why specific tools were used, information about the selected tools, important design considerations during the development, and how the user study was done.

3.1 Planning the project

Early in the process a rough project plan, including a gantt chart, was produced to easier keep track of the project. The gantt chart was a rough guess with many of the task overlapping each other in time.

A project diary was used and updated weekly with answer to the following questions: What was the weekly plan? What has been done? Which problems has been encountered? Which experiences, negative/positive have been gained? Which people have been communicated with regarding the project? and are you in phase with the project plan? If not, which measures will be taken? With the intention of discovering potential issues, and in that case update the plan as early on as possible.

3.1.1 The preparatory Phase

To get a better knowledge of how to approach the object localization problem literature studies were done in both in the areas of computer vision and AR. Web pages, forums and books were the main sources of information, and online video lectures and scientific articles also contributed to the study. These literature studies provided understanding of which software and hardware that could be used to build the system, many of the algorithms used were inspired by how they are applied to games.

3.1.2 Deciding devices and framework

To simplify the implementation of the system, it was decided to take help of different devices and frameworks that would help with the work as much as possible. To decide which device and frameworks to use, a research about the differences were done and the ones most fit for the project were selected.
3.1.2.1 Important features

To be useful in the project, there were several requirements the tools had to fulfill. First of all, they had to be compatible with each other to avoid unnecessary integration problems. Since the project was built around using an AR HMD, and the HoloLens had the important feature inside-out tracking, a lot of the other tools were chosen with HoloLens in mind. Second, since there was limited time it was important that there existed some kind of tutorials or documentation on how to get started with all different parts early on. These features were inspected early on in the project to get knowledge if the system seemed to be implementable or not.

3.1.2.2 Proposed devices

There were only a few devices to choose between, since the area of tracking the location in an indoor environment is very new. The top two contenders were:

- Microsoft HoloLens
  Microsoft HoloLens is an HMD unit which can detect their position relative to the world around them and add virtual objects to the environment.

- Tango
  Tango developed by Google is more of an AR platform than device, however it uses smartphones as devices. It uses computer vision to enable mobile devices to also detect their position relative to the world around them and to add virtual objects to the environment.

HoloLens is an AR HMD which allows the user to walk around in the environment having free hands and a display straight in front of the eyes. Tango is made for handheld devices, making it slightly clumsier to use than the HoloLens when interacting with common everyday objects at the same time as holding the phone. To provide as much freedom in movement to the user as possible, the HoloLens was chosen as the device to work with.

3.1.2.3 Proposed frameworks

First of all, Unity was chosen to work with since they had easy to start guides, and support for HoloLens.

There exists a number of Software Development kits (SDK) for AR. Some of the most popular ones are Vuforia, ARToolKit, Catchoom, CraftAR and Blippar. After a comparison between these AR SDKs, Vuforia got selected to be used for tracking. It was the only AR SDK that had easy to start guides to track 3d objects, test samples for the HoloLens and were official partners with Unity.

3.2 Final device, frameworks and tools

3.2.1 HoloLens

Microsoft HoloLens is the first self-contained, holographic computer, enabling people to engage with digital content and interact with holograms in the world around them. HoloLens is an AR headset, also known as a HMD (head-mounted display). The AR headset aims to
merge virtual objects into the real world, and allow users to interact with those objects via
different inputs such as gaze, gesture and voice.

3.2.1.1 Hardware
The HoloLens features hardware include an inertial measure unit, four "environment understand-
ing" sensors, a depth camera, a video camera, a microphone, an ambient light sensor, and more. See table 3.1 for all of the Microsoft HoloLens specifications.

The most unique piece of hardware within the HoloLens is the Microsoft Holographic Pro-
cessing Unit (HPU), which is a custom-made coprocessor specifically manufactured for the
HoloLens. The HPU processes and integrates data from the sensors, handling tasks such as
spatial mapping, gesture recognition and voice/speech recognition.

<table>
<thead>
<tr>
<th>Table 3.1: Microsoft HoloLens specifications[5]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
</tr>
<tr>
<td>Inertial measurement unit, (accelerometer,gyroscope and magnetometer),</td>
</tr>
<tr>
<td>4x &quot;environment understanding&quot; cameras,</td>
</tr>
<tr>
<td>1x Depth camera with 120°x120° angle of view,</td>
</tr>
<tr>
<td>4x microphones,</td>
</tr>
<tr>
<td>Ambient light sensor.</td>
</tr>
<tr>
<td><strong>Controller input</strong></td>
</tr>
<tr>
<td>Gestural commands</td>
</tr>
<tr>
<td>Clicker device</td>
</tr>
<tr>
<td>Gaze</td>
</tr>
<tr>
<td>Voice input</td>
</tr>
<tr>
<td><strong>Display</strong></td>
</tr>
<tr>
<td>See-through holographic lenses (waveguides)</td>
</tr>
<tr>
<td>2x HD 16:9 light engines</td>
</tr>
<tr>
<td>Automatic pupillary distance calibration</td>
</tr>
<tr>
<td>2.3M total light points holographic resolution, 2.5k light points per radian</td>
</tr>
<tr>
<td><strong>Sound</strong></td>
</tr>
<tr>
<td>Spatial sound technology</td>
</tr>
<tr>
<td><strong>Operating system</strong></td>
</tr>
<tr>
<td>Windows Holographic</td>
</tr>
<tr>
<td><strong>Platform</strong></td>
</tr>
<tr>
<td>Windows 10</td>
</tr>
<tr>
<td><strong>CPU</strong></td>
</tr>
<tr>
<td>Intel 32-bit (1GHz)</td>
</tr>
<tr>
<td><strong>Memory</strong></td>
</tr>
<tr>
<td>2 GB RAM</td>
</tr>
<tr>
<td>1 GB HPU RAM</td>
</tr>
<tr>
<td><strong>Storage</strong></td>
</tr>
<tr>
<td>64 GB (flash memory)</td>
</tr>
<tr>
<td><strong>Connectivity</strong></td>
</tr>
<tr>
<td>Wi-Fi 802.11ac, Bluetooth 4.1 LE, Micro-USB 2.0</td>
</tr>
</tbody>
</table>

