Universidade de Aveiro
2021

Rafael
Magano Maio

Augmented Reality Serious Game for Intelligent Wheelchair

Jogo Sério de Realidade Aumentada para Cadeira de Rodas Inteligente

Rafael Magano Maio

Augmented Reality Serious Game for Intelligent Wheelchair

Jogo Sério de Realidade Aumentada para Cadeira de Rodas Inteligente

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requesitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor José Nuno Panelas Nunes Lau, Professor associado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro e do Doutor Paulo Miguel de Jesus Dias, Professoar auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

Esta dissertação foi apoiada pelo projeto IntellWheels 2.0 - POCI-01-0247-FEDER-39898.

o júri / the jury

presidente / president

Professor Doutor Joaquim João Estrela Ribeiro Silvestre Madeira
Professor Auxiliar, Universidade de Aveiro

vogais / examiners committee

Doutor Miguel Correia Melo
Investigador Auxiliar, Universidade de Trás-Os-Montes e Alto Douro

Professor Doutor Paulo Miguel de Jesus Dias
Professor Auxiliar, Universidade de Aveiro

Agradecimentos

Quero agradecer ao meu orientador Nuno Lau, ao meu coorientador Paulo Dias e ao colaborador João Alves por todo o apoio e conhecimento que me deram, assim como o acompanhamento regular durante a realização deste trabalho. Quero também agradecer a todas as pessoas que arranjaram disponibilidade para participar nos testes de usabilidade realizados. Por fim, gostava de agradecer à minha família e aos meus amigos, que me foram dando motivação ao longo deste ano.

Palavras-chave

Realidade Aumentada, *Tracking*, Navegação, Dispositivos Móveis, Interação Humano-Robô, Cadeira de Rodas Robótica

Resumo

A qualidade de vida das pessoas com mobilidade reduzida é um assunto importante e todos os dias a tecnologia também evolui no sentido de os ajudar. As cadeiras de rodas inteligentes são um resultado desta evolução, no entanto, a adaptação para usar novas tecnologias nem sempre é fácil. Os jogos sérios podem tornar o processo de aquisição de competências para utilizar uma cadeira de rodas robótica num método divertido e imersivo. A realidade aumentada mostra-se também um auxílio na aprendizagem e no treino da utilização de várias novas tecnologias, incluindo a robótica. Permitindo a visualização e interação com objetos virtuais no ambiente em que a cadeira de rodas inteligente está inserida, a realidade aumentada possibilita a criação de um jogo sério. Consequentemente, nesta dissertação desenvolvemos um jogo sério de realidade aumentada, tirando vantagem das capacidades de tracking do ARCore, que foram estudadas e avaliadas. Foi elaborado uma ferramenta de configuração do ambiente de realidade aumentada onde o jogo sério é realizado. O jogo tem o suporte de uma ferramenta de visualização que, no fim, permite visualizar a performance do utente e os seus erros. Uma experiência para comparar os métodos de configuração e avaliar o jogo sério foi realizada com a colaboração de um conjunto de utilizadores. Esta experiência permitiu verificar as vantagens que os métodos de configuração utilizando o computador podiam trazer em relação aos que usam o telemóvel e vice-versa, como também verificar que o jogo sério de realidade aumentada podia ser uma mais valia para auxiliar o controlo de cadeiras de rodas robóticas. Para além do desenvolvimento do jogo sério e das ferramentas necessárias para que este seja possível, foi também desenvolvida uma biblioteca que permite transformar jogos Unity em jogos de realidade aumentada. Esta biblioteca permite ultrapassar algumas das limitações impostas pela ferramenta de configuração.

Keywords

Augmented Reality, Tracking, Navigation, Mobile Devices, Human-Robot Interaction, Robotic Wheelchair

Abstract

The quality of life of people with reduced mobility is an important matter and every day technology also evolves in order to help them. Intelligent wheelchairs are a result of this evolution, however, adapting to use new technologies is not always easy. Serious games can make the process of acquiring the skills to use a robotic wheelchair a fun and immersive method. Augmented reality is also a support tool in learning and training how to use several new technologies, including robotics. Allowing the visualization and interaction with virtual objects in the environment which the intelligent wheelchair is inserted, augmented reality make the creation of a serious game possible. Consequently, in this dissertation we developed an augmented reality serious game, taking advantage of ARCore's tracking capabilities, which were studied and evaluated. A tool for configuring the augmented reality environment where the serious game is played has been developed. The game is supported by a visualization tool that, at the end, allows the visualization of the user's performance and his errors. An experiment to compare the configuration methods and evaluate the serious game was carried out with the collaboration of a set of participants. This experience allowed us to verify the advantages that the configuration methods using the desktop PC could bring over those using the smartphone and vice versa, as well as verifying that the augmented reality serious game could be an asset to help control robotic wheelchairs. In addition to the development of the serious game and the tools necessary to make it possible, we also developed a library which allows transforming Unity games into augmented reality games. This library allows to overcome some of the limitations imposed by the configuration tool.

Contents

List of Figures

List of Tables

Listings

Acronyms

AR Augmented Reality. i, iv, x, 1–16, 19, 21, 22, 43, 51, 54–56, 58–61, 63, 68–71, 73, 79–81, 84, 85, 115

ARG Augmented Reality Game. 9, 68, 70, 85, 114

ARSG Augmented Reality Serious Game. 2, 51, 63, 64, 79–81, 83–85, 92

AV Augmented Virtuality. 4

COM Concurrent Odometry and Mapping. 7

CPU Central Processing Unit. 21

CSV Comma-separated values. 26

FoV Field of View. 13, 14, 16, 24, 25, 63, 65

GCP Google Cloud Platform. 21–23, 54, 56, 60, 63

GPS Global Positioning System. 9

HMD Head Mounted Display. 5, 8, 13, 15, 16

HRI Human-Robot Interaction. 11–14, 84, 85

IMU Inertial Measurement Unit. 7, 18, 19

IT Information Technology. 2

LHS Left-Handed Coordinate System. 12, 23, 27

MR Mixed Reality. 4

PC Personal Computer. v, 5, 8, 12, 51, 52, 54–58, 61, 62, 65–67, 74–76, 79, 84, 85, 98–100, 117, 120, 121

PDA Personal Digital Assitant. 5, 6, 8

PNG Portable Network Graphics. 23

RGB Red Green Blue. 20

RHS Right-Handed Coordinate System. 12

RVA Realidade Virtual e Aumentada. 54, 55

SDK Software Development Kit. 2, 5–8, 23, 55, 70, 84

SLAM Simultaneous Localization and Mapping. 19

TCP/IP Transmission Control Protocol/Internet Protocol. 65

UI User Interface. iii, v, vi, 54–57, 60–64, 72, 94, 95, 97–100

VIA Variable Impedance Actuated. 13

VIO Visual Inertial Odometry. 6, 7

VR Virtual Reality. 4, 5

Chapter 1

Introduction

1.1 Motivation

Robotic wheelchairs usage is increasingly growing, complementing the ordinary wheelchairs, thanks to all the new features and comfort that these can bring to their users. Figure 1.1, from the *Wheelchair Market Size, Share & Trends Analysis Report*[1], shows that both motorized and manual wheelchair markets are increasing over the years and it is expected to continue to rise, estimating to expand to 8% from 2021 to 2028. However, there are few people with a good experience interacting with robots and this lack of experience makes the initial training difficult for most wheelchair users.

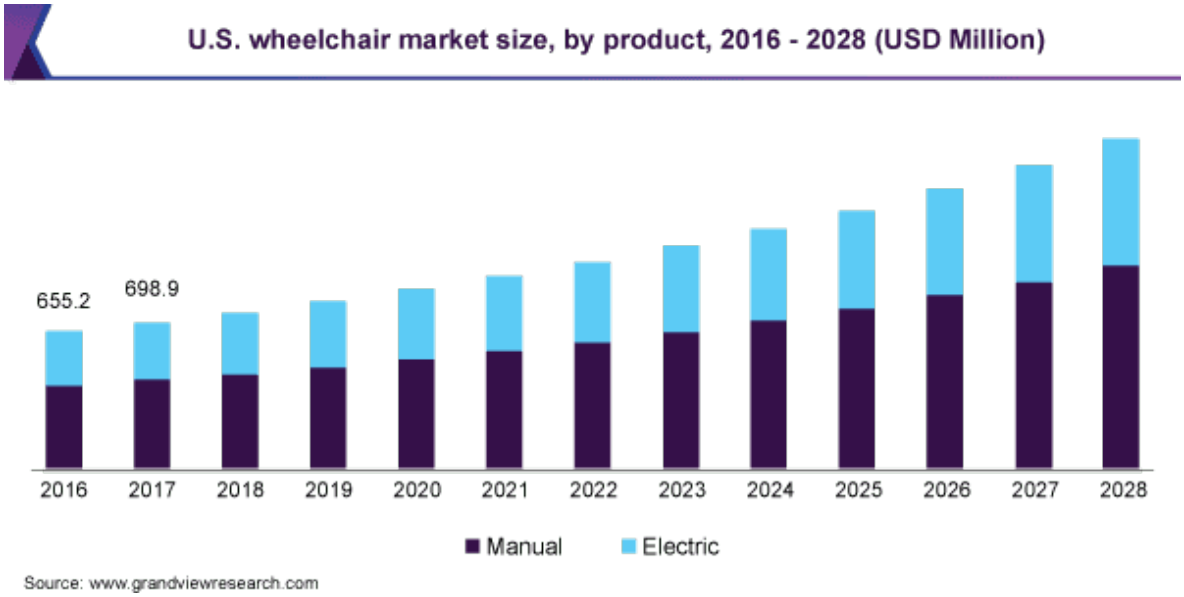


Figure 1.1: United States wheelchair market size, in USD Millions for manual and electric wheelchairs[1].

