VRIJE
UNIVERSITEIT
BRUSSEL

Graduation thesis submitted in partial fulfilment of the requirements for the degree of Bachelor in Computer Sciences

ROBOMR: A MIXED REALITY ROBOTIC GAME

ARTHUR VINCENT CHOMÉ

Academic year 2018–2019

Promoter: Prof. Dr. Beat Signer
Advisor: Payam Ebrahimi, Ahmed K.A. Abdullah
Faculty of Sciences and Bio-Engineering Sciences

VRIJE UNIVERSITEIT BRUSSEL

Afstudeer eindwerk ingediend in gedeeltelijke vervulling van de eisen voor het behalen van de graad Bachelor of Science in de Computerwetenschappen

ROBOMR: A MIXED REALITY ROBOTIC GAME

ARTHUR VINCENT CHOMÉ

Academiejaar 2018–2019

Promoter: Prof. Dr. Beat Signer
Advisor: Payam Ebrahimi, Ahmed K.A. Abdullah

Faculteit Wetenschappen en Bio-ingenieurswetenschappen

Abstract

In this thesis, we propose a framework for the navigation of virtual agents inside a dynamic robotic Mixed Reality environment. It will allow virtual objects to track a physical robot using spacial mapping. We will illustrate the findings of our developed framework by realising a Mixed Reality robotic game where a user can steer a robot with a head-mounted device. Using path-finding algorithms, our application will find the shortest and most efficient path that the robot can use to reach its destination. The framework we want to develop would allow virtual agents to track and follow the robot around. The user can interact with these virtual objects by shooting them down using the physical robot. This ensures that there is interaction between the environment's physical and virtual objects.

Acknowledgements

I would like to thank my advisor Payam. He gave me great feedback for completing my research paper every week again and this in spite of his other responsibilities as a PhD student. I had no experience in writing a formal research paper and gathering good references proved to be tricky at times. Luckily, Payam gave me good advice at every step of the process to prevent rooky mistakes. He was a great advisor overall.
The same can be said about Ahmed. He gave good recommendations and showed great patience when mistakes occurred. He also introduced me to LateX[1], Google Scholar[2] and many more handy tools for writing a paper. In short, a very good advisor.

[1] https://www.latex-project.org/
[2] https://scholar.google.be/

Contents

1 Introduction

Video games and computer graphics have come a long way from their inception in the 1950s[1]. Going from simple vector graphics to fully 3D rendered games, the player's gaming experience and immersion has increased dramatically over the past decade. The next step in the search for greater immersion is to completely surround the user in the game space instead of limiting it to the screen. This can be done with Mixed Reality giving the user the impression of being physically present in another world.
Mixed Reality is a set of technologies that involves combining elements of the real and virtual world ~~together~~ to create a new world where the two can interact with one another. As indicated in Migram's 1994 paper[14], Mixed Reality can be classified somewhere in between the real and virtual environment on the virtuality continuum. The two sides of the spectrum refer to an environment where everything perceived is part of the real world and an environment where everything perceived is computer-generated (virtual reality) respectively. This makes MR a pretty broad research field.

[1] https://en.wikipedia.org/wiki/Video_game_graphics

In recent years, there has been a surge in public availability for Mixed Reality technologies. For Augmented Reality alone, market forecasts predict a total revenue of around 80 billion US Dollars by 2021[5]. Research for MR however has existed for a considerably longer time going back as far as the Virtual Fixture system developed in 1992 at the United States Air Force Research Laboratory [2]

The goal of our thesis is the development of a framework for navigation in an MR environment that lets virtual elements hover around certain physical objects. We will demonstrate our findings by developing an MR robotic game where one uses a Microsoft HoloLens[3] to steer a robot around that is able to interact with the digital world. An example of this can be found in the problem identification section.

[2] https://en.wikipedia.org/wiki/Virtual_fixture

[3] https://en.wikipedia.org/wiki/Microsoft_HoloLens

1.1 Problem Identification

During the Build Developers Conference[4] held by Microsoft in 2015, Alex Kipman demonstrated the HoloLens' cross-space capabilities involving the B15 robot [5]. In the demo, we can see the user control the B15 by indicating waypoints that the robot improves with path-finding. They also managed to overlay a holographic robot on top of the physical robot that follows the real robot around based on the vision of the Microsoft HoloLens. This was done in order to make the experience more immersive.

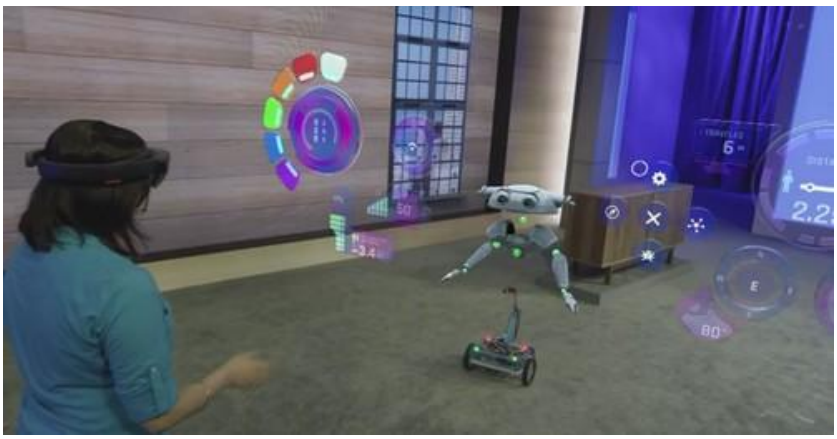

Figure 1.1: Using the HoloLens and a virtual interface, the user can specify a path for the robot to follow. This image was adopted from the Youtube video in footnote.

The issue here is that not all the source code of the project is freely available to the public. The whole goal of the research thesis is to develop a software application inspired on this Microsoft project and make it freely available for researchers to use and build upon it. However, there are some problems we have to account for namely navigation and the interaction between the virtual agents and the robot.

1.1.1 Navigation

To the best of my knowledge, there exists no publicly available framework for navigation inside a dynamic environment. There is —however— a master student who for his final thesis has to develop a similar MR framework with the ability for navigation at the WISE Lab[6]. Unfortunately, his work is not

[4] https://en.wikipedia.org/wiki/Build_(developer_conference)
[5] https://www.youtube.com/watch?v=xnrHFV34PfM
[6] https://wise.vub.ac.be/

ready for release. This is why developing such framework will be the focus of our thesis.

1.1.2 Interaction Virtual Agent and Robot

From the perspective of the user, there should be a seamless interaction between the robot and the virtual objects. The robot should be able to see the game objects and interact with them based on what it sees from the head-mounted device of the user. The perception problem also applies to the game's virtual objects: what vision do they have of the physical robot?

1.2 Objectives

Having dealt with the problems of implementing a Mixed Reality application similar to Kipman's demo, our research question is thus as follows: "Can we make a framework that would display similar features to Microsoft's demo and make it available for researchers?". For solving the stated problems, we have to formulate some objectives.

Navigation Framework

We will have to define a navigation framework for a dynamic Mixed Reality environment. It should allow virtual objects to track the physical robot. It will require some efficient path-finding algorithms as well as spatial mapping and tracking algorithms.

System Design

I would like to explore the feasibility of implementing an alternative system architecture than the classic client-server model[7] for my framework. We could alter the configurations as to make the objects of the virtual space and the objects of the physical space equally important. Making the application peer-to-peer based as to spread the computation over multiple components could help in this objective.

[7] `https://en.wikipedia.org/wiki/Client-server_model`

1.3 Methodology

For this thesis, we have chosen to use the Design Science Research Methodology for information systems applied to Mixed Reality systems[16]. We will clearly elaborate our activities for each step:

- Problem identification and motivation: this has been done in section 1.1.
- Define the objectives for a solution: the objectives for the success of this research effort have been specified in section 1.2. The overall objective would be to develop a navigation framework for MR and explore possible architectures to do so.
- Design and development: the design and expected features for our solution has been established after repeated meetings and discussions. Chapter 3 gives a more high-level explanation to our solution while chapter 4 discusses the more low-level implementation details.
- Demonstration: we will provide a demo of our software application during the final thesis presentation.
- Evaluation: we will assess to which extent the developed product answered to the stated problems. The goal is to disclose the usability of the robotic game and —as a consequence— our framework. We will also discuss development issues that we faced. This can be found in Chapter 5.

The focus of this phase is on the problem identification and defining the objectives. We will also provide a brief overview of the technologies we plan to use in the second phase. The other steps —like the project development— will be covered in that second phase.

1.4 Research Paper Outline

In line with our research methodology, our scientific research paper has following structure:

Chapter 2: a discussion of the general concepts used for the project by referencing previous research papers. This related work sections helps us in specifying our thesis requirements. It is the basis for the offered solution of chapter 3.

Chapter 3: a presentation of the final mixed reality robotic game with a focus on the developed navigation framework: its features, requirements, relevance for future research by other scientists...