3.2.1.2 Inside-out tracking
An existing method to do the inside-out tracking is implemented in Microsoft HoloLens. Microsoft has not shared the exact algorithm, but it is believed to be based on SLAM and the Kinect Fusion algorithm[39] since the creator of both Microsoft’s Kinect and HoloLens is the same person, Alex Kipman[40]. Microsoft has a method called spatial mapping which is used to scan the surroundings and creates a 3D model which is used to localize the user in the environment, this is further explained in Section 3.2.1.3. Microsoft has shared a very simple explanation on how the HoloLens tracking works:
"Early map makers would start by noting prominent physical or geological features, and then fill in details by placing items relative to these markers. As the cartographer explore more and more of an area the accuracy of a map would improve. HoloLens maps a room in a similar way, tracking works to recognize surfaces and features of a room. When holograms are world locked, that is placed relative to real objects. Then the objects are anchored relative to these surfaces. As the user moves around the room, the details of a room continue to be mapped so that the placement of holograms relative to these surfaces becomes more accurate and solid. Since tracking is more accurate at closer distances, the image the HoloLens has of a room continues to increase in accuracy over time as the user moves and views objects from various angles. You can even have multiple anchors in a world locked scenario, with each anchor having its own coordinate system. The holographic APIs helps you translate coordinates between the different coordinate systems. As the HoloLens continues to track and get better information about its environment the translation between these coordinate systems will get more precise.

3.2.1.3 Spatial mapping

Spatial mapping is a process of mapping real-world surfaces into the virtual world. It is a feature that is used by the HoloLens to scan its surroundings with help of sensors, these scans provide a 3D mesh of the environment which is saved on the device. The spatial mapping is designed for indoor environments and can handle noise such as people moving through a room, or doors opening and closing. The main areas where spatial mapping is usable is navigation, occlusion, physics, placement and visualization. The spatial mapping can be used as a simulated surface of the world around, e.g., an virtual ball can be dropped on a table and start to roll on the table until it falls down onto the floor. It can also be used to occlude objects, e.g., when having a wall between you and the virtual object, the spatial mapping of the wall can be used to "hide" the virtual object just as you cannot see through walls in real life.

The HoloLens maps its world by using so called Surfaces, these Surfaces are orientated in the world in a way that is convenient to the system and there is no guarantee that these Surfaces are arranged in any particular orientation or to intersect a given world space, in a good way.

3.2.2 Vuforia

Vuforia was used to recognize and place the virtual representation of the recognized objects in the virtual environment corresponding to the real world position.

Vuforia supports the HoloLens in Vuforia Unity Extension 6.1 and newer, this extension was used together with some "How to" guides on Vuforia developer library to set up the basic structure of the recognition part within Unity.

To be able to recognize the target in the system later on, it needs a target source (a feature scan or image) which is then inserted into vuforia target database. Depending on the target type it can be good to use either feature scans or planar images. Image based targets such as publications, packaging and flat surfaces is better tracked with planar images. While Object targets such as toys, non-flat products and those who have complex geometries fit
better to have a feature scan. The planar images can be images of the various sides of a package, and the user need to specify the real size of the object manually. While the feature scan need the user to download an application in Android Play Store to scan the target, print a paper with a special pattern to use as reference to the target object. When this scan is complete the scanner produces an Object Data file.

The target source is then uploaded to the "Vuforia Target Manager" database. The user is able to store up to 20 different targets to use in a single device. When all the objects to be used are added to this database, the database can be downloaded. The database will be downloaded and imported into Unity as an unity package. Within Unity, each target within the database need to be added to an virtual object.

### 3.2.3 HoloToolKit

HoloToolKit is an open source collection of scripts and components intended to accelerate the development of holographic applications targeting Windows Holographic platform. The toolkit is used in many parts of the system, e.g., gaze, gesture, spatial mapping and spatial understanding.

The use of gaze and gesture is part of the input module. The gaze can perform operations on the position and orientation to stabilize the user’s gaze; moreover, it is also used to create a cursor via raycasting a line that follows the center of the user’s view. Gestures are used to be able to click with the cursor, the only gesture used is the "tap" gesture as shown in Figure 3.1. There are also another gesture called bloom, this gesture is however reserved by the operative system to exit the current running application.

![Figure 3.1: How to do the tap-gesture](image)

The spatial mapping is used to allow visualizing spatial mapping data in the HoloLens or in the Unity Editor. A part of this is to add and update spatial mapping data for all surfaces discovered by the surface observer running on the HoloLens, i.e. to scan the environment.
and create a mesh from it.

The spatial understanding is used to understand more about the environment. In this project it make decisions about what floor is, and where it is in the virtual environment. To do this, the mesh from the spatial mapping is processed using heuristics, e.g., the largest and lowest horizontal surface with greater surface area than $1m^2$ is considered to be the floor.