AR is a technology that allows "augmentation" of the real world, through digital information generated and presented to the user. This "augmentation of reality" allows the user to increase the information obtained by their sensory system, which can help to perform tasks in the real world

[1]https://www.grandviewresearch.com/industry-analysis/wheelchair-market

Billinghurst et al. (2014). Hereupon, AR provides a possible solution to this problem. AR technologies have many advantages and are commonly used as a tool to help the interaction of humans with robots and other devices, making this interaction more visual. Hence, all the user actions become more clear to him, it will be easier to know how to adapt and what to do next. Besides this, AR increases the user engagement and interaction, providing richer experiences Orús et al. (2021).

While in standard games the primary goal is entertainment, in serious games the primary purpose is learning, informing, adapting, among other, this means that serious games use the power of games and their entertainment to assist the application objective. This way, people can enjoy the time spent in training and getting better on that concrete activity. Activities with different purposes can be supported by serious games, from military to health and exercise therapy Laamarti et al. (2014). The topic of this study is the latter one, health and exercise therapy, applied for motor disabled people and AR can be an interesting technology to support a serious game in this field due to the physical movement that entails.

1.2 Goals

Our goal is to create an AR tool for serious games to support health and exercise therapy for people with motor disability, as well as to study strategies for creating them. Using this tool, we intend to create two Augmented Reality Serious Game (ARSG) for wheelchair users, one game defined by static virtual objects around the real world and the second one characterized by a virtual moving object. It is intended to compare and investigate which might be more enjoyable for learning and testing the control of robotic wheelchairs. In addition to developing these games as an intuitive and easy learning path to achieve all the knowledge that motor disabled people need to have to drive this type of wheelchairs, we aim to build a tool for creating new AR games in a simple way.

To accomplish this, we studied the major difficulties in using a robotic wheelchair and which AR clues can be propitious to help users to overcome these obstacles. Besides this, we investigated which AR technology would be suitable for our application and why it is more convenient than other AR technologies, as well as its limitations.

The games aim to be identical, but with distinct goals, one containing a set of static virtual objects that represent the path, for the user controlling the robotic wheelchair, to follow. The second one consists of following a path outlined by a moving object that is supposed to be chased.

The application also plans to have a versatile configuration tool for the AR environment, so it can be used in different scenarios with distinctive objectives. Similarly to the ARSGs, since this tool is intended to be used by common people, without any specific knowledge in Information Technology (IT) and in AR (with the major difference that it is supposed to be used mostly by people without reduced mobility), it has to be designed to be intuitive and easy for the common user to operate.

1.3 Document Structure

This document is divided into 6 chapters. After this short introduction, chapter 2 describe AR concepts, associated technologies, some related work to this study and import decisions that are made based on these. In chapter 3, we delve into the chosen Software Development Kit (SDK) technology and into its relevant features for our use cases. We built an experiment to evaluate the performance of ARCore motion tracking technology. We present the results gathered from the experiment, ending this evaluation section with the discussion raised from the obtained results. Then, in chapter 4, we explain the developed application, its concept and architecture, detailing the technical implementation

of each application module and completing this chapter with the created library for effortlessly adapt other 3D applications and games into AR applications. Proceeding to chapter 5, where we present the conducted usability tests, that were developed to verify advantages and limitations of the created application, as well as to study and to evaluate the developed configuration tool and serious games. Finally, in the last chapter, limitations, conclusions and future work suggestions are discussed.

Chapter 2

Augmented Reality

In this chapter, we intend to introduce what is AR and the major technologies that are used in this field. After studying those technologies, analyzing their pros and cons and comparing them, one stood out as preferable. Besides that, this chapter aims to describe the functionalities and characteristics of the wheelchair used and finally, study the related work merging the area of AR with robotics and serious gaming, and specifically AR applied for people with reduced mobility that use a wheelchair on daily basis.

AR was first seen in 1992 when Louis Rosenberg[1] invented a system called *Virtual Fixtures* which enabled the overlay of sensory information on a workspace to improve human productivity. Thanks to that, nowadays, AR is used in many different areas. Milgram (1994) categorizes six areas where AR is practiced: Personal information systems, industrial and military applications, medical field, entertainment, for the office and, lastly, education and training. But what is AR?

Billinghurst et al. (2014) defines AR as a variation of Virtual Reality (VR). VR technologies completely immerse a user inside a virtual environment, so the user is not able to see the real world around him. In contrast, AR allows the user to see the real world, with virtual objects composited with the real world. Therefore, AR supplements reality, rather than completely replacing it.

To simplify these reality concepts and their differences, Milgram (1994) presented the "virtuality continuum", which can be illustrated by figure 2.1. At the continuum ends, we have the real environments and the virtual environments, as they are opposite environments, the first one without any virtual object in it and the latter one, as the opposite end, not having any sample of the real environment, being completely immersive. Between these edges, the continuum is composed by the Mixed Reality (MR) which can be described as the hybrid environment, where real and virtual environments are merged. The AR is comprehended inside the MR, together with the Augmented Virtuality (AV).

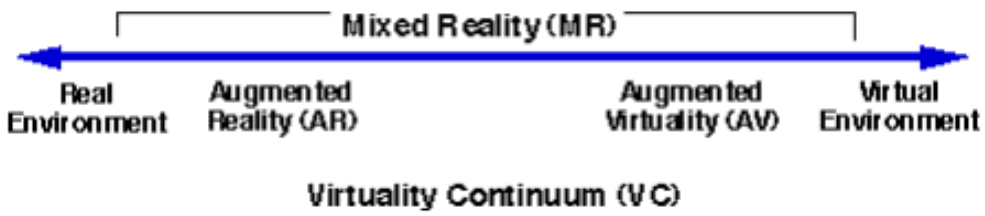


Figure 2.1: Simplified representation of the "Virtuality continuum". (Milgram, 1994)

[1] https://www.interaction-design.org/literature/article/augmented-reality-the-past-the-present-and-the-future

Silva et al. (2003) identifies the three components present in all AR applications. Those components are:

- Scene generator, software or device responsible for rendering the scene.
- Tracking System, the objects in the real and virtual worlds must be properly aligned with respect to each other.
- Display, when combining the real and virtual world two basic choices are available: optical and video technology. Each of them has some tradeoffs depending on factors like resolution, flexibility, field-of-view, registration strategies, among others.

2.1 AR Technologies

There are a vast amount of display technologies and SDKs for working with AR. Here, in this subsection, we are going to present the most relevant.

2.1.1 AR Displays

In the category of display devices for AR, in spite of being mostly used in VR, the Head Mounted Display (HMD) allows for the user's visual field to be completely immersed in the augmented environment making any virtual objects seem present and steady in the real world. In a more detailed way, Rolland and Cakmakci (2005) describe the HMD composition as a modulated light source with drive electronics viewed through an optical system which, combined with a housing, are worn on a user's head via a headband, helmet, or around an eyeglasses frame. There are a few equipment that fit in the HMD display system description like Oculus Rift, HTC Vive or HoloLens who are very well known in the field of VR and AR, however they are very expensive.

(a) HoloLens equipment.

(b) HoloLens components.

Figure 2.2: Example of a HMD: HoloLens (HoloLens hardware[2])

It is also important to mention the handheld and the spatial displays. Carmigniani and Furht (2011) describe both. Handheld displays are small computing devices with a display that the user can hold in their hands. They use video-see-through techniques to overlay graphics onto the real environment. There are currently three distinct classes of commercially available handheld displays that are being used for AR system: smartphones, Personal Digital Assitants (PDAs) and Tablet PCs.

[2]https://docs.microsoft.com/en-us/hololens/hololens1-hardware

(a) Using PDAs devices for the invisible train game. (b) Using smartphones for art history classes.

Figure 2.3: Example of handheld AR display usage for collaborative games. (Wagner and Schmalstieg, 2006)

Lastly, although they are the least used for AR applications, the spatial displays make use of video-projectors, optical elements, holograms, radio frequency tags, and other tracking technologies to display graphical information directly onto physical objects without requiring the user to wear or carry the display, what makes it an important type of display for collaboration between users.

Figure 2.4: Example using spacial displays for augmented anatomy in a classroom. (Johnson and Sun, 2013)

2.1.2 AR Software Development Kits

Beyond the importance of analyzing the displays possibilities, it is also necessary to study the available SDKs and choose one for our use case.

An SDK is commonly defined as a set of tools that can be used to create and develop applications. In general, an SDK refers to a full-suite software module that includes everything developers need for a specific module within an application.

Hanafi et al. (2019) compares three of the most important and commonly used AR SDKs:

- ARKit[3] allows developers to easily create unparalleled AR experiences for iPhones and iPads. It uses Visual Inertial Odometry (VIO) (Ismail et al. (2020)) to correctly track the world. Another

[3]https://developer.apple.com/augmented-reality/arkit/

interesting feature of VIO is that the information picked up by the camera sensor is merged with CoreMotion information, this lets the device accurately sense its development in the place. ARKit can detect horizontal planes, track and place objects on smaller feature points and also enables the device to comprehend the scene by the camera and differentiate level surfaces, this allows the user to put virtual objects on the genuine surface.

- ARCore[4] in contrast is supported by Android devices, plus iOS devices in the newer released versions, but with the need of small changes on the developed code, (the versions with iOS compatibility were released after Hanafi et al. (2019) study) and utilizes a procedure called Concurrent Odometry and Mapping (COM) (Nerurkar and Zhao (2017)), to comprehend where the device is in consideration to its surroundings. The visual data is joined with inertial estimations, from the device Inertial Measurement Unit (IMU), to find the pose (position and orientation) of the camera over time. ARCore also detects feature points and planes, but not only the horizontal planes, the vertical planes can be also detected. Planes with a poor texture, like a clean white wall, may not be properly detected, due to the low number of feature points. It enables users to cooperate with objects in the environment too, for that, utilizes hit testing to take a (x, y) facilitate relating to the smartphone's screen and casts a beam into the camera's perspective of this reality, enabling and performing the interaction using the beam vector together with the virtual objects coordinates. ARCore is able to improve its environment understanding as the position of the device is changing, while the user is placing virtual objects over the scene. ARCore can place an anchor to track object's position over time. One last feature to state is that users can trigger AR experiences while pointing their device's camera at a specific image.