Chapter 4: the implementation of the framework and the subsequent application: this includes its system architecture, design choices, used tools and technologies and a discussion of the programming process.

Chapter 5: a critical reflection on the development process leading to the final application and an evaluation of the final result.

Chapter 6: a conclusion to the paper. We repeat our solution and in which ways it deviates from the original requirements and discuss the possibility of future work.

2 Related Work

Now that we identified our problems and formulated our objectives for the research topic, we will do a literature study to refine them. The found related work represents the current knowledge the scientific research community has about the field. The discussed papers focus on Mixed Reality for robotics, navigation in a Mixed Reality environment, computing architectures, target selection, occlusion in Mixed Reality and the Microsoft HoloLens. This broad selection will allow us to either validate or refine our research question and the objectives. This is very important since a brand-new paper about something that has already been researched before would be redundant. This work should be useful and allow researchers to build upon it.

2.1 Overview

Mixed Reality technologies allow us to perceive things that others cannot through an enhanced view of our surroundings. Its utilities and applications are very broad from simulating real-life tests for robots[7] to pedagogical uses like educating an assembly line worker[5].
An application we could more relate to, would be the popular game Pokémon GO[1]. It is known for bringing augmented-reality technologies to the public,

[1] https://en.wikipedia.org/wiki/Pokémon_Go

increasing its accessibility. The game allows players to explore the world enhanced with computer-generated Pokémon like Pikachu or Charizard. They can proceed to Pokéstops that function as the game's general stores and to Pokégyms to battle other players. All of this gets displayed on the screen of the user's smartphone. The player can also move to wild Pokémon to try and catch them[2].

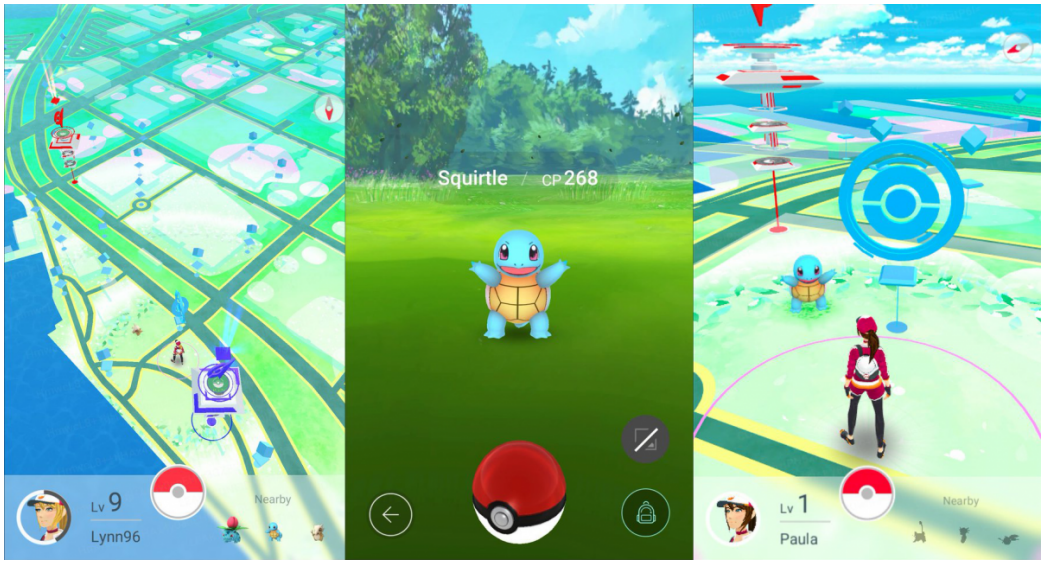

Figure 2.1: These screenshots show different views for the player. To access Pokéstops and Pokégyms, the player must travel the real world. His mobile screen shows his current position allowing for orientation. These pictures have been taken from Edubilla[2].

2.1.1 The Microsoft HoloLens

It is a head-mounted device that allows for the mapping of virtual agents over the physical world and it will be a key technology for our application. It has been such a success for Microsoft that it spawned a collaboration[3] with technology firms Asus and Samsung. These two firms now develop their mixed-reality software applications around the specifications of the HoloLens. The paper by Gabriel Evans[5] evaluated the HoloLens on an application for educating people to use an assembly line. Users would get directions on

[2] `http://www.edubilla.com/articles/play-and-games/people-goes-crazy-behind-the-augmented-reality-game-pokemon-go/`

[3] `https://en.wikipedia.org/wiki/Microsoft_HoloLens`

where every piece of hardware should go to assemble a product, making for a very educative experience.
The conclusion of the paper is that the HoloLens is a more than capable device to deliver a Mixed Reality experience. The hardware capabilities of the HoloLens "allowed for virtual content to be spatially located correctly enough for assembly instructions". The display of the head-mounted device was also more than sufficient for the very detailed user interfaces programmed for the application to improve ease of use.

2.2 Robotics

Referencing the work of researcher Wolfgang Hönig[7], we can see that Mixed Reality has come a long way in robotics. The combination of these two disciplines has proven to be a great way to come to even more new findings. This follows out of Hönig's statement that "the central contribution of this work is to establish Mixed Reality as a tool for research in robotics".

2.2.1 Benefits

With Mixed Reality, we can create an environment where physical objects seamlessly interact with virtual objects. It would therefore be feasible to make physical robots interact with virtual components. This brought forth some very neat advantages for the involved scientists in the field of robotics improving their research and overall productivity.
Scientists can now witness remote tests performed on their built machines by way of Mixed Reality. They no longer need to travel great distances to the testing grounds as they can follow everything through a head-mounted device. They thereby meet in one virtual environment instead of a physical one. This improves their overall presence and collaboration considerably since there are no more travel distances between collaborators. This situation is described as spatial flexibility[7] and refers to a situation where "interaction between physical and virtual environments in MR allows experiments with robots to be performed remotely". This is all made possible by the recent advancements in internet connectivity.
Another advantage is the enormous safety benefits linked to using virtual environments. No longer will human participants be put in potentially hazardous situations by interacting with experimental machines. These machines are still to be tested and could potentially show unexpected behaviour dangerous for the humans involved. Using humans for testing would be a thing of the past and experiments would only use virtual models.

MR applications have the ability to add computer-generated information to their environment that can easily be changed at will. This would make for much richer testing grounds for the experiments. “This expedites and simplifies debugging because the difference between implementation and simulation becomes even smaller”.

2.2.2 Experiments

What follows are practical experiments extracted from Hönig’s paper[7]. The research team conducted three experiments using cameras mounted on Unmanned Aerial Vehicles[4] (UAV).These are so called "drones" and are aircraft that do not have a human pilot on board. All control of these devices happens remotely.

UAVs Following a Human Model

The first experiment[7] focused on making drones follow a virtual simulation of a human around. But since human movement is “complex and unpredictable”, researchers believed their whole experiment to be not “very accurate”. The whole purpose of the experiment was to use human models with MR to limit safety risks since no real humans would actively participate with the experiments. It is worth mentioning that the team used the Unity game engine[5] which we will use extensively in the second phase of the thesis.

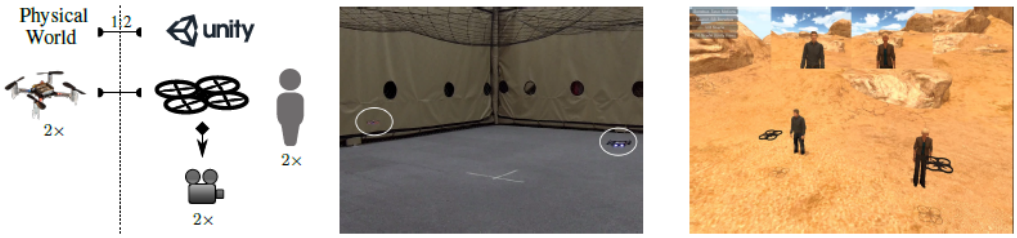

Figure 2.2: Above pictures show 3 views on the experiment. From left to right: abstract image of the used Mixed Environment, a view of the physical environment and the virtual environment generated by the Unity engine. With MR, the two quadcopters follow virtual objects around who simulate human movement. These 3 images have been adopted from Hönig’s paper[7].

[4] https://en.wikipedia.org/wiki/Unmanned_aerial_vehicle
[5] https://en.wikipedia.org/wiki/Unity_(game_engine)

Spreading UAVs on a Wide Area

For the next experiment, the team implemented a swarm algorithm that managed the spreading of drones over a wide MR environment. In the environment, its computer-generated elements continuously changed to challenge the swarm algorithm on its robustness. This way, the team could test how a group of drones coped with an ever-changing environment. It "allows for swarm algorithms to be tested in a simulation that more closely depicts the actual physical motions of a robot".

Figure 2.3: 3 views on the experiment. From left to right: abstract image of the MR environment, the physical environment with the drones spreading out and the virtual environment representing what the drones see. The pictures have been extracted from Hönig's paper[7].