3.3 Workflow

3.3.1 Working with the system

When developing the system, test-runs were made both in real life using the AR HMD, and simulated on the computer in Unity, both with their positives and negatives.

When compiling to the AR HMD, the whole feeling of the system could be easier comprehended. Here it was easier to think of proxemics and its implications on the user, how far away was the object, what could be seen at certain times, how to make the user move, how to interact with virtual and real objects together, etc.

All object recognition work was done using the AR HMD to actually recognize the real object, e.g., to learn at which distance to track the actual objects, and how to visualize the real object in the virtual world to give the user feedback of when the object is recognized.

The system can also be simulated in Unity with certain limitations, such as not being able to recognize objects. This was used together with spatial understanding to simulate a scan of an already pre-scanned mesh from the HoloLens. To get the pre-scanned mesh from the HoloLens into the computer, one must connect to the HoloLens web-interface when the computer and HoloLens are connected to the same wifi, and download the mesh .obj file from there.

Figure 3.2 shows a representation of a the spatial map generated by the HoloLens. With help of HoloToolKit, this mesh can be used as a representation of the real world when running the system. A simulated scan of a pre-scanned environment can be seen in Figure 3.3.

It was faster to simulate the system in the Unity editor while working on certain parts, rather than walking around and scan the entire environment using the AR HMD every time the system was compiled.

3.3.2 Design

When designing the system, there were always coming up new problems that had to be solved. The system prototype was re-designed a number of times to make it easier for user’s to understand how to use it and get better feedback.

Proxemic theories was used to design the system, e.g., distance is used as an indication and reveal possible interactions, design choices let the user’s focus fade from the background into the foreground, how the user’s movement can alter the experience, correcting possible mistakes and how to design feedback to show different information depending on
physical orientation and position.

Personalization was also in mind when designing the system, e.g., people in a store might want to walk with or without a shopping cart, they might have different preferences when it comes to how open the area they walk on are, etc. The system have changeable variables that can be set to have a minimum area where it will be possible to walk, e.g., if the shortest path are a path where an user together with a shopping cart will not get through, the user will be directed to the location in a wider path where both the user and the shopping cart will fit. The object penalty (see Section 4.2.2) can also be raised to make the user avoid confined spaces more or less depending on the use case.

These design choices were made continuously in the project, with many different people testing the system during the process. By taking note of how these people reacted and things they said, the design was altered to give future users a better user experience.

3.4 User tests

There was two different kind of user tests made, one in an office where I worked, and another in a food store.

3.4.1 Office

The people who tested the system in the office were testing different versions of the system. This was a way to get feedback during the development, with help of comments and reactions of the user’s. These tests were not planned beforehand like the test in the store, this was mainly used to let people interested in AR have something to try. Then, their comments and reactions were utilized to improve the system.
3.4 User tests

3.4.2 Store

The user study was used to see how different user’s are experiencing the system, and if they are more efficient while using it. The scenario is to let user’s navigate through a food store to find and buy various items while using the system, this was timed and compared to how long time it takes to find other items at a similar distance without the system. The user’s were asked to walk at a normal pace. After the navigation test, they were asked to answer a questionnaire about their experience with the test.

There was two groups of people $g_1$ and $g_2$; and, two sets of items $s_1$ and $s_2$ with three different items in each set. $g_1$ started to find the items in $s_1$ without the system, when all the items are found they searched for the items in $s_2$ with the system. Later, $g_2$ will do the same but starts with $s_2$ and moves on to $s_1$. Each user was doing the test alone, so that no sneak-peaks could be made by another user.

Three of the users have never been at the store before, while three others are familiar with the store. This data will be used to compare how the experience with the system will be for user’s regularly navigating the test environment, with how it is for user’s not familiar with the environment.

To set up the test, a food store will be scanned and the position of the items to be found will be saved. Only half of the store will be used as a test-area to keep it as simple as possible, this area will be around 13x29 meters. After the scan, the system in whole, and an explanation of how to use everything will be presented to the user’s. To let the user’s become a bit familiar with the tap gesture, they will have to start their search by tapping the guide button in the menu, which will be at their start position. It will be explained on how to orient themselves, and how to buy items. When this short tutorial is done, the system test will start.
Chapter 3. Methods and Tools

3.4. User tests

When the test begins, the user’s will be presented with three images of items they have to find. The stopwatch starts and they will begin the search. The time it takes for an user to tap and buy the items will also be saved, to see if this is something that slows down the user’s. When the user’s is searching for the items, both with and without the system, I will walk behind them and tell them what they are searching for in case they forgot.
Chapter 4

Implementation

This chapter will explain the flow of the system, the consideration of proxemic interactions, and the algorithms used to implement the system.

4.1 The flow of the system

The flow of the system is explained in this section to easier understand where the algorithms later in this chapter fit in, and to give the reader a wider overview to understand how everything is connected.

4.1.1 Setting up the navigation

A number of steps is required to setup the system before the navigation can begin. In Figure 4.1 a flowchart of the initiation is shown from the user perspective, it contains several steps which will be explained below:

1. Start the application
   Go to the preferred starting position in your environment and start the application from the HoloLens menu.

2. Scan
   When the application have started, a text will show up telling the user to walk around in the environment and scan. When a minimum area to work with have been scanned, the text will turn yellow and the user will be able to air-tap to finish the scan.