- The last one, Vuforia[5] is an SDK that recognizes the 2D planar image, as well as different types of 3D visual objects (plane, box, cylinder...), text and environments recognition, VuMark (a combination of pictures and QR-code). In order to Vuforia users recognize and track real objects, using digital 3D models of the objects, they need to be geometrically rigid and have stable surface features, this means that objects like malleable objects or objects with shiny surfaces would not be recognized. It achieves its goal of tracking images by the feature points that are found in the image itself, by comparing these points with a known image target resource from the database. This image target needs to be under moderately bright and diffused lighting. VuMark allows the freedom for a customized design while, at the same time, encode data and act as an AR target, this can present uniquely identifiable instances.

Hanafi et al. (2019) also summarizes these contents and the advantages and disadvantages of each SDK mentioned above in a few tables. Exploring and updating them, we built a unique table (table 2.1) with all the features that we found that would be interesting to highlight. These features are divided in three categories: the tracking feature, which can be in images, in the device itself and in the detection of plans, the platforms that support the corresponding SDK and the license pricing and its code publicly accessibility.

These SDKs were chosen because they provide a native platform, Unity[6] support and work with commodity hardware, but there are some other SDKs that are worth mentioning. For example, WikiTude[7], that seems to have a good overall tracking technology and works for both Android and iOS, but has a significantly higher pricing, compared with the SDKs described above. Maxst[8], uses

[4]https://developers.google.com/ar
[5]https://developer.vuforia.com/
[6]https://unity.com/
[7]https://www.wikitude.com/
[8]http://maxst.com/

	Tracking			Plataform		License	
ARSDK	**Image Target**	**Motion Tracking**	**Ground & Plan detection**	**Android**	**iOS**	**Free**	**Open Source**
ARKit	Not stable	Stable most of the times	Stable	No	Yes	No	No
ARCore	Not stable	Stable most of the times	Stable	Yes	Yes	Up to a certain point	Up to a certain point
Vuforia	Very stable	Not stable	Detects based on orientation and not with feature points	Yes	Yes	Yes	No

Table 2.1: SDK technologies comparison.

ARCore and ARKit technologies for its own SDK, however, it has a poorer documentation, which makes it less interesting for our study. Some others SDKs, like DeepAR[9], do not have the tracking technology needed. These and other SDKs could also be interesting choices and have been taken into consideration, but due the lower ratings, besides the previous described features and others, they were not deepen as the ARKit, ARCore and Vuforia.

2.1.3 Considerations

Relatively to the display types, after analyzing the different types of displays, the handheld displays appear to be a good choice for the work needed, more specifically the smartphone devices, since it is very uncommon nowadays to find someone who never touched one, or even who does not own one. PDAs present much of the same advantages and disadvantages of the smartphones, but they are becoming a lot less widespread than smartphones since the most recent advances, with Androids and iPhones. Tablet PCs are more expensive and heavy which can be seen as a disadvantage towards smartphones. It would not be fair to say that HMDs are not a good option too, however, due the higher user experience on smartphones and these being cheaper, smartphones seem to be the best choice.

Additionally, the SDK technology that seems to bring the best advantages for our application purpose, is the ARCore software, which reported to have great feedback on its motion tracking and environment understanding features, proving to be a better option than Vuforia. Furthermore, the anchors feature also seems to be an interesting component to overcome some difficulties. Lastly, being free and open-source justify this choice over ARKit software.

2.2 AR in Games

AR is commonly used in games, creating a gaming field within the real environment. Games generally involve mental or physical incentive and can be conceived in many platforms, with distinct complexities and for many purposes. In this section, we divided games into non-serious games and serious games, having their objective as a division mark. The immersion that AR brings can increase the game interest, amusement, information or educational capability, depending on its focus.

[9]https://www.deepar.ai/

2.2.1 AR in Non-Serious Games

The main objective of non-serious games is to amuse and cheer the people that play them, in contrast with serious games, which the primary goal is educational or informative. However, as in serious games, AR can improve its principal purpose, promoting a higher enjoyment while playing the gaming.

The objective of a study performed by Garzón et al. (2019) is to highlight how AR improves the user experience in three areas: retail, medicine and entertainment. We focused on the entertainment area, more specifically, the gaming inside the entertainment. The authors mention the importance of AR in users interaction as it encourages people to move more and wander outside breathing fresh air in an AR play space, whereas non-AR experiences limit these users to a screen.

A very well-known non-serious game that fits this description, in the entertainment gaming field, is Pokemon Go[10]. When Niantic, Inc.[11] launched Pokemon Go, millions of people, around all globe, left their homes to come outside and walk around with an amazing AR immersive experience. Applications of this type require the player location, they work without markers, instead they use the Global Positioning System (GPS) or some digital compass that helps detecting the user's position on the environment.

Figure 2.5: Location based entertainment game: Pokemon GO. (Pokemon GO[9])

Another important portion of the previous mentioned study states that Augmented Reality Games (ARGs) are gaining impulsion as learning guides, real-world games together with virtual elements, can create exciting gaming experiences that may lead to higher learning motivation. Results from a few studies (Fotaris et al. (2017)), show that the use of ARGs in learning raises the learning performance by 58%, plus student motivation and enjoyment by 10%, these values are great and we should have this into consideration. However, these systems also have limitations, such as the distraction of students by the virtual factors, but if proper approaches are used on the system development, AR-based learning has high potential.

[10]https://pokemongolive.com/en/
[11].https://nianticlabs.com/en/

2.2.2 AR in Serious Games

After learning about entertainment gaming and serious gaming in AR, we analyzed the work done by Garzón et al. (2019) that combines both. The aim of his work is to understand the possible benefit when educational and entertainment experiences are set together. To analyze this, two distinct case studies in this field were presented, *ARPuzzle* (Liarokapis et al., 2005) and *ARBreakout* (Liarokapis, 2006).

ARPuzzle (figure 2.6) is a simple game, consisting of six physical pieces and focuses on interaction and collaboration between players, each piece represent parts of City University campus in London. The objective of this game is, after visualizing the 3D geographic map, to reconstruct the puzzle using the six pieces that represent each part of the map. The feedback is given in various formats: text, images and spatial sound. Players can pick up the markers and perceive the geometrical and geographical information, besides that, the player can get recommendations from other players or play with them around the environment since the game allows collaboration.

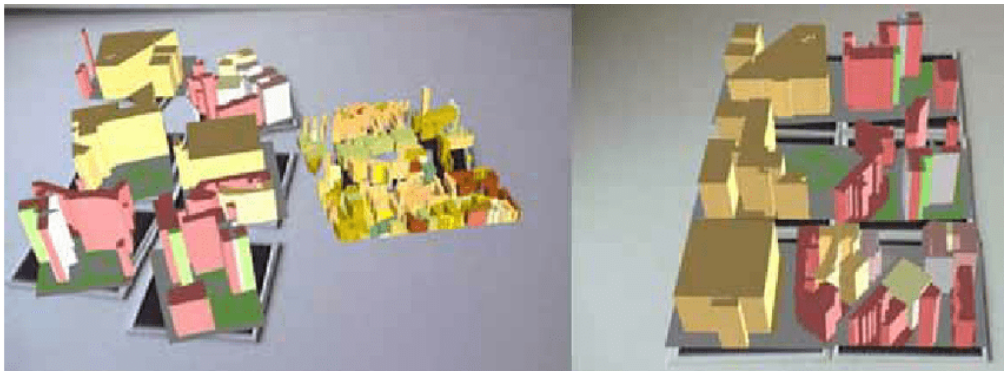


Figure 2.6: Serious game: *ARPuzzle*. Unsolved puzzle on the left image and the solved puzzle on the right image. (Liarokapis et al., 2005)

On the other side, *ARBreakout* (figure 2.7), is based on breakout, an old arcade game, which was transformed to support a tangible AR environment. The goal of the game is to shoot at the bricks making them disappear from the game area, not having a serious purpose such as *ARPuzzle*. This game presents some extra visualization and interaction features. The player can use a handled device with a camera to bring the Breakout game into the physical world. This makes the players feel more immersed into the scenario and promotes collaboration as well. Players can manipulate the AR environment through the controls of the handled device, keyboard and mouse, or in a tangible way. Another interaction technique, is the possibility to zoom, moving the scene closer and further to the camera instead of scaling the game.

To analyze these games, an evaluation was conducted with 30 students playing the first game and other 30 students playing the second one. In both, a questionnaire was completed by each student to find out important issues, such as visualization of the game, interaction techniques, educational motivation, among others. Comparing the results of each game, it was possible to detect that the majority of the users claimed that *ARPuzzle* contains simpler graphics, however, because of the different purpose of the games, *ARBreakout* scored less, due to his purpose being less educational. It is important to declare that all users said that more realism would help improve the immersion in both games and that *ARPuzzle* was more attractive and easy to play, due the absence of any input device (keyboard or mouse) to interact with the game. Although *ARBreakout* having less positive feedback, participants stated that is easier to adapt to the game compared to virtual reality or video games. All of this, shows

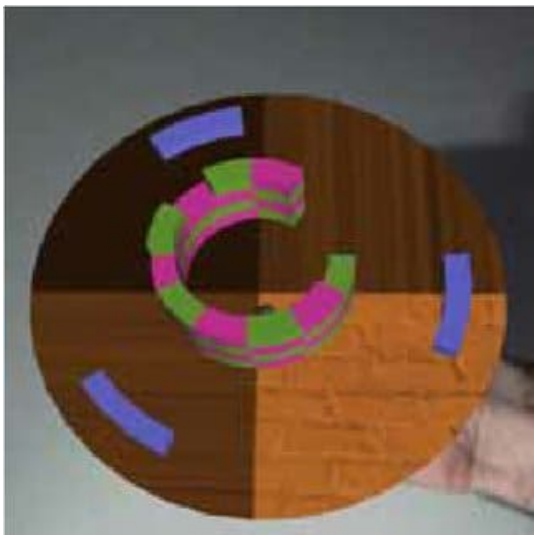


Figure 2.7: Serious game: *ARBreakout*. (Liarokapis et al., 2005)

and allows to conclude, that in general, tangible AR interactions are preferred relatively to traditional ways of playing games.