Robot-tracking UAV's

The final experiment involved controlling two ground-based mobile robots based on a drone's camera vision with the purpose of moving a box around in a specific direction. This was realised using technologies such as augmented reality tags[6] that calculate the UAV's camera position relative to the robot and some advanced image processing algorithms. This is quite useful for our paper since it highlights key technologies that we can use for tracking and moving our own robot using some off board camera. The drone would in this case be the user with the HoloLens.

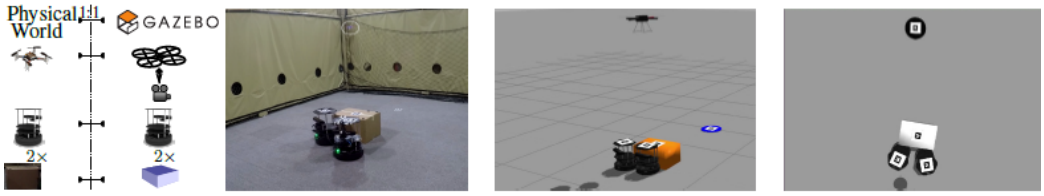

Figure 2.4: 4 views on the experiment. From left to right: abstract image of the experiment, the physical environment with the box and the two robots, the virtual environment and the virtual view of the quadcopter hovering above the robots. These pictures have been extracted from Hönig's paper[7].

2.2.3 Evaluation

The 3 experiments are interesting applications of MR. The first experiment tackles the issue of tracking a virtual object by a physical machine. Be it a drone, be it a Lego robot. The successful tracking could be applied on our project where the robot interacts with the virtual "targets".
The second experiment focuses on ever changing environments. Making our application robust enough to handle such surroundings could improve its reliability considerably. In the context of our MR game, this is handy when there is a lot of virtual movement that could put some strain on the robot. Think about higher levels with a lot of "enemies" to shoot down.
The last experiment involves making ground-robots move a box by way of visual perceptions from a UAV camera. In our application, it must be possible for the user to move the robot to any location he wants. This is one of the key requirements for our final robotic game. Though the controlled robots have the objective to move a box in a certain direction, this can be easily generalised to user-controlled movement in any direction.

[6] `https://en.wikipedia.org/wiki/ARTag`

2.3 Navigation

Navigation[7] involves determining the position of an object in an environment and directing it to another specific position that you would like that object to be at. We will be focusing on the navigation of virtual objects and the user inside an MR environment. To do so, we will analyse what the scientific world already knows in this field and aboard a few papers that might serve our cause. We will conclude by revising our research objectives based on our findings.

2.3.1 Navigation for Humans

Navigation in MR focuses on providing appropriate information to the human users about their surroundings to more easily move across it. This way, users can keep track of the environment without the issues related to disorientation due to the added strain of computer-generated information.
We reference Cheng's paper[2] for further insights. The paper presents a Mixed Reality application that offers a tour of the Next Gene 20 Project[8] in Taiwan. It would allow the users to explore the built projects in a more informative and immersive fashion all while correctly navigating them through it. This would be perfect for our robotic application. A possible feature we could have in our game based on these possibilities, would be of assisting the user of he ever were to lose track of the Lego robot. The system would then give feedback to the user to correctly navigate him or her back to it.

Next Gene 20

The Next Gene 20 Project[8] is an International Architecture Project from a few years back. It was hosted in north-eastern Taiwan and consisted of a group of both local and foreign architects co-operating to build different houses following the landscape architecture[9] style. Similar to the Museum of the Future[8], this project focuses on enhancing the user's experience by adding computer-generated information to the site he is visiting. All this with head-mounted devices. The whole project shows us that current MR technology is more than capable of navigating a human through an environment with added computer-generated information.

[7] https://en.wikipedia.org/wiki/Navigation
[8] https://inhabitat.com/next-gene-20-project-taiwan/
[9] https://en.wikipedia.org/wiki/Landscape_architecture

Figure 2.5: Stourhead in Wiltshire, England. It represents a typical work of landscape architecture. The image was adopted from a Wikipedia article[9].

Added Technology

To add to the user's immersion, the paper questions if an MR interface can do something more than offering the traditional input tools like a mouse, keyboard or joystick. Even for 2008, more technologies surged that added to the user's experience.
The Mixed Reality system introduces a new hand-held device that gives the user more information about his surroundings if he were to place it on certain locations. "Tangible controllers move around on the real navigation map to create perfect connection between the navigation map and 3D virtual reality environment." This technology makes the user's experience more informative as a whole.

2.3.2 Navigation for Virtual Components

The previous section has shown us that our current knowledge allows us to perfectly help a human navigate in a mixed environment. It even showed us that his tools for expressing himself (user input) has come a long way from the traditional mouse and keyboard devices. But how about navigating a non-human, non-physical object? A computer-generated object that does not exist but in the enhanced reality we try to build up?

Tracking Physical Objects: Effective Mapping

To navigate a virtual object in a real-life environment, we must be able to correctly and precisely perceive the physical surroundings through our sensors. We will refer to Richard A. Newcombe's paper[15] that presents a new surface mapping and tracking algorithm. It would allow us to overcome one of the biggest issues for Mixed Reality[10]: the incorrect depth calculation of the environment resulting in poor mapping of virtual objects and a poor performance for the whole application.
The featured KinectFusion method is able to do a real-time mapping of the environment using only a hand-held depth camera. Applying this method on our application with the Microsoft HoloLens would make it more robust since KinectFusion delivers a more complete mapping of its environment. Overall, this seems like a good mapping algorithm for our application.

2.3.3 Path Finding Algorithms

So far, we focused on the perception the virtual agents and the robot must have of each other to effectively cooperate. We also discussed the navigation of the user and the virtual agents relative to the robot together with some basic environment mapping.
We will now go in more detail in the navigation of the robot and more precisely the path-finding algorithms it relies on. In the application, the user must specify a path it has to take that must also be improved if necessary by these algorithms.
We will refer to Abiyev's 2015 paper[1] for further insights on the issue. In it, the researcher and his team compare a set of path-finding algorithms with different complexities and performances.

Experiment: Robotic Soccer Game

The experiment in Abivey's paper[1] resolves around optimising a robotic soccer game[10] for 12 fastly moving robots in teams of 6. This is done —like mentioned above— by comparing a set of different algorithms while presenting an improvement of an already existing algorithm. The visual perceptions are supplied by two overhead cameras positioned 4 meters above the soccer field. Path finding is discussed within the context of a very quickly changing environment where opposing robots obstruct newly calculated paths and steal the ball from one another.
Though some discussed algorithms are slower than other ones featured in

[10] `https://www.youtube.com/watch?v=qNaBUs7gP_A`

the paper, they are less complex to implement and still fast enough for our application. We will be focusing on those slower algorithms. To make our robotic game applicable on even the most difficult environments, we will also discuss the improved algorithm featured by the paper.

Figure 2.6: Using omni-wheel robots and some fast path-finding algorithms, a robotic soccer game gets realised. This image has been taken from a Youtube-video[10] of the Robocup Robot Soccer Finale of 2013 in Eindhoven.

A*-algorithm

This algorithm discussed in the paper[1] is overall very popular for solving the path-finding problem. The algorithm works by starting at a begin point in the environment and tries to find a route to a specific endpoint with waypoints in between. Every waypoint has a specific distance to cover to reach it from a neighbouring waypoint. During the algorithm's execution, we maintain a tree of paths that all start from a specific begin point. At each iteration, the algorithm adds another node to it until a path to the end location has been found. Though not as efficient as other algorithms in this paper and too slow for a robotic soccer game where the environment changes a lot and quickly, it is still a great fit for our goal. Especially since it does not need any prior knowledge of the surroundings it gets applied to. This is a big selling point for our application since it has to be applicable on any surroundings. Another thing to consider is that the environment is fairly static: there should not be any fast moving entities in our environment that block up paths we created for the robot forcing us to find alternate paths.

RRT Smooth

The abbreviation "RRT" stands for "Rapidly-exploring Random Tree". It is a path-finding algorithm that works by arbitrarily generating trees that fill up our problem space. The original algorithm quickly finds paths between two locations but it is sometimes not the shortest possible path that can be found. Worse still is that the paths found can sometimes be quite long and overall quite bad.

Iterative RRT Smooth

The paper[1] therefore introduces an optimised iterative version that finds “feasible and near optimal solutions in short time and shortens the path length considerably”. It is part of the more complex algorithms featured in the paper. Since the user can move around changing the camera angle and making robot tracking more difficult, it can be useful. Especially when there is a lot of user movement.

2.3.4 Evaluation

We have covered the navigation for both the robot, the user and the virtual agents and discussed some path-finding algorithms for the robot specifically. In the process, we made some interesting findings.