3. Finished scan
   When the scan is finished, a menu comes up in front of the user letting it to continue the scan to make a larger area, or to generate a grid. When continuing with the scan, the system will go back to the scan step again.

4. Generate grid
   This step starts with calculating where all floor positions are in the scanned area with help of HoloToolKit and spatial understanding. The floor positions furthest away from each other are used to generate a grid between these points. The grid is a set of nodes, where each node at first is set to unwalkable, and then set to walkable if a floor position exists within that node’s area. The valid floor nodes are then given values
4.1. The flow of the system

depending on how close they are unwalkable nodes, this information will be used as heuristics in the pathfinding. The grid will remember the last seen positions of the objects to be tracked later. This will be used with the A* pathfinding algorithm to find the shortest path (depending on heuristics) between the AR HMD and the object to be found.

5. Hide mesh
The visible mesh can be toggled on and of. When scanning the environment it can be good to see what has been scanned and not, this can however make the system a bit laggy compared to when not showing the mesh. Hence, to make the system run smoother and look more appealing for the user, we turn off the visible mesh when the scan and grid generation is done.

6. Place objects
The Vuforia framework is used to recognize the objects and their virtual positions in the real world. The position is then converted to the corresponding node position in the grid. When an object is recognized, it will get an outline around itself.

![Flowchart](image)

Figure 4.1: Flowchart of how the system is initiated

4.1.2 Using the navigation

When the scan and grid generation is finished the system is ready to guide the user around in the environment, in Figure 4.2 we can see a flowchart of this. The steps of this flowchart
are:

1. Look at the menu
   The menu is located in front of the user when the scan is completed.

2. Decide objects to find
   When deciding which objects to find, the user can select the objects directly by gazing at the corresponding button in the menu; so that, the cursor in the center of the view touches a button, and then air-tapping. This will place the object in a first in first out queue. The user can also press the suggest route button where the system will compare the distance between all objects and the current position of the user and save the shortest available route.

3. No visible line in the view
   If no line is visible in front of the user, a red arrow will indicate that the user’s orientation is off and need to re-orientate and face towards the position the arrow is pointing at. The red arrow points against the center position of the line and will only be shown if the user is facing the wrong direction.

4. Follow the line
   A blue line will be visible and if the user follows that line it will eventually fade to green and become shorter when the object of interest is close by. This line is created with use of the node points found in the path. The line is then using the node positions from about 0.5 to 3m away from the user, and then using a Bézier curve to make the line follow the path smoothly.

5. Look at object
   When the object is found, the user can pick it up and look at it. The system will then recognize the object (given it is a pre-learned object) with help of the Vuforia framework. Now there is a blueish outline following the object as long as it stays within arm-length distance and in view of the HoloLens camera.

6. Air-tap the object
   The cursor in the center of the view will appear on the object when the user gaze towards it. Now the user can air-tap the object to mark it found. When found the object outline turns green as an indication that the user is finished with that object. At the same time, the menu is updated and removes the item from the first position in its route queue.

7. Objects are found
   If not all objects in the queue are found the user can continue the route to follow the line to the next object. If all objects are found, the user can either choose to search for new objects or to be finished.
4.2 Planning

4.2.1 Spatial Understanding

The spatial understanding was used to find the floor positions of the scanned environment’s mesh. Figure 4.3 represents a processed 10x10m room and how the found floor positions is distributed. The blue squares represent the floor positions, while the green represents the room mesh. To make it easier to comprehend, the number of visible floor positions in this
image was set to 500 of about 10 000 found positions. Then, the implemented system uses all 10 000 to gain higher precision when setting the walkable nodes in the grid. Figure 4.3 also shows how obstacles such as the office island and the pillar above it is ignored. The minimum size for each floor area found was set to 80x80cm with the intention of the space being able to contain at least one human standing in that area.

Figure 4.3: An example of a processed 10x10m room, the green represents the mesh, and blue squares represent the floor positions of the room

4.2.2 Graph representation

The finished graph is represented by a grid of nodes where the nodes contains information about if they are walkable or not, and a movement penalty weight which is decided on how close the node is to an obstacle.

A grid representation of the room shown in Figure 4.3 can be seen in Figure 4.4 where the red areas are unwalkable nodes, and the gray area being walkable nodes. The varying colors in the gray area represents how close the corresponding nodes are obstacles, light gray colored nodes are far away from obstacles while darker ones are nearer.

First of all, the grid size is decided by using the spatial understanding to find the posi-
tions furthest apart from each other, that are two new points with the largest and smallest value of their world position. With this data, a new grid is made with all nodes being unwalkable. The grid gets updated by taking all positions from the spatial mapping and raycasting a line from them onto the grid, telling the node hit by a raycast to be a walkable node.

The movement penalty weights are then added to the unwalkable nodes and smoothed out to nearby nodes when using a box blur on the grid.

![Figure 4.4: A grid representation of a 10x10m room](image)

### 4.2.3 Algorithms

#### 4.2.3.1 A*

A* is used to find the path between two positions in a grid by using a heuristic estimate of the cost to get to the goal from the starting position. An example of a calculated A* path can be seen in Figure 4.5.