2.3 AR to Support Human-Robot Interaction

Another application case that AR is used for is robotics. Robotics, by itself, is a very vast field and has many diversified branches, affecting a large number of people and industries in many and different areas.

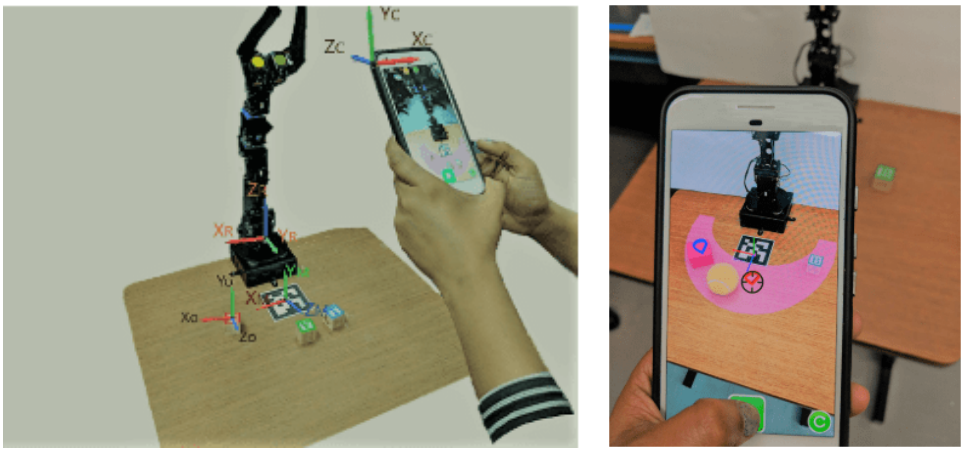
Human-Robot Interaction (HRI) is the study, as the name implies, of the interaction between users and robots. Nowadays, with the increase of robots in society, HRI becomes very important, so modern robots can be part of society daily basis, and in a way, that people can easily use, control and interact with them, without the need of knowing all the vast theory behind it.

It is very common to join AR and Robotics, so the task of HRI, when considering the control of users on robotic machines or devices, is simplified with clues, being the most usual ones, the visual clues. There are several studies in many and the most diverse application fields on robotics.

One that is worth being brought up was written by Chacko and Kapila (2019), that proposed an AR interface for HRI using markerless and marker-based technologies, on a smartphone-based application that uses the ARCore technology. The interface and the robot manipulator are integrated to render a system that allows pick-and-place tasks from any desired location in its workspace. The objects that will be manipulated need to be previously known or covered with markers to detect their positions, being this a considered limitation. The markerless technology, beyond detecting the planar surface that restrains the work area, also enables the placement of virtual anchors in the locations selected by the user on the detected plane, plus identifies the coordinates of physical objects in the workspace, hence, helping the manipulation of unknown and randomly placed objects. When the user selects the target object's start and goal locations, virtual objects appear on those locations, such locations are communicated to the robot and visually rendered for the user about the intended locations on the task.

In a more detailed way, the system is composed by a smartphone, a four degree of freedom robot manipulator and a laptop for processing the data coming from the handheld device. The robot

manipulator takes place on a table and a marker is fixed close to it with known coordinates in the robot coordinate system. While pointing to this environment with the smartphone, it is possible to see the robot workspace, the robot itself and what surrounds it. To send pick-and-place instructions to the robot the user needs to touch the mobile screen. The system is illustrated by figure 2.12.



(a) The HRI system: a robot manipulator and a smartphone hosting the AR application. (Chacko and Kapila, 2019)

(b) User selects an object and its goal location. (Chacko and Kapila, 2019)

Figure 2.8: System and interface images from the proposed AR application. (Chacko and Kapila, 2019)

Once their application is initiated, the starting location of the device becomes the origin and a search for the known marker and for planar surfaces is initialized. Once the marker is detected an anchor is attached to it for a continuous track. The markers pose and size allows the localization of the robot in the world space. After this, the user can select any object on the plane. An anchor is created at the hit pose on the surface and a virtual object is instantiated there too, providing visual feedback to the user in order to verify if the anchor was instantiated at the intended position. Hence, this data is communicated to the desktop PC to command the robot to manipulate the object. Two variants of this system were developed, one that enables the user to pick and place one object at a time, selecting the desired locations, and another one that allows the user to select the start and goal locations of multiple targets and send them at once.

The estimation of the coordinates of the virtual objects is made in a Left-Handed Coordinate System (LHS). Each virtual object's coordinate is transformed from the camera coordinate frame to the marker coordinate frame, also expressed in the LHS. Now, the resulting coordinates are transformed to a Right-Handed Coordinate System (RHS), in order to use and transform them in the robot's coordinate system.

To analyze the proposed interface, a test with 34 participants with little or no knowledge of such interfaces was conceived by Chacko and Kapila (2019). The test consists in accomplishing three tasks. First the participant needs to physically place a block at an arbitrary location in the robot workspace, with the purpose of understanding if the user is able to pick and place objects with the aid of the AR rendering of the workspace. Next, for the second task, the user is asked to pick an object in the robot's workspace and place it a marked fixed location, testing not only the accuracy of the user to transmit pick-and-place intentions to the robot, but also to check the robot's accuracy to execute the user's command. Lastly, participants are challenged to pick multiple objects from different locations,

verifying the interface capability to give multiple commands simultaneously. After concluding the three task, a questionnaire was administered to each user.

After the completion of the experience and given the results it was concluded based on the performance results that the AR interface had an acceptable performance. The questionary-based results indicated a positive satisfaction of the participants and that they highly recommend this AR interface for HRI, being its simplicity an important role in this recommendation, allowing novice users, with the minimum assistance to interact with the robot using this interface.

Makhataeva et al. (2019) presents an AR application for humans to visualize safe zones and potential danger zones in a Variable Impedance Actuated (VIA) robot workspace. The relevance of its work is to prevent accidents caused by the fast robot motion or the failure of the intervening technological equipment, which can be prevented giving visual knowledge about the VIA robot workspace zones. Makhataeva et al. (2019) created a safety-index evaluation based on distance and velocity factors, so it could be possible to generate a color-rendering aura around the robot. This application was built using the Unity game development platform for the simulation environment together with Vuforia to detect AR markers and project the virtual robot with the corresponding aura in the real-world (illustrated in figure 2.9).

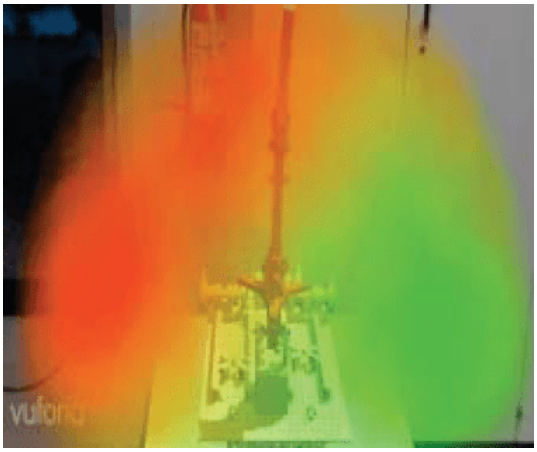


Figure 2.9: Real-world safety aura using AR markers and the web camera (Logitech HD 720p). (Makhataeva et al. (2019))

The last study on HRI improvement using AR technologies that we will refer was conducted by Hedayati et al. (2018) and has the goal to assist robot teleoperation tasks, since these require a high level of operator ability and are mentally demanding. In this work, Hedayati et al. (2018) presents three AR interface designs for aerial robots to replace current interface designs, viewed on display glasses or traditional screens (mobile device, tablet or laptop).

These interface designs are: The *Frustum* design, which provides information on the aerial robot's aspect ratio, orientation, and position, while giving users a clear view of what objects are within the robot's Field of View (FoV). The *Callout* design, attaching a panel with the live video feed above the robot, removing the need for the operator to take the eyes off the robot to check something, like the battery indicator. Lastly, the *Peripherals* design provides a live video within a fixed window in the user's periphery.

To evaluate these interfaces Hedayati et al. (2018) performed an experiment with 48 participants operating a Parrot Bebop quadcopter and wearing a Microsoft HoloLens HMD. Since the *Frustum* and *Callout* designs require a real-time understanding of the robot position, motion tracking cameras

are used to localize the robot, providing these values to the HoloLens application. Regarding the live video feed, from the robot's camera, displayed in the *Frustum* and *Callout* designs, is broadcast to the HoloLens through a desktop computer. The participant task was to pilot the aerial robot and take photos of rectangular colored frames, prioritizing capturing these images as fast, as precisely and with the fewest number of pictures as possible. Accomplishing this task, it was desirable to measure the accuracy (how well participant photographs captured the inspection target), the completion time (the total flight time), operational errors (the number of times they crashed the robot or otherwise caused it to land prematurely), distracted gaze shifts (the number of times the participant was distracted looking away from the robot) and the distraction time (the total time spent not looking at the robot).

This experiment revealed that *Frustum* and *Callout* designs improved the accuracy over the baseline interface and the *Peripheral* design showing even further benefits, outperforming the remaining designs. Using the *Frustum* and the *Peripheral* design, the task completion time was decreased and every AR design significantly reduced the number of errors. It was also possible to verify that both the distracted gaze shifts and the distraction time were smaller compared to the baseline. It was also found a positive effect on the confidence operating the robot. These results allowed Hedayati et al. (2018) to conclude that the created interfaces demonstrated significant improvements over the modern interface, allowing to complete aerial tasks faster and more precisely with a safer operation. Users rated these interfaces as more useful, more comfortable and encouraging when operating the robot, leading to easier flights. They believe that the AR interfaces success is due the access to the live video feedback without taking the eyes of the robot, since traditional interfaces force users to make contextual switches, sacrificing either the awareness of the robot in the environment or the ability to monitor the robot's camera feed at any given time. Overall, this study shows that AR can be a great tool to mediate HRI.