Navigation for Humans

Cheng’s paper[2] showed us that our current knowledge in MR allows us to perfectly navigate a person in a mixed environment. Navigation in our application could be used for assisting the user in finding its robot if he were to lose track of it. Note that this is only a suggestion and not a fixed objective for the second phase of the thesis.

Navigation for Virtual Agents

We found an efficient mapping and tracking algorithm [15] that would allow us to have a better grip of our surroundings. This would make for a better positioning of the mixed environment’s virtual objects and their navigation. In turn improving the application’s performance and reliability.

Path Finding

Abiyev's paper[1] discussed an extreme case of path-finding with two remote overhead cameras applied on fast-moving robots playing a competitive game together. Path finding on a slow Lego robot should be easy in comparison. Another worry here would be to implement efficient methods for the user to express where the robot should move to. In other words: providing user input.

2.4 User Input

The ability for the user to interact with our system is crucial. It has to be able to express such actions as making the robot move around to a certain location and making it interact with the environment including the virtual agents. Some advanced methods for user input need to be discussed to realise such expressiveness for the user. We will be relying on Kyto's paper[11] for an overview of modern techniques of pinpointing which is the "use of multimodal pointing techniques for wearable (AR) interfaces". These would allow users to select virtual or real objects in their environment. The paper focuses on eye gaze and head movement as primary methods for user input and also provides discussions to combine these. This revealed some performance improvements.

2.4.1 Eye Gaze

Aside from Kyto's paper[11] for a broad overview of pinpointing techniques, we will discuss eye gaze more specifically by relying on Lupu's survey[12] that focuses on eye tracking algorithms.
The functioning of eye gaze works by deducing the eye position and calculating the area they are pointing at and then implicitly or explicitly specifying whether you want to select that area. Explicit selection can be done with -for instance- a handheld device. Though eye gaze can be a good technique for the user to express himself within the application, it "suffers from low accuracy due to calibration errors and drifts of wearable eye-tracking sensors"[11]. It also has the problem of unwanted selection since a lot of the user's eye movement happens unconsciously.

2.4.2 Head Movement

A next pinpointing technique in Kyto's paper[11] is head movement. Though more precise than eye gaze for pinpointing, it has proven to be quite tiresome to use since the whole head has to be moved instead of only the eyes. To alleviate the user's efforts, we could use it as a supporting technology for eye gaze.This would be a way to cope with the low accuracy of eye gaze improving the user input's preciseness.

2.4.3 Evaluation

After reading Kyto's paper[11], it became clear that both eye gaze and head movement have their advantages and disadvantages. Eye gaze for instance is —when it comes to input speed— overall faster than head movement but it lacks accuracy because of technological limitations and the very fast nature of eye movement.
Head movement is more accurate but requires more effort of the user to handle since he has to move the whole head and neck area instead of just the eyes. Head movement proved therefore to be a tiresome input method. Moreover —when it comes to user input speed— it is slower than eye gaze.
By combining these two techniques, we could combine the best of two worlds: the accuracy of head movement with the speed and natural feel of eye gaze. This while limiting user fatigue by too much head movement.

Robot Control

The user can use both eye gaze and head movement techniques to control the robot. This would allow him to select the location in the MR environment of where the robot should move to. Any path optimisation would be handled by the path-finding algorithms that we already covered in the previous section.

Target Selection

Under "target", we mean the virtual agents that track the robot. These have to be shot down using the robot. Though for a video game it would be quite precise to use a handheld device for manual input, we are obliged here to use the full functionalities of the Microsoft HoloLens. Since selecting a hovering virtual agent in an MR environment is a work of precision, head movement supported with eye gaze looks like the best option.

2.5 Occlusion

Occlusion in Mixed Reality means that virtual objects that are "further" away from the user than physical objects can hide behind these "real" objects. Occlusion allows the user to have an increased perception of virtual objects by giving him the impression that virtual objects are positioned behind real objects. This would create the illusion that they are part of the environment[4]. A good paper that discusses the occlusion problem would be the paper[17] of researcher David Walton of the University College London[11].

2.5.1 Previous Techniques

Walton's work[17] starts by offering on overview of known methods for occlusion. This is a good starting point to explain what the presented new method does better. The problem with all existing methods is that they typically require some proper prior knowledge of the environment and more specifically on the physical objects on which occlusion must be applied. This must be either inserted by hand or "reconstructed using a dense mapping algorithm". Since our application should be applicable on any environment, we must find a method that does not need any prior information about it.

2.5.2 Cost Volume Filtering Occlusion

The good news is that Walton's paper[17] presents a new method for occlusion that does not need prior environment knowledge. It does —however— need a RGBD camera. Based on the colour levels it perceives and by depth estimations of the environment, improved occlusion is achieved.
It works by using "incomplete data to accurately simulate the occlusion of virtual content by real content in video see-through augmented reality scenes". The algorithm is very efficient, so much so that it is "capable of detecting partial occlusion at a pixel, allowing it to avoid aliasing along occlusion edges".

2.5.3 Evaluation

Immersion is important in Mixed Reality: good applications in that category should make users feel they are in a totally new world[13]. Part of this realization is to make occlusion happen, it would make virtual objects able to —al be it partially— position themselves behind real objects. Virtual

[11] `https://en.wikipedia.org/wiki/University_College_London`

Figure 2.7: The top two pictures present less than optimal results. The lower result was generated using the Cost Volume Filtering Occlusion method. The pictures were adopted from Walton's paper[17].

agents hovering next to the physical robot will sooner or later be "behind" it. The application must then make sure that occlusion happens.

2.6 Mapping of Virtual Agents

We will discuss the appropriate spatial mapping and tracking algorithms to overlay virtual objects on the physical environment. This all the while not having any knowledge about the terrain that gets mapped. This essentially means that an application having this feature can create a new mixed environment out of any environment. This is quite a novelty and is attributed to the emergence of better cameras and other sensors. It has already been discussed —be it briefly— in the navigation section of the paper but will now be treated in more detail. For further insights, we will be referencing Comport's paper[3] that presents a new "3D model-based tracking algorithm".

Moving Edges Algorithm

This algorithm is used for tracking moving physical objects. Globally speaking, "tracking is dependent on correspondences between local features in the

image and the object model. In an image stream these correspondences are given by the local tracking of features in the image". More trivially explained, it is called the moving edge algorithm because it finds the edges of an object with some simple tracking algorithms. This way, it can identify and track a physical object around.

Virtual Visual Servoing

Normal visual servoing[12] is a technique that consists of steering a robot based on the feedback it gets from a visual sensor monitoring all of its movements. This way, it can improve its current task. Virtual visual servoing consists of correcting virtual objects —not robots— based on their current positions in the real world as seen by our head-mounted device. This is important since the mapping of the virtual agents hovering around the robot must be as precise as possible for the sake of usability.

2.6.1 Experiments

The researchers demonstrate the features of their proposed tracking algorithm by way of a series of experiments[3]. They all use a commercial digital camera to perceive their environment. If using such device is possible with the given algorithm, this must mean it is quite efficient for not requiring expensive hardware to operate[10]. Given that the experiment succeeded with a commercial camera, using the Microsoft HoloLens should therefore be no problem. The distances of the real-life objects are calculated relative to the camera. More specifically, it gets calculated by way of the moving edges algorithm discussed above.

Tracking in Indoor Environments

The first experiment involves tracking in an indoor environment. With tracking, we mean that the used visual sensor or camera follows elements in the physical environment. The image estimation here has to be of high precision if we want the tracking algorithm to succeed. The experiment successfully tracked four 3 dimensional circles around.

Tracking in Outdoor Environments

A little bit trickier than previous experiment since outdoor environments provide more natural factors (rain, etc.) hindering the mapping. The experiment had to track a cylinder and correctly place a virtual object relative to

[12] `https://en.wikipedia.org/wiki/Visual_servoing`

it. “Despite very noisy images, tracking was still achieved” making the experiment an overall success. This experiment could be applied to effectively make virtual agents hover around the physical robot.

2.6.2 Evaluation

This article was chosen to tackle the problem of mapping virtual objects on the physical environment. To do this, the virtual objects' positions towards the physical robot must continuously be corrected or checked. If not, the placements of virtual objects can be so bad that the immersive experience would be severely undermined. The principal technology used for this is virtual visual survoing with the moving edges algorithm. The article[3] confirms that we can correctly position virtual game objects by way of a moving camera perspective, in this case the Microsoft HoloLens.

2.7 System Design

There are different system architectures that we could consider for our prototype. Kipman's demo was for instance server-client based but it could be interesting to research for some alternatives. Especially since it could offer some advantages for future development. Hence, we will explore our options.

2.7.1 Server-Client Based Application

The most trivial choice would be a client-server architecture[13]. It organises a system in such a way that part of its components provide computations for other components that request them. The former are called servers and the latter clients.
This architecture can easily be applied to our prototype MR game. Yet, we will not be using a remote server[6] for this but instead use the Microsoft HoloLens to host the application's server. This allows our prototype to work more independently and it eases testing. We also have a Mindstorm[14] robot that can also do some computation all be it to a lesser extent.