The pseudocode can be shown in Algorithm 1. The first two lists are created, the OPEN and the CLOSED. The OPEN list stores all the nodes with an already calculated F-cost

![Algorithm 1](image)
OPEN //the set of nodes to be evaluated
CLOSED //the set of nodes already evaluated
add the start node to OPEN

loop
    current = node in OPEN with the lowest f_cost
    remove current from OPEN
    add current to CLOSED

    if current is the target node //path has been found
        return

    foreach neighbour of the current node
        if neighbour is not traversable or neighbour is in CLOSED
            skip to the next neighbour

        if new path to neighbour is shorter OR neighbour is not in OPEN
            set f_cost of neighbour
            set parent of neighbour to current
            if neighbour is not in OPEN
                add neighbour to OPEN

Algorithm 1: A* pseudocode[46].

and are to be evaluated. The CLOSED list is composed by the set of nodes which already have been evaluated. Then the starting node is added to the OPEN list. A loop is entered, and within the loop a temporary variable current is created, which are the node from the OPEN list with the lowest F-cost. Then, the current node is removed from the OPEN list and added to the CLOSED list. If the current node is the target node we assume to have found the path and exits the loop.

If not, we will go through all neighbours of the current node. If the neighbour is not traversable such as there being an obstacle in the way, or if it is the CLOSED list, we will skip this neighbour. Otherwise, if the new path to the neighbour is shorter than the old path, that is when the neighbour already is in the OPEN list but the newly discovered path is shorter than that node’s shortest path so it has to be updated with the new path. Or if the neighbor is not in the OPEN list we will set the F-cost of the neighbour by calculating the G-cost and H-cost (see Section 2.4.2.2). Then, we set the neighbours parent to the current node to be able to retrace the path back to the start when we get to the end node. If the neighbour is not in the OPEN list, the neighbour is added to the OPEN list.

The loop will continue until the current node is the target node and then exit. After the pseudo code, a method traversing through the parents from the target node until we find the starting node is run to get our path.

The worst possible case to find the path would be to examine the entire graph, that is, to visit every node and check every neighbour. This will result in a complexity of $O(|V| + |E|)$ where $|V|$ is the nodes and $|E|$ their neighbours.
4.2.3.2 Blur

Blur was used on the weights of the grid to make the path more smooth. To make this the grid was first blurred horizontally, and then blurred vertically\[47\]. How this was implemented is represented in Figure 4.6, the steps used in the figure are:

1. Starting with an original grid with weights, we add a kernel (a smaller grid) to the top left node. The kernel is represented as red and green squares, with the center of the kernel in red. All values in the kernel are added together and put into a new grid containing the horizontal blurs, the position of that value is put in the same position of the red square but in the new grid.

2. For the rest of the row, the kernel is shifted one index at a time, the value added to the new grid is then added with the new kernel value, and the value who left the kernel is subtracted from that value. In Figure 4.6 it becomes 3+3-1=5. This goes on until the new grids first row is finished.

3. The kernel moves down to the next row and repeats the procedure of step 2. When all rows are finished, the whole procedure restarts but with the new grid, and the kernel

Figure 4.5: Example of A* run on a 10x10m room with the node size 15cm. The blue dot represents the start position
moving vertically instead.

4. When both the horizontal and vertical procedures have finished, the only step left is to divide all values with 9 (The number of possible neighbors for a node). The new grid is now a blurred version of the original.

![Figure 4.6: Horizontal and vertical blur](image)

### 4.2.3.3 Bézier path

A Bézier path is used to interpolate points in the path found via the A* algorithm. A tutorial for this can be read at the site devmag [48]. A Bézier path is a set of Bézier curves stitched together, these curves have 4 control points each and are called cubic curves. Figure 4.7 shows a cubic Bézier curve with four control points, each marked as a red dot. The formula for cubic Bézier curves is:

\[
[x, y] = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t)t^2 P_2 + t^3 P_3
\] (4.1)

Where \( P_0, P_1, P_2 \) and \( P_3 \) are the control points. \( t \) is a value ranging between 0 and 1. The value 0 corresponds to the start of the curve while the value 1 is the endpoint of the curve. Values between 0 and 1 corresponds to other points on the curve. \([x, y]\) is the vector representing the curve.
4.2. Planning

Figure 4.7: A cubic Bézier curve
Chapter 5

Results

This chapter will present the final prototype and the results gained from the user test. The template of the questionnaire is presented as an appendix, see Appendix A.

5.1 Prototype

The prototype is able to navigate and find multiple objects in an environment if the environment has been scanned beforehand. Figure 5.1 shows how it looks when starting the system, the user will be asked to scan the preferred environment. When scanned a large enough area, the text will turn yellow (Figure 5.2), and the user can tap to finish the scan. When a scan is finished, a menu will appear in front of the user (Figure 5.3). The menu has some options, to show some of the positions found on the floor, to continue the scan, to generate a grid, and to toggle the scanned mesh on and off. The menu also contains a "User Search Menu" tab (Figure 5.4), which is used to choose which of the available items to find.

The path will be visible to the user via a blue line (Figure 5.5), and will slowly become shorter and fade to green (Figure 5.6) when getting closer to the location. If the user is facing the wrong direction relative to the line, a red arrow will show up to direct the user to look another way (Figure 5.7). The items to be found have a blue outline (Figure 5.5) at the beginning, which will become green (Figure 5.6) as an indication that the item is found.

5.2 User study

This section will present the result from the user study, in terms of time, and how the test users reacted when using the system.

5.2.1 Time

Six people tried the system in a food store, the people was divided into two groups with three people in each group. Group one searched for item set one without a HMD (Figure 5.1), and for item set two with a HMD (Figure 5.3). Group two searched for item set two without a HMD (Figure 5.2), and item set one with a HMD (Figure 5.4).
5.2. User study

Chapter 5. Results

Figure 5.1: The view when starting the system.