2.4 AR for Helping People with Motor Disabilities

Some people experience certain difficulties in controlling motorized wheelchairs for a variety of reasons, from lack of experience to a possible user's difficulty to get used to it. In addition, motor disabled people, due to their physical limitations, also have to face adversities in reaching or visualizing some places (for example, a tall closet). Fortunately, studies have already been carried out with the aim of using AR to help people with these difficulties and the results have been quite interesting and satisfactory.

Rashid et al. (2017) conducted a study in the context of Smart Cities, which aims to provide independence and autonomy to users who visit stores in wheelchairs and intend to do their shopping alone, without the help of others. In this study, the target users do not have the difficulties to maneuver the wheelchair, but rather the difficulty to reach certain areas of the stores, like a shelf out of their arm's range. These users were divided into two groups, the first with motor disabilities at the legs and the second group, which in addition to this limitation, also have lack of mobility with their hands.

The interaction of group 1 users with the world, was, for example, in a movie store, pointing the tablet or smartphone to a closet shelf, thus being able to visualize on their devices the movies that are out of their FoV (figure 2.10). Now, they are able to obtain the most diverse information of a given movie without the need to grab it. Group 2, on the other hand, due to their additional hand problems, which would lead, for example, to the impossibility of being able to accurately point to a specific part of the shelf, could not use the same method. To attenuate this, Rashid et al. (2017) proposed the use of a second touchscreen next to the bookcase. In fact, this second system could be used, not only for these users, but also for the ones belonging to group 1, as well as people without any motor disability.

Users who already had contact with similar technologies showed ease using the application cor-

Figure 2.10: Close up view of the hand-held screen, showing the superimposed item information, and the green square indicating the active area (the user clicked location). (Rashid et al., 2017)

rectly. However, the others demonstrated some adversities. But, in general, the results and opinions from the users regarding the proposed system were satisfactory. This study brought interesting conclusions, such as, for example, users in group 1 having preferred the use of smartphones over the tablet and the preference of users in group 2 for larger screens with larger fonts.

The number of studies joining AR with robotics and joining AR with wheelchairs is quite abundant, although the combination of AR with robotic wheelchairs it is not common, so the works in this field are few.

Another study in this area was performed by Zolotas and Demiris (2019), associating AR technologies with a branch of robotics, the robotic wheelchairs. This one has an experiment with able-bodied volunteers, some with experience with robotic wheelchairs and others with no experience at all. The goal is to help the users to resolve the misalignment between their actions and the respective internal models of the robot. Whenever the human actions do not raise the intended system response, the role of AR is to help visually reveal the robot's inner workings to human operators. To accomplish that, a HoloLens HMD was used and three visualization clues were given.

The first virtual aid is a collision sphere paired with a directional arrow. The spheres highlight collisions in the physical environment, this increases the contextual awareness of users, enabling them to identify why their actions can lead to unexpected behavior. The arrows signal where these collisions are situated from the operator's perspective. The second visualization is a mini-map panel, this mini-map is annotated with laser scan readings and forecasts of the robot's estimated poses after applying input commands. Lastly, they extended the display to supplement the rear view with a virtual wheelchair avatar during backwards movements, this supplies users with a wider contextual perspective of the robot's navigation. These AR help features are presented in figure 2.11.

This last mentioned work had interesting results and conclusions too. They indicate that volunteers in the experimental study recovered faster from getting stuck when guided with AR. Furthermore, with AR, users took less time to traversal doorways. Most of the users claimed that the AR assistance helped

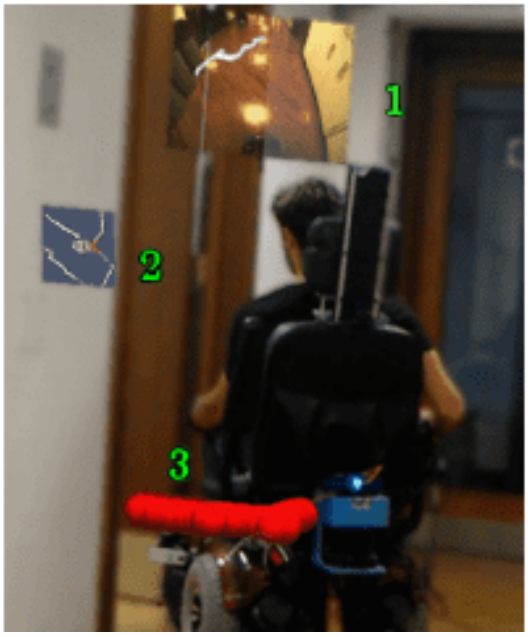


Figure 2.11: Composite image of the visualizations rendered during assisted robot navigation. The rear view display (1) overlays virtual objects onto an image taken from a camera situated on the back of the seat. The mini-map panel (2) depicts a bird's eye view of the wheelchair configuration and its surroundings. Finally, red spheres(3) are placed atop of real-world referents to highlight collisions. (Zolotas and Demiris, 2019)

---

to understand where the problem was, as well as explaining why the safety algorithm was changing the way the whelchair behaved. These, findings support the fact that subjects overcame incidents more effetively.

One last paper that we will mention on this area, was written by Zolotas et al. (2018). The previously mentioned and this one have many similarities, both the objectives, the technologies and the way that AR helps on accomplish them have a lot in common, if not getting in to many details. However, since the experiment made and some conclusions were different, its reference is also important.

Here, each participant on the experiment had to complete a navigation route four times in a row. In order to to investigate the effectiveness of the AR clues, some subjects had the AR visualization and the remaining did not. The authors used the total time to complete each trial as a performance indicator of the overall AR feedback. The results on this measure indicated that the group with visualizations demonstrated more variable performance, with a greater decrease in time relative to their first trial. On the fourth trial, the last trial, this one without AR visualization, there was a slight dip in performance but Zolotas et al. (2018) concluded that no strong claim could be made for any dependency on the AR assistance. The HMDs lower the FoV and it was noticed that individuals with upper body mobility impairments are prone to colliding with obstacles outside of their viewing capacity during typical wheelchair navigation procedures (like rotating in place or reversing).

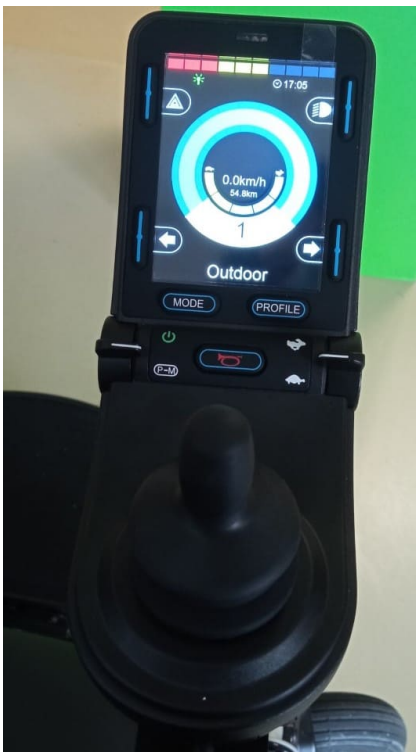
Analyzing the related work in this area, we highlight that AR is a promising technology to help disability people and in general, the studies that were carried out had great and beneficial results, besides that, there are a few improvements that can and should be taken into consideration.

2.5 IntellWheels 2.0 Project

The IntellWheels 2.0 is a recent project that takes advantage of intelligent wheelchairs for assisting individuals who have disabilities. The project has several partners which include the University of Aveiro, the Faculty of Engineering of University of Porto and the companies Optimizer, RehaPoint e Ground Control Studios.

The wheelchair exhibited in figure 2.12a, related to the IntellWheels 2.0 project, is a robotic wheelchair that uses a motor, a joystick and some sensors to move. The wheelchair used for this work is identical to the power wheelchair TA IQ MWD[12], with a small number of additional features (a few lift options and additional sensors for other robotic related purposes), therefore its feature description is also quite identical to this model.

(a) Wheelchair side picture.



(b) Wheelchair controller picture.

Figure 2.12: TA IQ MWD wheelchair pictures.

The TA IQ MWD wheelchair has five speed levels, a maximum speed of 12.5 km/h reachable in 3 seconds, a turning radius of 45 cm to easily spin in tight spaces. All of its six wheels (four castor wheels and two driving wheels) are enabled with suspensions for smooth and comfortable trips. A controller, composed by a display, various buttons and a joystick, is integrated in a swing-away bracket, so users can have control over the wheelchair speed, direction and over the lifts remotely controlled, it can also be used to obtain some current information, like the battery indicator or the current speed (figure 2.12b). It is possible to adjust the height or the angle of various components for each individual user, like the armrests or the leg rest, beyond that, in a easier way using the controller, the seat can also be lifted, so places with different heights can be reached with low effort.

[12]https://www.ta-service.dk/uk/power-wheelchair/produkter/896-ta-iq-mwd

A group of students from the University of Aveiro (IntelChair) performed a study with a wheelchair also integrated in the research and improvement of intelligent wheelchairs, but based in a different power wheelchair model, hence, making all of the integrated sensors available on their website[13]. We analyzed and considered those that are important to mention. The wheelchair, used in this study, is equipped with a camera for object recognition and collision detection, as well as a laser rangefinder and IMU for room mapping and navigation. In order to manually control the wheelchair, a virtual joystick is used. While using the joystick, the user is reading and sending the latest angular linear velocity commands making the wheelchair move. Furthermore, a laptop can be used to connect all these different nodes. The work made by this group of students allowed, after mapping a specific floor, using the integrated equipment, the wheelchair to move to a certain place of the floor by itself, without user interaction, in a safe way, since prevents object and wall collisions.

[13]https://xcoa.av.it.pt/~pei2018-2019_g07/

Chapter 3

ARCore Analysis

As previously seen, in section 2.1.2, ARCore software stood out as the ideal AR technology, so we started exploring this technology together with Unity, a cross-platform game engine developed by Unity Technologies with support to many AR technologies, including the ARCore. In this chapter, we will describe the diverse technology features provided by ARCore and the conducted studies to evaluate the robustness and how to get the best out of some of them.