[13] https://www.webopedia.com/TERM/C/client_server_architecture.html
[14] https://www.lego.com/en-us/mindstorms

2.7.2 Peer-To-Peer Based Application

The problem of server-client based applications is that having some system components do all the computations can create quite a performance bottleneck. Especially by adding too much clients that could overwork the servers. Luckily, we can divert part of the workload to other components like our Mindstorm robot if need be. It is also important to consider that the application may have more components than originally envisioned in future development.
All these problems —namely scalability and the server-client dependency— could be alleviated by using a peer-to-peer architecture[15]. In it, all system components would be of equal importance and all provide and request computation. Having no real dependency between the nodes could prevent a lot of system crashes since no component would be critical for the system to work.

Shared Environment

A way in which we could expand the system —both in features and components— would be to implement a shared Mixed Reality environment that would allow multiple users to interact with each other. Note that we are considering this feature. Like mentioned above, for the sake of scalability and crash-prevention, using a peer-to-peer architecture seems like a good option. It would potentially allow an unlimited number of participants to the system.

Solaris System

Keller's paper[9] presents the Solaris system[16] that uses a peer-to-peer architecture to create a shared virtual environment. In it, every system component would provide and request computation which solves the performance bottleneck of having centralised servers do all the work by spreading it over all system participants. Solaris is also highly-dynamic adapting very quickly on system occurrences like ever-changing connections between nodes or users dropping in and out as they like.

[15] https://www.techopedia.com/definition/454/peer-to-peer-architecture-p2p-architecture
[16] https://en.wikipedia.org/wiki/Solipsis

User Interaction

A neat feature of the Solaris protocol[9] would be that it allows nodes to communicate their presence to neighbouring nodes as far as their hardware allows it. This means mobile computers like smartphones are still able to contribute to the peer-to-peer network, even if their computation abilities are pretty limited. This only lowers their radius of communication with the adjacent nodes.
Having a radius on which other closely positioned nodes can be in, it calculates which nodes are close enough to interact with. When two nodes are both within communication range, the system makes sure the nodes can interact with each other. It should be noted that if a node's computation capacity is very low, it might not be able to communicate with all nodes that are close enough to it.

2.7.3 Evaluation

We considered both the client-server and peer-to-peer architecture for our application. Having an alternate system architecture or even designing our framework as to be able to use both, could significantly extend its usability. Though the Solaris protocol is mostly applied to Virtual Reality, we could extend its usage to a Mixed environment. In it, the avatars' position would be the physical position of the users with each carrying a head-mounted device. They would all perform and request computation to make sure they can interact with one another and with the virtual elements in the environment. Again, this is only an exploration of what our application could offer.

2.8 Requirements

Using the gathered concepts and techniques from previous research work, we would like to formulate and develop a new application. It will be used as a framework for the navigation of game objects within a Mixed Reality environment.

2.8.1 Tools

To allow for Mixed Reality, we need a device capable of enriching the user's visual perception of the environment with additional computer-generated information. In this regard we have the Microsoft HoloLens[17] at our disposal.

[17] https://en.wikipedia.org/wiki/Microsoft_HoloLens

For development and debugging, we're using our own laptop.
We require a remotely controllable robot as an anchor point to place virtual objects in the environment. It must also have mobile capabilities since our navigation framework must also be able to correctly map game objects even when the environment changes dynamically. For this purpose, we have the Lego Mindstorms[18] robot.
To allow communication between these 3 devices
As for the software, we require different IDE's. We need the Unity editor to implement the server, Eclipse for developing the Java client and Microsoft Visual Studio for the deployment of the project on the HoloLens. We also discovered a number of toolkits to help us in the development process such as Microsoft's HoloToolkit[19] and the Vuforia[20] toolkit.

2.8.2 Functional Requirements

Based on these tools, we are able to develop a Mixed Reality application. We still need a list of basic functionalities that the software program should abide to.

Robot Navigation

The previously mentioned robot should be programmed to move in different directions based on the user's indications. This is necessary as to show that the framework can track a moving real-life object around and correctly place game objects on top of it.

User Input

The user must be able to direct the target robot in a certain direction. This can be realised in different ways. Voice commands —for instance— would require identifying certain keywords spoken by the user that would trigger the server into sending instructions in which way the robot should move. More complicated input methods would be specifying waypoints using eye gaze that the robot should follow using a path-finding algorithm. Depending on how the project development goes, different input methods will be developed.

[18] `https://en.wikipedia.org/wiki/Lego_Mindstorms`
[19] `https://github.com/Microsoft/MixedRealityToolkit-Unity`
[20] `https://developer.vuforia.com/`

Object Tracking & Mapping

A target object has to be overlayed with virtual objects. For this to be possible, it must be formally identified in the environment and tracked efficiently. Game objects must still hover on top of the robot by following it in its new course.

Spatial Awareness

This feature along with finding a path for the robot to follow and mapping virtual objects requires some advanced spatial awareness. For this to be possible it would require extensive geometry of the environment to allow for virtual objects to interact seemingly with the real-life world.

3

Mixed Reality Robotic Game

In this chapter we give our theoretical solution for the problem domains discussed in the thesis introduction. We start by giving a general overview of our application and discuss the created navigation framework for the correct placement of virtual objects inside a dynamic environment. We also give our requirements for evaluating said framework. We then go through the process of realising a functioning prototype for the application.

3.1 Application Overview

Our Mixed Reality application consists of two main components: the client and the server. The two are connected with an established computer network using the internet protocol suite to forward data packets to each other. There is an inherent master-slave relation between the two because the server is sending all packets through the network to the client. The client only has to connect at the start with the server and then listen to its input for packets that the server sends. The client and server in our application are hosted on different devices mentioned in the literature study.

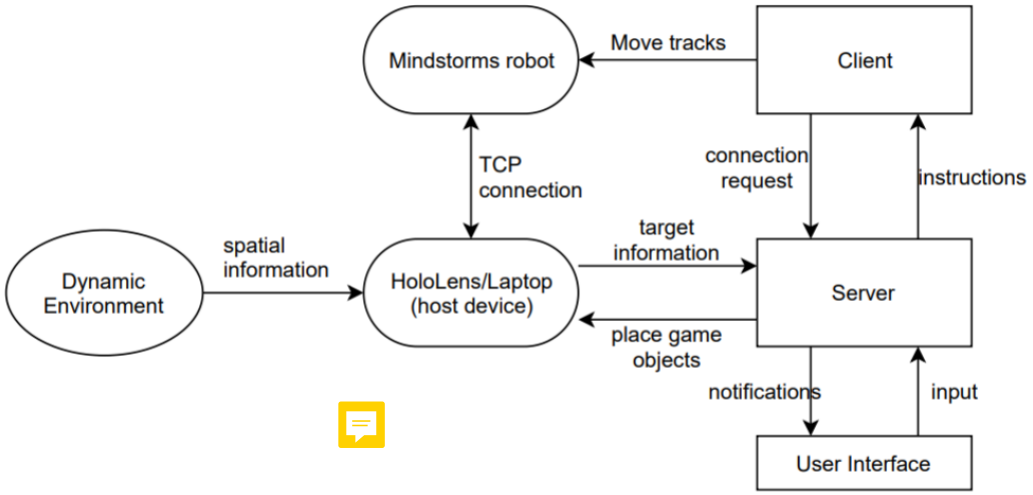

Figure 3.1: Squared shapes represent software components, oval shapes host devices. As we can see, both the robot as the HoloLens/laptop are connected to each other. This allows the client to connect to the server and the server to send back instructions that it gets from the user input. The HoloLens provides the server with spatial information as to allow to find the target robot in the environment to correctly navigate virtual game objects.

3.1.1 Server

The server can be run on both the HoloLens and our laptop depending on whether we're debugging or deploying the application. The HoloLens makes Mixed Reality possible by enriching the wearer's visual perception of the environment with virtual information. It allows for user input such as voice commands and hosts the application's server who can directly react to the user input by sending data packets to the clients. The user —and by consequence the HoloLens— is the source of all network communications with the client(s). For the server running on our laptop, the user can command the connected clients by way of the arrow buttons of the keyboard.

3.1.2 Client

The client is hosted on the Mindstorms robot. The EV3 brick is the robot's brain and has less computation abilities compared to the HoloLens. This does not matter since the client is not supposed to do a lot of computation in the application only being required to poll its input endpoint for new data packets of the server. The role of a client is to be a requester of a service where the server functions as the provider of that service. One of the application's requirements states that object tracking should be possible even in a changing, dynamic environment. The brick is connected with tracks as to allow for basic movement as to show that even when the client is dynamically moving, we can show that object tracking is still achieved.

3.2 Navigation Framework

The focus of this thesis was to create a framework for the navigation of game objects inside a dynamically changing environment.