Figure 5.2: The view when a scan is big enough to start processing.

Figure 5.3: The menu, showing the available options in the setup menu.
Chapter 5. Results

5.2. User study

Figure 5.4: The menu, showing the available options in the user menu.

Figure 5.5: The blue line, representing the path.

Figure 5.6: The green line, representing the path when close to the location.
5.2. User study

Figure 5.7: The red arrow, provides feedback to the user when facing the wrong direction.

Figure 5.8: The item to be found (The dark color within the square is not visible in the AR HMD).

Figure 5.9: The item marked as found (The dark color within the square is not visible in the AR HMD).
Table 5.1: User test Group 1, Set 1, no HMD. Time presented in seconds = s.

<table>
<thead>
<tr>
<th>User</th>
<th>Found item 1</th>
<th>Found item 2</th>
<th>Found item 3</th>
<th>Total time</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>16</td>
<td>43</td>
<td>76</td>
<td>76</td>
</tr>
<tr>
<td>User 2</td>
<td>15</td>
<td>80</td>
<td>102</td>
<td>102</td>
</tr>
<tr>
<td>User 3</td>
<td>47</td>
<td>104</td>
<td>243</td>
<td>243</td>
</tr>
</tbody>
</table>

Table 5.2: User test Group 2, Set 2, no HMD. Time presented in seconds = s.

<table>
<thead>
<tr>
<th>User</th>
<th>Found item 1</th>
<th>Found item 2</th>
<th>Found item 3</th>
<th>Total time</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>44</td>
<td>241</td>
<td>303</td>
<td>303</td>
</tr>
<tr>
<td>User 2</td>
<td>112</td>
<td>216</td>
<td>297</td>
<td>297</td>
</tr>
<tr>
<td>User 3</td>
<td>35</td>
<td>60</td>
<td>92</td>
<td>92</td>
</tr>
</tbody>
</table>

Table 5.3: User test Group 1, Set 2, with HMD. Time presented in seconds = s.

<table>
<thead>
<tr>
<th>User</th>
<th>Found item 1</th>
<th>Clicked item 1</th>
<th>Found item 2</th>
<th>Clicked item 2</th>
<th>Found item 3</th>
<th>Clicked item 3</th>
<th>Total time</th>
<th>Total time excluded click time</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>34</td>
<td>85</td>
<td>102</td>
<td>116</td>
<td>128</td>
<td>139</td>
<td>139</td>
<td>63</td>
</tr>
<tr>
<td>User 2</td>
<td>22</td>
<td>40</td>
<td>57</td>
<td>98</td>
<td>109</td>
<td>118</td>
<td>118</td>
<td>50</td>
</tr>
<tr>
<td>User 3</td>
<td>24</td>
<td>91</td>
<td>106</td>
<td>120</td>
<td>135</td>
<td>145</td>
<td>145</td>
<td>54</td>
</tr>
</tbody>
</table>

A one-sample t-test was made to determine whether the mean time it took to find a number of items in a food store is significantly lower while using an AR HMD to guide you, compared to not using it. The hypotheses was set to:

\( H_0 : \mu = \mu_0 \), the mean time it took with a headset (\( \mu \)) equals the mean time it took without a headset (\( \mu_0 \)).

\( H_1 : \mu < \mu_0 \), the mean time it took with a headset (\( \mu \)) is less than the mean time it took without a headset (\( \mu_0 \)).

The values used in the t-test was: mean time without the headset = 185 seconds, mean time with headset = 101 seconds, standard deviation when using the headset = 35.4 seconds, and a sample size = 6 people. The null hypothesis is rejected at a 1% significance level, and the t-test resulted in p-value = 0.001011 < 0.01; hence, the null hypothesis is rejected.
Table 5.4: User test Group 2, Set 1, with HMD. Time presented in seconds = s.

<table>
<thead>
<tr>
<th></th>
<th>User 1</th>
<th>User 2</th>
<th>User 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Found item 1</td>
<td>13</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Clicked item 1</td>
<td>25</td>
<td>17</td>
<td>9</td>
</tr>
<tr>
<td>Found item 2</td>
<td>56</td>
<td>35</td>
<td>25</td>
</tr>
<tr>
<td>Clicked item 2</td>
<td>65</td>
<td>44</td>
<td>28</td>
</tr>
<tr>
<td>Found item 3</td>
<td>80</td>
<td>56</td>
<td>46</td>
</tr>
<tr>
<td>Clicked item 3</td>
<td>87</td>
<td>70</td>
<td>49</td>
</tr>
<tr>
<td>Total time</td>
<td>87</td>
<td>70</td>
<td>49</td>
</tr>
<tr>
<td>Total time excluded click time</td>
<td>59</td>
<td>34</td>
<td>39</td>
</tr>
</tbody>
</table>

Table 5.5: Overview information, how easy test user’s thought different parts of the system was. Data is presented as number = n.

<table>
<thead>
<tr>
<th></th>
<th>Very hard</th>
<th>Hard</th>
<th>Neither easy nor hard</th>
<th>Easy</th>
<th>Very easy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to learn</td>
<td>n</td>
<td>n</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Follow the path</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Buy item</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Move on to next item</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

This means that the mean time to find a number of items in a food store is significantly lower when using an AR HMD to guide you, compared to when not using one.