3.1 ARCore Features

Here, we will describe ARCore fundamental concepts related to this work, their advantages, limitations and the empirical work, as well as the basic tests developed to explore those. The ARCore features that will be characterized throughout this section are the motion tracking, the environment understanding, the anchors technology and the augmented images. The light estimation and depth technologies will be also described, although they have not been explored and tested so thoroughly.

The first limitation we point out is that not all Android and iOS devices support ARCore. The ARCore page has a wide list of supported devices[1], all of the Android ones, run Android 7.0 version or newer, while for iOS devices, targeting iOS 9.0 or later.

3.1.1 Motion Tracking

In order to know where the device is located in the world, ARCore uses a technique, also very common in robotics, called Simultaneous Localization and Mapping (SLAM). It detects visually distinct features, feature points, to compute its change in location while moving, which is combined with the device's IMU to estimate the camera pose relatively to the world.

The position of the camera at a given time is always relative to its starting pose, in other words, relative to where the device is found when the application launches the ARCore session. However, the camera orientation is automatically adjusted accordingly to the device gyroscope sensor.

We did some tests to evaluate the behavior and the precision of ARCore motion tracking. While printing the camera position on the screen, we walked around 20-30 meters and we returned to the initial position. Making this test, it was possible to find out that the motion tracking computation that ARCore performs it is not perfect but seemed robust, only failing the coordinates for the initial position for a few decimeters. Besides this, we also explored the behavior of the motion tracking on a space with no texture, like walking while pointing the device to a long white wall with no texture.

[1] https://developers.google.com/ar/discover/supported-devices

In this case, the coordinates of the virtual camera device did not change, resulting in a big flaw when returning to the regular environment. Later on, the accuracy of motion tracking was studied in depth.

3.1.2 Environment Understanding

ARCore continuously looks for an agglomerate of feature points strongly connected that appear to be on common surfaces, like tables or walls, making these surfaces stored in its world model as horizontal or vertical planes. It is also possible to create depth maps, images that contain data about the distance between surfaces from a given point, using the main Red Green Blue (RGB) camera from the device.

Testing this feature, it was possible to verify that the environment understanding described above is stable. After giving some acknowledgment about the surrounding environment, virtual objects placed on the ground, tables and other flat surfaces (horizontal and vertical), seem to rest a stable and correct position.

Light Estimation

Light Estimation is a module from ARCore capable of detecting and estimating the lighting of the real environment, which grants the possibility to trigger certain virtual actions based on the color, intensity and direction of the main existing light, together with the overall ambient light coming in from all directions in the scene (ambient spherical harmonics). Furthermore, this allows the technology to light the virtual objects under the same conditions as the environment around them, originating a more realistic experience. This module was not used since the lighting of the scenes for our application is mostly uniform.

Depth

Another strong component of ARCore is the depth understanding, which allows the creation of depth maps that can have many purposes. It can be used for occlusion, i.e., the object will be rendered so that, when it is partially/totally behind a real object, it is partially/fully hidden, and to improve the interaction realism and immersion with real surfaces, like a user's hit test to place a virtual object or the behavior of a virtual object that collides with or moves along a surface. The Android device used for the evaluation presented below and for the usability tests, the Samsung Galaxy S7, does not support the Depth API, however, devices that support it can take advantage of this technology in the built application (the lists that support the depth API can also be found in the list of supported devices[1]).

3.1.3 Anchors

Anchors enter here as the ARCore feature which fixes a location and orientation in the real world. These anchors are usually created together with Trackables, which are special objects that ARCore will track over time, like geometrical planes and feature points (there is no public information about what are this feature points). Anchors are used for ensuring that one or more virtual objects, attached to the anchor, stay at the same position and orientation in space over time. As the anchor pose adapts to world space updates in each frame, it maintains its location in relation to the world space (within the limits of the accuracy of the algorithm) and updates the objects attached to it, accordingly. Some good practices using anchors, as documented, are:

- Create anchors in the context of a Trackable, such as a table with some texture. Anchors can also be attached to the ARCore Session, appearing to stay at the same pose in world space throughout the user experience.
- Use the same anchor when virtual objects are nearby, so the distance between them is always steady. Using less anchors helps to reduce Central Processing Unit (CPU) costs and as anchors are adjusted independently of each other, can cause the virtual objects to align and rotate relatively to each other, breaking the AR illusion where virtual objects must maintain their place in relation to each other.
- Create a new anchor when a new virtual object is far away (farther than eight meters) from the ones previously created, preventing unexpected rotational movement due to ARCore's updates to world space coordinates.

While a user moves around the environment, the anchor and the virtual objects associated to it can slightly move from their original position, in ARCore world space. This is caused by errors in the calculation made by the motion tracking technology. However, when the user points the camera to the anchored place, and this place is acknowledge as such, the virtual object will realign to its starting position.

There is also a variant of these anchors, the cloud anchors, that unlike the previous ones, are hosted on the ARCore Cloud Anchor API[2], one of the services offered by Google Cloud Platform (GCP)[3]. This hosting allows collaboration between users and persistence, we will focus on the persistence advantage. Cloud anchors, have two main functions: host and resolve. During hosting, ARCore uploads data for the anchor to the ARCore Cloud Anchor API service, which returns a unique ID for that anchor. On the other hand, while resolving, the application can use the unique ID, obtained in the host function, together with the ARCore Cloud Anchor API service, to recreate the anchor in the location where it was hosted.

To host an anchor, ARCore uses a 3D feature map of the space surrounding the anchor (the center of interest). To obtain this feature map, the device's rear camera must map the environment in and around the center of interest from different viewing angles and positions for about 30 seconds before the host call. In order to give some feedback, we developed a surrounding area around the virtual object that turns green when the quality of the feature points found for that specific anchor is good, indicating that probably the anchor will be resolved with high accuracy (figure 3.1). This feature points quality estimation is achieved using ARCore *estimateFeatureMapQualityForHosting* function. In case the surrounding area turns yellow, instead of green, indicates that the quality for the obtained feature points is sufficient to resolve an anchor, but also that it would be better to obtain a few more feature points to result in more accurately resolved poses. Otherwise, if the surrounding area do not change its original color, the local is not good to place the anchor, because the number or quality of the feature points found is low. This behavior is described in algorithm 1.

To resolve an anchor, that was previously hosted, causes the application to periodically compare visual features from the scene against the 3D feature map that is saved in the ARCore Cloud Anchor service. When it matches with the place where the anchor was hosted, it will return the pose of the cloud anchor. Now, with the pose of the cloud anchor, it is possible to attach one or more new or locally stored virtual objects to it. The returned pose of the cloud anchor, when resolved, is always relative to the initial device pose and to the world tracking until the resolve function occurs.

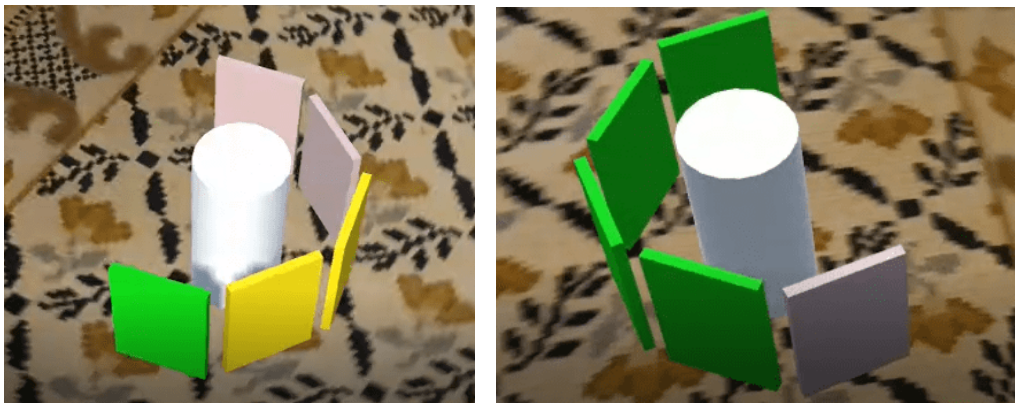
Limitations on cloud anchors:

[2]https://console.cloud.google.com/apis/library/arcorecloudanchor.googleapis.com

[3]https://console.cloud.google.com

Algorithm 1 Algorithm for verifying the anchor quality.

```
Require: anchor ≠ NULL
  anchor_walls ← [1..5]
  if camera_is_looking_to_the_anchor() then
      for wall ∈ anchor_walls do
          if camera_is_looking_to_anchor_wall(wall) then
              quality ← estimateFeatureMapQualityForHosting(getPose())
              if quality = GOOD then
                  paint_anchor_wall(wall, green)
              else if quality = SUFFICIENT then
                  paint_anchor_wall(wall, yellow)
              end if
          end if
      end for
  end if
```



(a) Gathering the feature points from the left side of the virtual anchor.

(b) Gathering the feature points from the right side of the virtual anchor after the left half being checked.

Figure 3.1: Verifying the quality area for the anchor. White equals insufficient, yellow equals sufficient and green means good.

- If using the API Key option for authentication (which consist of an encrypted string that identifies an application) it is possible to host an anchor for 24 hours. It is also possible to use the Keyless authentication option to host a cloud anchor for up to 365 days. This authentication can be configured in Unity together with the user GCP account.
- There is no public information about how the storage in the ARCore Cloud Anchor API is performed.
- An ARCore session, which holds and manages the global and tracking information from the AR system in the application, can contain up to 20 Cloud Anchors at a time.
- It is possible to have 30 host requests active at a time or up to 300 resolve requests in process at a time.

It is possible to manage the hosted cloud anchors through the Cloud Anchor Management API[4]. Here, we can list all the hosted anchors, delete one or all of them and change their lifetimes.