3.3 System Design

3.4 Towards a Prototype Application

4 Implementation

This chapter discusses the implementation of our navigation framework and of the Mixed Reality application using said framework. We will also discuss all software tools and technologies used for this project and provide the architecture of our final application as a starting point to discuss all system components in more detail.

4.1 Overview

Developing an application with the help of our framework was necessary to show its capabilities of navigating virtual objects inside a dynamically changing environment. In the process, we had to use various technologies for both the framework and its subsequent application.

4.1.1 Hardware

Our system uses an internet network for its components to communicate with each other. More specifically, the communication between the system client(s) and the server. Establishing such network required using a D-Link[1] router. The client program runs inside a Lego Mindstorms EV3 brick[2] with

[1] https://en.wikipedia.org/wiki/D-Link

[2] https://en.wikipedia.org/wiki/Lego_Mindstorms_EV3

corresponding tracks to give it displacement capabilities as the framework should be able to overlay its game objects even over moving real-life objects. The server program runs inside a desktop computer when using the Unity editor for development purposes and the Microsoft HoloLens for a real deployment. The Microsoft HoloLens is one of the first commercially available head-mounted devices on the market.

4.1.2 Programming Languages

Since the EV3 brick only provided a drag-and-drop interface for coding, this prompted us to replace the robot's firmware as to allow for more intensive coding. This resulted in a client written with the Java programming language. Our server program was written with different versions of the C# language depending on whether it was hosted on a desktop computer or the HoloLens respectively. This required developing a program compatible for both platforms which brought some additional difficulties.

4.1.3 Toolkits

Unity[3] is a cross-platform game engine that can be used to implement — among others— augmented reality and virtual reality games. The first iteration of the engine was launched in 2005 and its main purpose was to make game development more accessible to the public. As of 2018 it has been extended to over 25 platforms[3]. Its popularity has prompted the surge of different toolkits for Augmented Reality. These toolkits are collections of software development tools that provide functionalities for the development of applications, here in the context of Augmented Reality and more specifically Mixed Reality.

Vuforia Toolkit

The navigation framework's main requirement is the detection of real-life objects in the environment that need tracking and overlaying virtual objects on top of them. In achieving this end, the Vuforia[4] engine's software development kit helped us greatly. It provided us with the possibility of scanning the real-life objects to track in the Unity engine and its detection software which uses deep learning made the application work very smoothly.

[3] `https://en.wikipedia.org/wiki/Unity_(game_engine)#cite_note-Easier-4`
[4] `https://developer.vuforia.com/`

Mixed Reality Toolkit

This software development toolkit was developed by Microsoft[5] to be used on its Mixed Reality projects such as the HoloLens. It offers functionalities in terms of user input by hand gestures or speech, diagnostic tools, spatial awareness and much more. The toolkit was mostly used to implement voice commands on the robot and for activating spatial awareness used for indicating waypoints in the environment for the user robot to follow.

4.2 System Architecture

Our final application was developed using our navigation framework and consists of one or more possible clients and the server that can be hosted on different devices for debugging or deployment purposes. Though we hinted at the possibility to go for a peer-to-peer architecture in the introduction, we ultimately went for a client-server architecture.

4.2.1 Client-Server Model

Within this application structure, there exists a clear role between the providers of a service —called the servers— and the requesters of a service —called the clients— with both communicating using an established computer network and hosted on separate devices.

Benefits

The server-client model boosts characteristics that make it suitable for the prototype Mixed Reality application we would like to develop. Especially given the hardware at our disposal and the scale of deployment. Its centralized structure makes it simpler to implement compared to a peer-to-peer system where the distinction between service provider and requester is not clear. The network communication between components is much simpler and distinct. Since the scale of deployment is small with only few client devices requesting services and only one device having true computation capabilities —the HoloLens— there should be no bottlenecks for the system.

[5] https://www.microsoft.com/en-us/

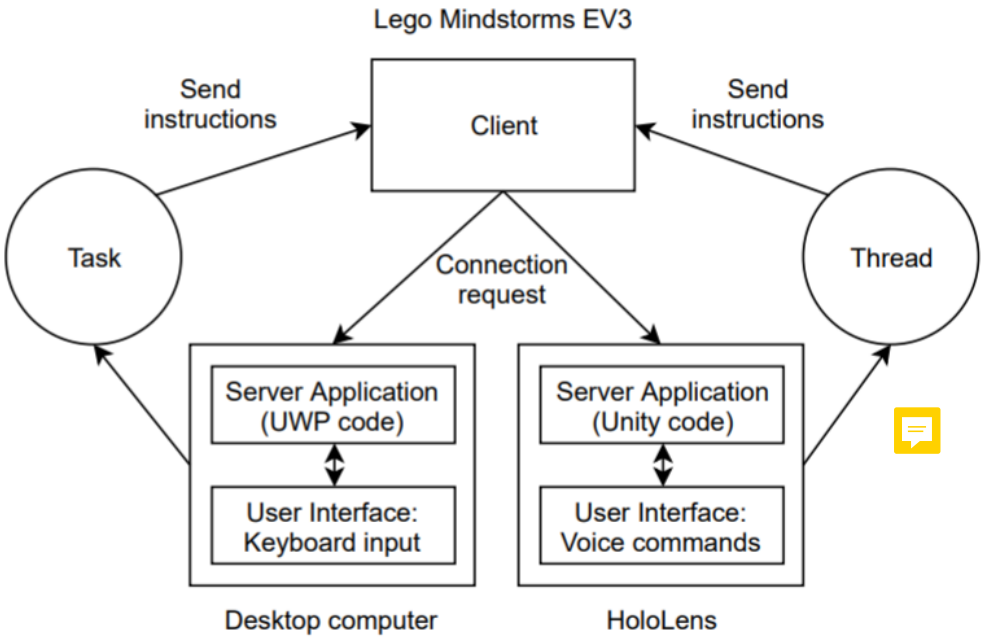

Figure 4.1: The application's architecture consists of a client running on the EV3 brick sending a connection request to the server hosted on a desktop computer or the HoloLens respectively. Depending on the host machine, different code strips will be used with the former using threads and the latter tasks. Both threads and tasks are created for every new client connection and manages passing instructions from the server to the client.

Flaws

Having a client-server architecture makes it harder to scale up the application when adding additional clients and servers. For every new client added we ought to reason if the servers in the system could handle this additional workload. Having clear system roles also adds critical components that would slow down or even stop the application if they would break down. In terms of reliability, the server-client model is not the best choice. Still, the model is sufficient for the design of a prototype Mixed Reality application.

Communication Flow

Our implementation makes use of network sockets for sending or receiving data within a computer network. To do so, the client-server connection must first be established. This is why the server listens for and accepts incoming client connection requests in the network. When a client is accepted, it will

associate a thread to a client and keep a reference to it as to allow the server to send data packets specifically to a certain client.
The key insight is that is the client who sends the connection request to the server but it is the server who sends all data to the client. The client continually checks its input socket as to find out which new instructions it has from the server.

4.2.2 Peer-to-Peer Model

Opting for a peer-to-peer architecture is advantageous if we would like to scale up our application. Given the current hardware at our disposal, this is very hard to implement and test out in the field.
Implementing a communication system in a decentralized manner is much more complex to realise than a client-server structure and only relevant if there are risks of tight bottlenecks between the clients and the server. For the given deployment scale of the application, it was much more appropriate to follow the client-server model.

4.3 Networking

Realising a wireless communication between our client and server required setting up our own private internet network using a router device. Configuring the router boiled down to connecting it to a modem for cable internet connection, connecting a computer to the established network and doing some further router configuration on that computer such as setting the password for the network and the administrator. Our network uses the internet protocol suite and associates a network (IP) address to all connected machines as to allow identification within the network. This would also allow all connected devices to forward data packets to each other.

4.4 Client

This system component connects to the server given its IP address in the network and continuously polls its input stream for new instructions. The server keeps a list of all connected clients and associates a thread to every one of them. Each thread handles the communication to exactly one client. This allows more than one client to receive instructions by the server.

4.4.1 Lego Mindstorms Robot

As moving ground unit, we used a programmable robot based on Lego building blocks with the Lego Mindstorms EV3 Intelligent Brick as its brain. It allows to upload code and functions as the robot's controller and provides its power to run the motors and sensors. There was however a problem with its firmware. Lego only offered the Ev3 Programmer App[6] to be used on the Mindstorms robot to reprogram it. This only consisted of a primitive drag and drop programming interface and it was too limited to allow for some real coding.

4.4.2 leJOS Firmware

The programmable brick's firmware limitations prompted us to use leJOS[7] as a replacement. It includes a Java virtual machine allowing us to program our robot with the Java programming language. The installation involved uploading it on an SD card and inserting it in the EV3 brick. Starting the brick with the card would configure it to use the replacement firmware. Writing the Java code involved using the Java libraries for the leJOS firmware allowing us to have access to the brick's motors and sensors. By simply using the brick's IP address, we could execute our code and upload it to the brick.