5.2.2 Questionnaire

A questionnaire was made with the same six users that was testing the system. The age of the user’s were between 20 and 29, half of the testers was familiar with the store since before.

Five of the users were positive about the usefulness of the system, and only one thought it was neither good nor bad. Two users were indifferent to whether the system was intuitive or not, while four users were positive. When asked about how intuitive it felt to have the blue path fade to green when it came closer to the location, two thought it was not intuitive, one was indifferent, and three thought it was intuitive.

A red arrow slowly came into view to show where to look when facing the wrong direction, five of the users thought it felt good to be guided with the arrow while only one thought it was annoying. The amount of help the color scheme provided to the user’s was scattered, with two users not caring about the colors at all, one was indifferent, and three user’s thought it was helping.

Table 5.5 shows how easy the users thought different part of the system was. The result is leaning to be an easy system to use, with only one user finding one of the tasks hard.

When asking about the interest for a similar system in the future, with a smaller and more affordable headset, all six test user’s was very positive that it would be something they would use.
Comments

In the questionnaire, a specific question was asked to gain comments on what users missed with the system. The question was as follows:

Is there anything you would like to add to the application, which you thought was missed? Please feel free to come up with new ideas and changes.

The following comments were given:

- "I was a bit too short for the AR headset, which I think made me feel a bit negative about it. The blue/green line did not always show up for me while walking in the store. It was like; popping up and disappeared over and over. Though I had an advantage cause I know the store since before. Overall the AR headset and this "guide thing" could be very helpful in my work with nutrition and helping patients find special foodstuff in the store."

- "Maybe show the red arrow to guide me to the next blue path immediately after an item is bought"

- "The arrow was hard to find, maybe place it so it always is visible wherever you look"

- "Fov needs to be bigger"

- ""Clicking" using HoloLens is really difficult!"

- "When an item is bought, print out the name of the item with a check mark somewhere in your point of view to make it more clear that you actually bought the item."
Chapter 6

Discussion

This chapter will present a discussion of the results with the research questions in mind, improvements, and things to think about when designing a similar system. The three research questions, we approached, were:

- How to locate multiple target objects in an indoor environment and guide the user to the objects while using an AR headset?
- How can the navigation be personalized to different users?
- Is AR headsets ready to be used with practical useful solutions?

The expected outcome of the project was to increase knowledge of how AR HMDs can be used to guide an user to real objects in indoor environments, and how personalization can be connected to this guiding experience.

6.1 Locating objects

The system works better than what was expected in the beginning of the project. It is possible to locate objects in the environment with the integration of common algorithms and methods together with new technology. A very important feature in this case is to have the system know its location in the world, and to have some kind of virtual representation of the world around itself. This data is needed to be able to process and generate walkable paths of the environment.

The inside-out tracking needed to create a virtual representation of a physical environment, have many pros, but also some cons. First of all, it enables the freedom to use the technology almost anywhere. To do so, it needs more processing power to recognize its position in the real world compared to outside-in tracking. Since inside-out tracking devices has to be independent and still have enough computing power, the price will be quite hefty. There could also be a problem for the inside-out tracking to recognize where it is in the environment, in case the environment have many identical looking areas.

A challenge that is very related to the indoor navigation problem is to help the users navigate between different floor levels, if this feature should work, more sophisticated algorithms
and heuristics needs to be implemented to help with more complex situations, e.g., stairs need to be found in the virtual representation and then be connected with the different floors, to generate new walkable nodes.

### 6.2 Design and Personalization

When testing the system, the accuracy of the paths location were set to be only a few centimeters, which seems to be enough for finding almost anything with good precision. This accuracy helps the user a lot to navigate and locate specific items in environments where the items are quite close to each other. Better precision equals to more nodes generated in the same area, and a bigger scan area also increases the number of nodes. If a bigger area is scanned, the precision will suffer, and vice versa. Hence, the area needs to be balanced against the precision to find the most optimal solution to the problem at hand.

The low field of view in the HoloLens made it so the users had to look almost straight at the point of interest, this made it harder to design a solution to certain problems, e.g., a common solution to show a path indoors, is to have a path showing the route along the floor. This could not be used in the system since it would make people look downwards while walking. Hence, losing track of their environment. Instead, an approach where the paths height varied was taken, this depended on how tall the user was. This way, the users were able to stay aware of their surroundings and see the path at the same time. The path line which became shorter and shifted color when the user got closer to the object, was meant to be an indication to the user that the object was close at hand and ready to be interacted with. However, some of the users got distracted when the line was all gone, thinking it was something wrong with the system. Some kind of extra feedback could be added when the users found the correct location, e.g., a sound or a small indication on the screen.

The system was tested in three different environments, an office, a store, and at an apartment. The apartment was only tested by myself to see if it behaved similar to the other environments. While the other two environments were tested by multiple people. A thing that differed in the environments is how big the minimum floor area could be when finding the floor positions. Apartments and offices with small doors between the rooms need to have a smaller minimum floor area than the large store can have. In the case of a too large minimum floor area, the system will not find a path through certain doors or smaller openings.

There can be a lot of people walking around in stores compared to offices and apartments, the users can have some kind of handicap, or the user’s can have different tools in the environments, e.g., a shopping cart in the store. In this case, it can be better to walk in the more open regions of the store and to avoid the cramped areas. In the home, and in the office we will probably not have these problems. This can be tweaked depending on the environment and the user’s preferences by altering the movement penalty of the objects, and the blur of the grid.