3.1.4 Augmented Images

ARCore also allows to trigger actions when it recognizes an image that corresponds to one previously saved in the image database. This can be used for many features, like creating a virtual object above the image. In addition, ARCore does an adjustment to the motion tracking device pose, when it gets lost and the application already recognized the image earlier. These images do not need to be fixed to be recognized, ARCore Augmented Images technology also responds to moving images. In the ARCore website there are several tips for the image selection that can be summarized in the following list (a tool for score the quality of any image provided to the application, is included in the SDK):

- The images resolution should be at least 300 x 300 pixels.
- Images should be in Portable Network Graphics (PNG) or JPEG file format.
- Avoid images containing a large number of geometric features, or very few features (e.g. QR-Codes...).
- Avoid images with repeating patterns.

If an image from the database is being tracked, its pose can be estimated relatively to the device pose, this feature was fundamental in the evaluation that we carried out and that we will explain in the next section.

3.2 Evaluation of ARCore Motion Tracking Technology

The information about how ARCore performs motion tracking is scarce. To evaluate its performance and associated errors, we built an experiment based on walking certain known distances, using the Samsung Galaxy S7 Android device. Then, we compared and analyzed the saved virtual coordinates given by the application to the real ones, measured with a measuring tape. The three-dimensional coordinate system offered by ARCore motion technology in Unity applications is made in the LHS.

The main goal of this evaluation is to explore the possibility of having an alternative to the cloud anchors. Having a robust and accurate motion tracking technology, it is possible to create the anchors, without the need to save them beforehand in the GCP or the need to manually create them each time that the application starts, from a file filled with each anchor pose and develop the main application to automatically create them when the user points the device's camera to the respective location.

3.2.1 Methodology

To verify the robustness and accuracy of the ARCore motion tracking module, we designed several experiments for analyzing the technology under different conditions. Two indoor scenarios were used for this evaluation: a house and the University of Aveiro soccer field from IRIS robotics laboratory. The experiment consisted on walking along a known route, in each of the scenarios, holding the smartphone with the hands. The places to perform an action (start, change direction and take results)

[4]https://developers.google.com/ar/develop/unity/cloud-anchors/management-api

were manually marked with sheets to reduce errors of human source. In each experiment trial, we took both the position and orientation of the points of interest using the interface button responsible for the *take results* action. The trajectories taken in each experiment location are represented in figures 3.2 and 3.3.

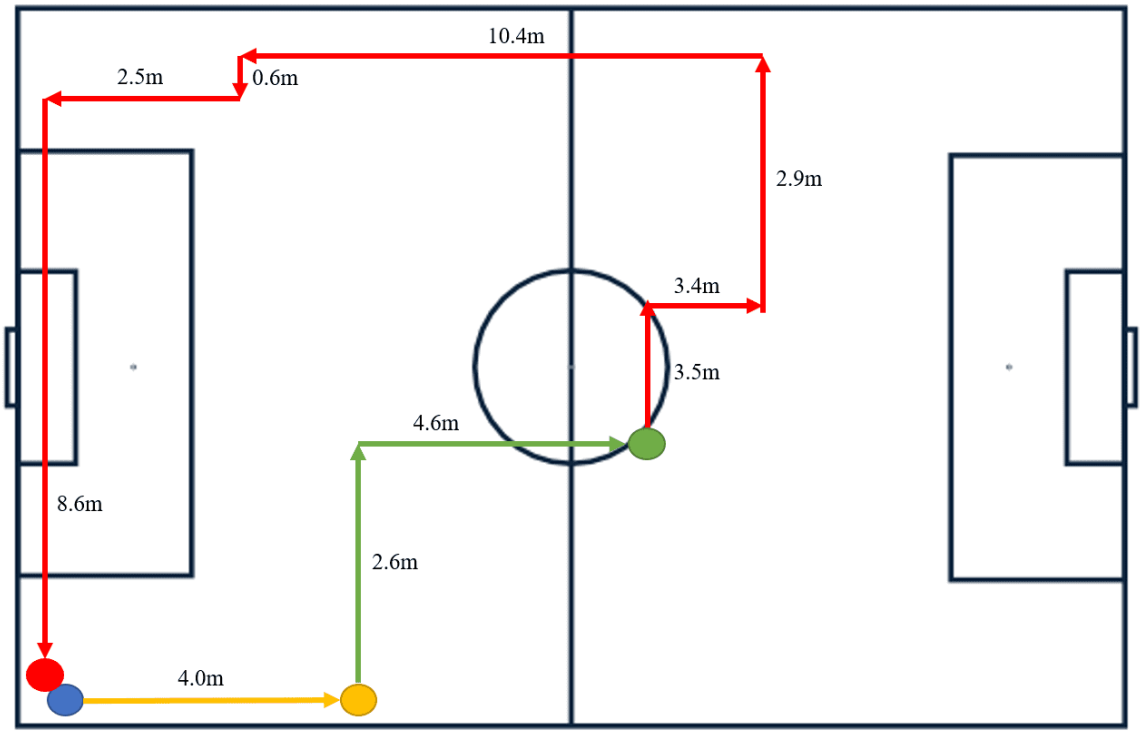


Figure 3.2: Representations of the path of the experiment in the University of Aveiro soccer field from IRIS Lab.

Schematics 3.2 and 3.3 description:

- The blue circle represent the location of the starting marker. This marker also represents the last point to take results.
- The yellow circle represents the marker of the first point to take the first results. Here, two results were taken, the first one when arriving to the marker and the second, after a 90 degrees rotation around the marker. The marker was not in the FoV of the device's camera while the rotation was being performed.
- The green circle represents the marker of the second point to take results.
- The red circle represents the third marker to take results. This marker is very close to the starting marker (blue circle).
- The yellow path represents a small walking path from 1 to 8 meters.
- The green path represents a medium walking path from 8 to 20 meters.
- The red path represents a long walking path of more than 20 meters.

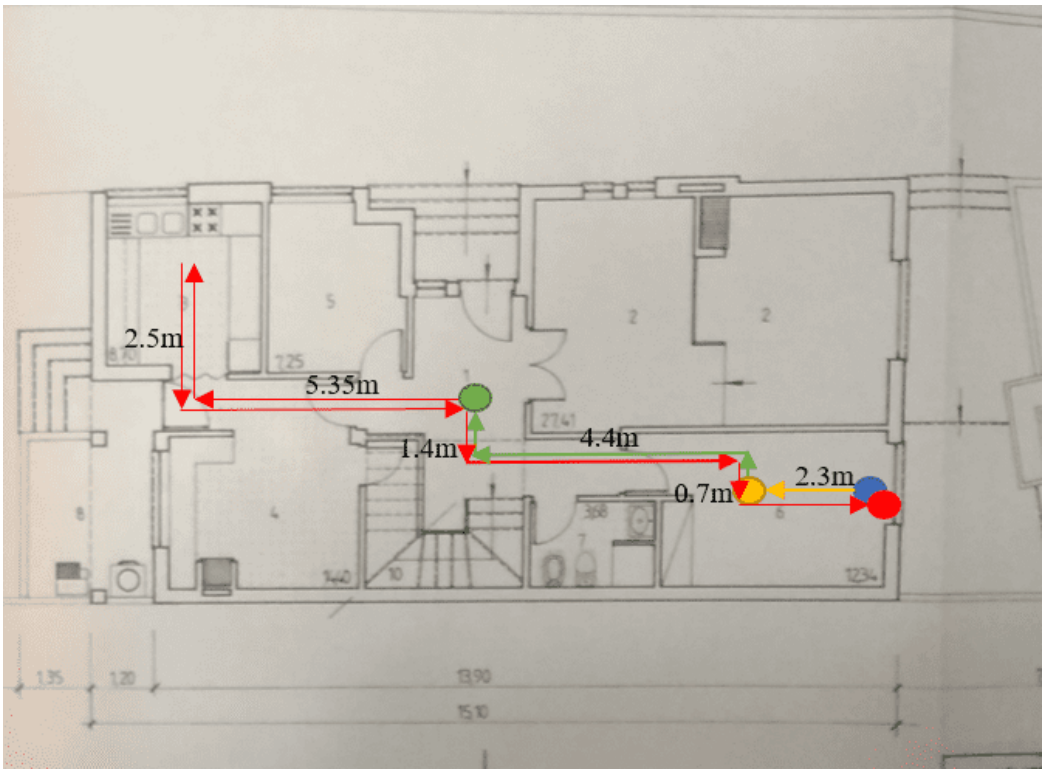


Figure 3.3: Representations of the path of the experiment in a house

---

- All the three paths (yellow, green and red) start at the starting marker.
- Arrows represent the direction of the corresponding path.
- Locations where a change of direction is carried out were also manually marked with sheets. These markers are not in the device's camera FoV while the rotation occurs.

The need to add the red circle, quite close to the blue circle, arises from an integrated calibration that ARCore automatically performs. When looking again to a marker that the application already saw, the current camera pose is computed and updated based on the camera frame when this marker was first seen, erasing all the produced drift until it. This adjustment was found during initial versions of this evaluation. We continued to gather the results when returning to the initial position and pointing to the marker corresponding to the blue circle (after the path being traveled and all the other results obtained), in order to verify if the adjustment was always done and how accurate this adjustment was.

We used several distances to check if the error is dependent on the walked distance, i.e. if there is any notable drift. On the other hand, the usage of two scenarios will allow to verify the results of ARCore technology inserted in different environments, concerning luminosity, surface textures, physical objects around or other independent variables that can influence the motion tracking performance.

For evaluating how the user behavior would influence the motion tracking performed by ARCore technology, three methods were used in the experiment. These methods are dependent variables and they illustrate possible user behaviors using the application. The three methods are:

- **Steady method:** The device is relatively steady, pointing forward, i.e., to the path being followed. In the places of change of direction, the device rotates following the body rotation.
- **Side method:** Pointing the device to the sides while moving.
- **Obstruction method:** Obstructing the camera with the fingers, for about one second, every 2-3 seconds.

We used two walking patterns, a slow one and a faster one, being these dependent variables too. The slow walking speed consists in a walking pace of around 2.8km/h and the fast one around 5km/h. With these approaches, we intend to analyze if the user velocity can have any influence in the pose (position and orientation) accuracy.