4.4.3 Stream Socket

As to allow the it to receive data over the established computer network, the client is making use of stream sockets. This type of socket provides a connection-oriented and unique flow of data without record boundaries. They are implemented on top of TCP used by the router so that applications can run across any networks using the internet protocol suite[8]. The client itself only sends a client connection request to the server on the port it is listening to. The client afterwards will not send anything to the server with its socket but merely polls its input stream continuously to see if there are new instructions from the server.

4.5 Server

This piece of software will be running on both the Unity editor for debugging and the HoloLens for production. Its role would be to listen for incoming

[6] `https://www.lego.com/en-us/Mindstorms/apps/ev3-programmer-app`
[7] `http://www.lejos.org/`
[8] `https://en.wikipedia.org/wiki/Internet_protocol_suite`

clients trying to connect and then forward navigation instructions to the appropriate connected client. These navigation instructions would be given by voice commands when using the HoloLens and by way of the desktop keyboard when using the Unity editor.

4.5.1 Client-Server Communication

In our application, it is the client that starts the communication with the server by using one of its open ports. The server has a TCP listener that waits on a certain port for incoming client requests. When it receives a new client request, it starts a new thread (or task) that handles all communication to it. It also keeps a client reference in a list including its IP address on the used computer network. This way, we can search up a client, identify it on the network using its address and send specific data to it. This is all done by using the client's streamsocket where bytes of data can be send through.

4.5.2 Universal Windows Platform

Programming a server that would work on both the Unity editor and the HoloLens proved difficult because of library incompatibilities. The reason for this is that all Microsoft products since 2016 compile their code using the Universal Windows Platform[9]. It was developed as part of the Windows 10 operating system and its main goal was to allow the development of universal apps that would run on all Microsoft products such as the Xbox One and HoloLens without the need to be re-written for each product. To do so, they've settled on a version of C# that is different from the one used by the Unity game engine. This means that we have to use 'different' code for our server depending on whether we're using the Unity editor or the Microsoft HoloLens.

Preprocessor Directives

The problem with these different libraries is that the compiler has no idea as to know when the written code is destined for the Unity editor and when it's destined for the HoloLens. To alleviate this, one must use preprocessor directives. This is to give indications to the compiler which parts of code to ignore when using the Unity editor and the HoloLens respectively. This meant our developed server consists of two different code sections for different platforms.

[9] `https://en.wikipedia.org/wiki/Universal_Windows_Platform#Compatibility`

Multithreading

As to allow multiple clients to use the server, the use of threads was necessary. Failing to do so would result in a server that would only manage one client at a time. At the time of writing however, the Universal Windows Platform used by the HoloLens does not support threads instead only allowing us to use tasks. The use of them was combined with preprocessor directives as to give indications to the compiler which code to ignore when using the Unity editor and the HoloLens respectively.
Threads can be seen as workers to be used to fulfil some amount of work while tasks are just promises to complete a certain amount of work in the future eventually. The problem is that a task's interface is different from the thread's interface requiring some code rewriting.

4.6 Navigation Framework

Allowing the proper airborne navigation of game objects was the main focus of the thesis. We now discuss the part of the application that would allow for game objects to properly hover over a certain target object. It allows us to track a predefined object even when applied in a dynamically changing environment. To show the successful object tracking, a virtual object hovers on top of it.

4.6.1 Vuforia

Enhancing real-life objects with computer-generated perceptual information requires extensive object recognition and tracking. For this purpose, we made use of the Vuforia[10] engine capable of distinguishing objects purely by their visual appearance. On its developer portal, it provides all tools and SDK's required to achieve this purpose. It also provided valuable documentation for the proper configuration of our game engine and registering our targets for tracking.

Deep Learning

The Vuforia engine makes use of artificial intelligence and in particular deep learning as to allow object recognition and tracking in a dynamic environment. It works by creating artificial neural networks consisting of multiple layers to progressively extract higher level features from raw input. In other words, every higher layer tries to identify more advanced diagnostics of the

[10] `https://developer.vuforia.com/`

given data compared to the layer directly under it. In our context, the neural network would first try to identify the edges of the object and then try to find more advanced characteristics such as its shape, colour, etc. This process makes using Augmented Reality easy and fast.

Target Requirements

Vuforia allows us to track objects of different shapes such as simple two-dimensional images and even three-dimensional real-life objects such as a model car or an action figure. A limitation here is that the object in question may not be too big. Our Mindstorms robot in this regard is roughly 30 centimetres in length and 12 centimetres in width. Its size satisfies our requirements.

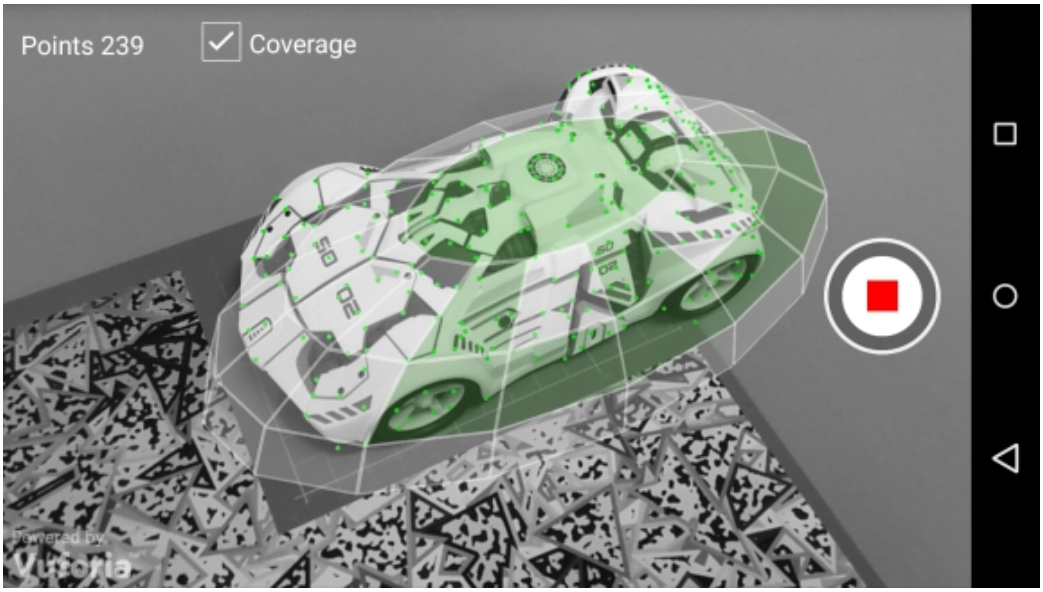

Figure 4.2: Using the Object Scanner app for Android, we are able to generate a scan of the wanted target object. The model car lies on a sheet of paper called the Object Scanning target which the app uses as an anchor point for the scanning process.

Object Scan

For the Vuforia engine to be able to track the real-life robot, we had to provide it with a three-dimensional scan. Scanning the robot was done using

the Vuforia Object Scanner app[11] for Android[12]. It gives us real-time visual feedback on the scan progress and allows us to assess the scan quality for given target. Our target scan can be exported as an Object Data file which includes required data to define the target for recognition and tracking purposes.

4.6.2 HoloToolkit

Microsoft created its own bundle of software development tools to help in the creation of MR applications in Unity. These building blocks would provide help for implementing voice commands, spatial understanding and diagnostics. Since there were alot of versions for given toolkit that differed in the functionalities and scripts on offer, finding the right documentation to use the Toolkit proved at times difficult.

4.6.3 Unity Configuration

Using given toolkits required some additional configuration of Unity. The HoloToolkit had to be fetched from Github[13], placed into the project's assets folder and to which Unity automatically started importing its assets to the project. These assets range from sample scenes, the scripts to prefabs. We then had to configure the toolkit to our Unity game scene and enable the support for Augmented Reality.
The Vuforia toolkit just had to be imported using the Unity installer for the Unity version that was used for the project. It was required to enable Vuforia support and add an Augmented Reality camera to the scene as to include the Vuforia engine. To activate it, one had to get a free developer key and load up a target scan for the real-life object that had to be identified.

4.7 Robot Navigation

Navigation of game objects using our framework should allow overlaying virtual content over a targeted real-life object. However, it must be robust enough as to still do a correct overlay even when the robot is moving. The EV3 brick is the brain of the robot and has 8 ports that can be connected to small motor tracks. The brick serves as its power base and controls these tracks as to provides some movement capabilities.

[11] `https://developer.vuforia.com/downloads/tool`
[12] `https://www.android.com/intl/fr_fr/`
[13] `https://github.com/`

4.7.1 Desktop Input

Both the HoloLens and the Unity editor have different interfaces that allow the user to control the robot. The advantage of the desktop interface is that it was perfect for debugging the robot when sending displacement instructions with the network.