### 6.3 Improvements

There are two gestures available on the HoloLens, the tap and the bloom. The bloom gesture is reserved for the operative system, it can only be used in applications to exit them.
Hence, tap was the only gesture available to interact with the environment. This made it very hard to decide on how to show and hide the menu. There were options to use voice commands, but this could be irritating in a future case of several users using a headset, and happens to trigger the menu of another user with a voice command. Instead, the menu was placed in front of the user whenever a new area got scanned. However, this solution is not very good either since the position of the menu gets locked.

The environment scan can be quite time consuming in the case that everyone using it needs to do a scan by their own. In the future, there should be a simple way to download an environment mesh for a place the user never been in before, e.g., when connecting to a wifi, the device should be recognized and provided with the mesh of the environment. Which then recognizes and maps the locations of where it is in the downloaded mesh relative to the environment.

With better camera technology and object recognition it might be possible to provide easier ways to interact with real objects in a digital presentation. In our user tests, the user’s had to buy the items by “tapping” in the air while looking at the objects, this was a problem and took a long time for many of the users. If instead the technology could be merged with the real world in a way that will not change the user, but instead inspect how user’s interact with the real world, e.g., instead of tapping an item to buy it in our system, the user’s can simply put the item in the shopping cart and the system will understand by itself that it is an item to be bought, and taking it out from the cart will remove the item from the list of bought items.

In rare occasions the AR HMD can lose track of where it is in the environment, sometimes it corrects the location perfectly, and sometimes the entire virtual mesh can get an offset relative to the real world. This offset was discovered after demoing the system for around one hour, at an event inside an office building with around 80 people walking around.

The mesh will also get an offset when the AR HMD and the application is restarted. This made it harder to do the user tests in the store, since I did not want to leave the store and rescan it between each user.

Since the AR HMD is untethered it also depends on its internal battery. At the current state, the AR HMD will only be able to be used for around three hours before it is empty, it needs much more battery time if this technology ever shall become a common thing people will wear everyday.

These problems should be fixable with future hardware and software updates, e.g., there could be more gestures added, support to recognize downloaded meshes in the environment, new stronger hardware can increase the stability of the environment tracking, etc.

### 6.4 Practical useful solutions

The user test showed that AR HMDs can be used in at least one practical useful solution, since the t-test in Section 5.2.1 resulted in that the mean time it takes to find objects in a food store while using a headset is less than the mean time it takes without a headset. This can of course be discussed a bit further, since a setup scan is needed and the system easily can be offset after a while. A lot of other application areas where only a small scan area is
needed are more ready in the current generation. However, upcoming AR HMD generations might solve the current issues and provide a stable version that can even become "the new smartphone".
Chapter 7

Conclusion

We have presented an approach on how to guide people in indoor environments while using an AR HMD. After that, several suggestions were presented on how to personalize the system for different users and environments, e.g., generate a virtual mesh of the real world, recognize real life objects, understand where walkable areas are in the mesh, and how to generate a graph with this information. Our approach showed that people using the headset were finding objects in indoor environments faster than without, and that it is a technology they would like to see more of in the future. In case of development of new solutions, a lot of help can be obtained via open source libraries.

Future work include increased stability of the tracking, more research of how people prefer to interact with real and virtual objects in symbiosis with each other, how to share information between users to update the environment more dynamical, and how the similar systems will perform in larger indoor environments.
Bibliography


[33] A. Schade, “Customization vs. personalization in the user experience.” https://www.nngroup.com/articles/customization-personalization/

[34] F. Lardinois, “The next frontier for google maps is personalization.” https://techcrunch.com/2013/02/01/the-next-frontier-for-google-maps-is-personalization/.


Appendix A

Questionnaire: Navigating in a store using HoloLens
How intuitive do you think the system was? *

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not intuitive at all</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Very Intuitive!</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

How easy do you feel it was to learn to use the system? *

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>It was very hard.</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>It was very easy!</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

How easy was it to follow the path? *

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>It was very hard.</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>It was very easy!</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

How easy did it feel to buy an item? *

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>It was very hard.</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>It was very easy!</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

When facing the wrong direction, a red arrow slowly comes into view to provide feedback on where to look next. How did it feel when the red arrow came into the view to show where the path was? *

- O Annoying, it felt like it was coming up in the wrong occasions.
- O It was annoying having the arrow slowly coming into the view, but it filled its purpose.
- O It felt good, the arrow filled its purpose.
- O It felt very good to be guided with the arrow, it helped a lot.

To give feedback to the user, the blue paths line slowly faded to green when getting closer to the location. How intuitive did this feel? *

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not intuitive at all</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Very Intuitive!</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Chapter A. Questionnaire: Navigating in a store using HoloLens

How much feedback did the color scheme help to provide you with? Example: Red when something was wrong, blue when something is in process and green when something is finished. *

1 2 3 4 5
Not at all. ○ ○ ○ ○ ○ A lot!

When an item was bought, how easy was it to move on to the next item? *

1 2 3 4 5
It was very hard. ○ ○ ○ ○ ○ It was very easy.

This technology could also be used in Airports to find the correct gate, in the library to find a specific book, etc. Would you use technology similar to this in the future to navigate in indoor environments? (If the headset is smaller and more affordable) *

○ Yes!
○ No!

Is there anything you would like to add to the application, which you thought was missed? Please feel free to come up with new ideas and changes. *

Ditt svar