Keyboard Input

The most straightforward way to control the robot when using the desktop as host device was by way of its keyboard input. With the keyboard's arrow buttons the user would schedule a new instruction for the robot. An internal variable is constantly changed when pressing these buttons and its current value or 'instruction' is passed through the network to the client for every frame update. This way, the client is provided by a constant bytestream of instructions that constantly tells it what to do.

4.7.2 HoloLens Input

Providing a way for the user to guide the robot in a Mixed Reality application gives the developer some creative freedom since traditional input methods such as a keyboard or joystick are non applicable. We can only rely on such methods as voice commands, eye gaze and three-dimensional interface buttons clickable by finger gestures. The use of these were made furthermore easier by the Mixed Reality Toolkit[14] developed by Microsoft[5]. It provided scripts and resources for the development of these input techniques.

Voice Commands

Using the previously mentioned toolkit, it becomes possible to develop a keyword manager script for the HoloLens that would listen and wait for the user to say out loud one of its keywords. These keywords are short and distinct and would trigger certain actions such as making the robot move forward or backward or make it complete certain paths such as a full rotation on itself.

Buttons

The toolkit offers assets of three-dimensional buttons for the user to use. By linking the click events with the server application, it would send new instructions to the client on every new button click. Using the interface

[14] `https://github.com/microsoft/MixedRealityToolkit-Unity`

buttons the user is capable of making the robot move in all directions (left, right, front and back).

5 Evaluation and Reflection

In this chapter, we evaluate our final framework and subsequent application and the ways in which we tested its correct working. We also give a brief discussion of the most persistent difficulties during development.

5.1 Testing

Making sure our program works correctly on all devices demanded some extensive debugging and testing. We used different tools for this end.

5.1.1 Client Side

Testing the client focused on its proper connection with the server and on properly receiving data packets through our network. Its written code contains a stream socket used as an endpoint for receiving packets. The socket could also be used to send bytes to the server but this is unnecessary since the communication flow of our application goes from the server to the client.

Unity editor for Debugging

Since the HoloLens was much slower in executing the server compared to the Unity editor, it was easier to use the latter of the two to send instructions

through the network. By using the host device's keyboard, we could send out instructions to the client with the socket.

Display Screen

To analyse which instructions would be properly received by our EV3 brick, we would use its display screen to print out all messages it would receive by the stream socket. Together with the Unity editor's keyboard input, we could precisely calibrate the robot's reactions on the flow of bytes as to make it more responsive.

5.1.2 Server Side

Testing the server proved easier for the Unity editor compared to the HoloLens. This is because executing our server program on the Unity editor would take only seconds by activating the play mode. Proper execution on the HoloLens would take much longer.

Unity Editor Testing

By running the Unity editor in play mode, the server would be executed in a matter of seconds. Evaluating the server happened in two steps.
The first step involved using the Packet Sender[1] app for sending and receiving network packets using the Transfer Control Protocol among others. We first had to specify the IP address and port number of the connected device on the network to which a test packet should be send. If no error message would occur after sending the package, we could be sure that the device was open for connection.
The user could give input for the server using the editor by pressing the arrow buttons on the desktop keyboard. The Unity console would print out all messages that would be past through the stream to the client

HoloLens Testing

Each time we would like to test improvements on the UWP server we had make a new build, compile it on our Microsoft Visual Studio[2] Integrated Developer Environment and then upload it on the HoloLens. This whole process would easily take a dozen minutes to complete.

[1] `https://packetsender.com/`
[2] `https://en.wikipedia.org/wiki/Microsoft_Visual_Studio`

To speed up deployment, we used the HoloLens Emulator[3] to test out client connection. In the first iterations of the UWP server, only a continuous stream of 1's would be sent through just to test the basic workings of the network socket. We used the emulator to test this basic functionality.
For further interface testing, we had no choice but to use the HoloLens. The voice commands and interface buttons would be calibrated on the robot's movements.

Multiple Clients

To allow for more than one client to connect to the server and request its services, we had to use threads for the Unity server and tasks for the UWP server respectively. In both cases, the server would keep a list of references to the connected clients to allow for client-specific packets to be sent.

5.2 Difficulties

Development was plagued by persistent deployment issues with the HoloLens and library incompatibilities when coding. The HoloLens is one of the first commercially available head-mounted devices on the market which means such technical issues could be expected. Integrating Vuforia with the HoloLens deployment proved problematic. It worked well on the Unity editor but crashed when applied on the HoloLens requiring further configurations.
The compatibility issues between Unity and the Universal Windows Platform are due to the different C# libraries being used for each respective platform. This made the server code much more complex since it was divided into two different code snippets for the Unity editor and the HoloLens respectively.

[3] https://docs.microsoft.com/en-us/windows/mixed-reality/using-the-hololens-emulator

6 Conclusion

6.1 Conclusion

We have worked in accordance with the Design Science Research Methodology for information systems[16]. In the introduction, we identified the problem for our research effort and defined our objectives for success. The literature study allowed us to gather concepts and techniques for Mixed Reality as a baseline to define our application's requirements. We then gave a high-level overview of our application followed by a more low-level implementation section. The whole thesis revolved around developing a framework that would allow us to create an application that could mimic Microsoft's demo[1] on the HoloLens' cross-space capabilities. It would allow to overlay computer-generated objects over a real-life target. We had to provide the user with a way to steer that object to show that our framework is also able to keep virtual objects correctly placed on the target.

6.2 Future Work

The developed framework's features was shown by way of a small Mixed Reality application. It used a client-server architecture with the HoloLens

[1] `https://www.youtube.com/watch?v=xnrHFV34PfM`

acting as server and the mobile robot as client. Using such architecture proved advantageous for a small application but it's really difficult to scale up. Also, if we would like to implement a multiplayer mode in the game —as was suggested in the related work— it would be easier to go for a peer-to-peer model where every player would communicate with neighbouring players. The feature of generating waypoints in the environment and making the robot follow them also requires additional work to be done.

A

Your Appendix

Bibliography

[1] Rahib H Abiyev, Nurullah Akkaya, Ersin Aytac, Irfan Günsel, and Ahmet Çağman. Improved path-finding algorithm for robot soccers. *Journal of Automation and Control Engineering*, 3(5), 2015.

[2] Yen-Ling Chen and Pei-Chien Hung. Intuitive augmented reality navigation system design–implementation by next-gene20 project. In *International Conference on the Association for Computer-Aided Architectural Design Research in Asia*, pages 351–360, 2009.

[3] Andrew I Comport, Éric Marchand, and François Chaumette. A real-time tracker for markerless augmented reality. page 36, 2003.

[4] Emmanuel Dubois, Philip Gray, and Laurence Nigay. *The engineering of mixed reality systems*. Springer Science & Business Media, 2009.

[5] Gabriel Evans, Jack Miller, Mariangely Iglesias Pena, Anastacia MacAllister, and Eliot Winer. Evaluating the microsoft hololens through an augmented reality assembly application. 10197:101970V, 2017.

[6] Thomas A Funkhouser. Ring: A client-server system for multi-user virtual environments. In *Proceedings of the 1995 symposium on Interactive 3D graphics*, pages 85–ff. ACM, 1995.

[7] Wolfgang Hoenig, Christina Milanes, Lisa Scaria, Thai Phan, Mark Bolas, and Nora Ayanian. Mixed reality for robotics. pages 5382–5387, 2015.

[8] Charles E Hughes, Eileen Smith, CB Stapleton, and Darin E Hughes. Augmenting museum experiences with mixed reality. In *Proceedings of KSCE 2004*, pages 22–24, 2004.

[9] Joaquín Keller and Gwendal Simon. Toward a peer-to-peer shared virtual reality. In *null*, page 695. IEEE, 2002.

[10] Ernst Kruijff, J Edward Swan, and Steven Feiner. Perceptual issues in augmented reality revisited. In *Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on*, pages 3–12. IEEE, 2010.

[11] Mikko Kytö, Barrett Ens, Thammathip Piumsomboon, Gun A Lee, and Mark Billinghurst. Pinpointing: Precise head-and eye-based target selection for augmented reality. page 81, 2018.

[12] Robert Gabriel Lupu and Florina Ungureanu. A survey of eye tracking methods and applications. *Buletinul Institutului Politehnic din Iasi, Automatic Control and Computer Science Section*, 3:72–86, 2013.

[13] Daniel Mestre, Philippe Fuchs, A Berthoz, and JL Vercher. Immersion et présence. *Le traité de la réalité virtuelle. Paris: Ecole des Mines de Paris*, pages 309–38, 2006.

[14] Paul Milgram and Fumio Kishino. A taxonomy of mixed reality visual displays. *IEICE TRANSACTIONS on Information and Systems*, 77(12):1321–1329, 1994.

[15] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011.

[16] Ken Peffers, Tuure Tuunanen, Marcus A Rothenberger, and Samir Chatterjee. A design science research methodology for information systems research. *Journal of management information systems*, 24(3):45–77, 2007.

[17] David R Walton and Anthony Steed. Accurate real-time occlusion for mixed reality. page 11, 2017.