We repeated each approach (method and walking speed) three times, collecting three results for each marker (represented with circles in figures 3.2 and 3.3) to do a mean and the standard deviation, giving a more accurate value. Obtaining a total of 3 x 2 x 5 x 3 (*nº of methods* x *nº of walking speed patterns* x *nº of take results actions* x *nº of repetitions*) results in each scenario.

The experiment was performed as follows: At the beginning, the markers were placed on the corresponding positions. The circles in figures 3.2 and 3.3, were replaced by A4 sheets with the markers saved in the application augmented images database, so the ARCore Augmented Images technology is able to recognize them, making their information (id and corresponding pose) accessible. The images used for the marker can be found in appendix A.1. The places that represent a change of direction were marked with white A5 sheets with a hand-drawn arrow pointing to the next orientation. The hardware starting position was controlled to be matching the previous ones (using the Samsung Galaxy S7 original box) and placed in a way to start tracking the world when the application was launched. This starting position is parallel to the first marker so that the Z-axis on the ARCore world coordinate system is parallel to the path between the first marker and the second one. Then, the smartphone is raised and pointed to the first marker (matching the blue circles in figures 3.2 and 3.3), so the marker can be recognized and its pose can be saved, pressing a button on the application interface. This first marker was used as a comparison marker, so each remaining result can be taken relatively to this one. Having the pose of the first marker saved, the walking speed, method and path of the corresponding trial were performed to the next marker, where the pattern of pointing the smartphone to the marker and pressing the interface button, when the image is recognized, were fulfilled and then this behavior was repeated until the path ends and all the results of the corresponding run were gathered.

For each trial, eight files were written and saved in a Comma-separated values (CSV) format. One of these files contains the markers poses, in every frame that ARCore is recognizing them, the pose and the corresponding timestamp is saved, with the goal to verify if the marker pose was stable while pointing to it. Another file is responsible for keeping the device's camera pose and the corresponding cycle timestamp. This file is written, in every application cycle (with a frequency of 30Hz), from the starting of the application to its ending. The remaining files are responsible for keeping the markers pose when the interface button is pressed. The first one, contains the first markers pose. And the remaining five have the corresponding markers pose relatively to the first one. Besides this, the error between the measured ARCore coordinates and the ones measured by hand, of the respective markers pose, is calculated and also written on the respective marker file.

3.2.2 Results

To organize and show the results gathered from this evaluation, four tables were built (two for each scenario), twelve two-dimensional plots (two for each method with different walking speeds in each

scenario) and twelve plots representing the two-dimensional plots in a three-dimensional coordinate system.

Table 3.1 summarizes the obtained results relatively to the position error in the house scenario, from the three runs for each method and walking speed. This position error is relative to the calculated 3D straight distance between the first marker and the following markers. Table 3.2 is identical, but summarizes the results relatively to the orientation error from the Y axis (corresponding to the vertical axis in the LHS) in the house scenario. The Y axis was chosen because the device is mainly rotated around this axis and in the goal application the rotations around the remaining axis are not critical. Table 3.3 and 3.4 are similar to those described above (tables 3.1 and 3.2) with the difference that the respective errors and standard deviations are calculated with the results obtained in the IRIS Lab soccer field scenario.

Figures 3.4, 3.6 and 3.8 represent the device's camera coordinates from each frame, so we can visually observe, in 2D plots, the path walked in the ARCore point of view in the house scenario. Figure 3.4 representing two runs using the steady method, figure 3.6 illustrating two runs with the side method adopted and figure 3.8 showing two runs applying the obstruction method. Each figure is therefore divided into two plots, the left plot showing the slow walking speed result and the right one representing the fast walking speed result. This representation was also drawn in 3D plots, figures 3.5 (steady method), 3.7 (side method) and 3.9 (obstruction method), to verify the possible existence of unwanted behaviors on the Y axis. In this set of figures, containing the 3D plots, the upper plots represent the slow walking speed and the bottom plots represent the fast walking speed. In each plot, the different colored lines represent the direction, being the black line the way to half of the path and the brown line the way back. Besides this, it is also possible to visualize where the markers are positioned in the real world (circles) and the results obtained in each marker from ARCore motion tracking (crosses). As in the blue and yellow circles positions two results were gathered, the light blue and light yellow crosses represent the second obtained result on the corresponding place. The equivalent representation and logic was performed for the IRS Lab soccer field scenario, with the major difference that in this scenario there is only one direction (represented with a black line), since in this scenario, the path is not divided in two equal half-paths. This results are presented in figures 3.10, 3.12 and 3.14 for the 2D plots and in figures 3.11, 3.13 and 3.15 for the three-dimensional ones. Every plot figure is respective to the first run of each method and walking speed in the respective scenario.

		Marker 2_A (Yellow circle)		**Marker 2_B (Yellow circle after rotation)**		**Marker 3 (Green circle)**		**Marker 4 (Black circle)**		**Marker 5 (Blue circle)**	
Walking Speed	**Method**	**Average distance error (m)**	**Standard deviation (m)**	**Average distance error (m)**	**Standard deviation (m)**	**Average distance error (m)**	**Standard deviation (m)**	**Average distance error (m)**	**Standard deviation (m)**	**Average distance error (m)**	**Standard deviation (m)**
Slow	**Steady**	0.04	0.03	0.05	0.02	0.02	0.01	0.11	0.12	0.19	0.23
Slow	**Side**	0.04	0.01	0.04	0.02	0.24	0.13	0.25	0.19	0.20	0.17
Slow	**Obstruction**	0.69	0.65	0.73	0.64	2.56	2.36	3.07	2.75	0.09	0.04
Fast	**Steady**	0.07	0.02	0.19	0.11	0.15	0.06	0.35	0.21	0.19	0.15
Fast	**Side**	0.14	0.16	0.14	0.17	0.42	0.37	0.44	0.23	0.25	0.30
Fast	**Obstruction**	0.65	0.91	0.65	0.87	3.06	2.25	6.60	8.08	6.53	8.63

Table 3.1: Summary of the results of the walked distance error in the house scenario.

		Marker 2_A (Yellow circle)		Marker 2_B (Yellow circle after rotation)		Marker 3 (Green circle)		Marker 4 (Black circle)		Marker 5 (Blue circle)	
Walking Speed	**Method**	**Average orientation error (Y axis) (º)**	**Standard deviation (º)**	**Average orientation error (Y axis) (º)**	**Standard deviation (º)**	**Average orientation error (Y axis) (º)**	**Standard deviation (º)**	**Average orientation error (Y axis) (º)**	**Standard deviation (º)**	**Average orientation error (Y axis) (º)**	**Standard deviation (º)**
Slow	**Steady**	5.31	0.09	5.89	0.10	2.11	0.12	4.67	4.72	1.27	1.46
Slow	**Side**	4.96	0.28	5.87	0.25	2.25	1.03	2.98	1.14	1.92	2.26
Slow	**Obstruction**	5.39	1.34	27.5	18.05	29.71	14.92	84.34	100.19	0.21	0.08
Fast	**Steady**	4.39	0.63	14.93	14.11	12.72	14.28	11.85	14.01	0.31	0.35
Fast	**Side**	2.71	0.60	3.73	0.42	2.62	2.55	2.90	1.80	2.39	2.89
Fast	**Obstruction**	6.79	3.79	6.48	4.03	39.85	42.24	77.80	65.52	70.66	74.56

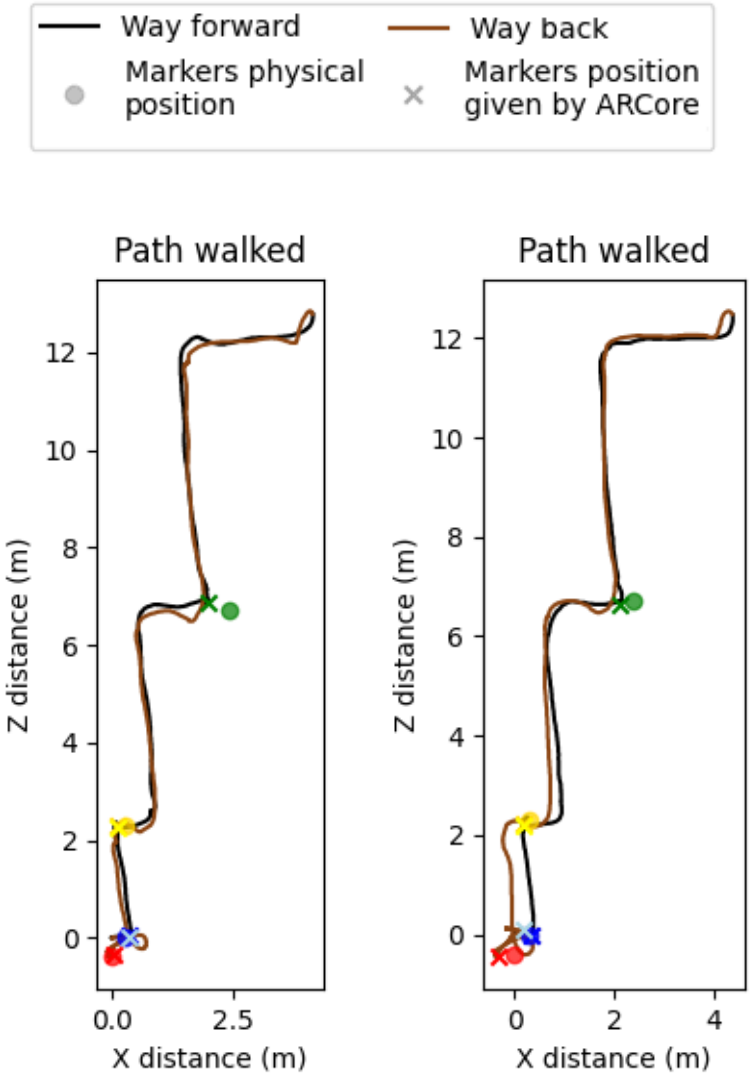
Table 3.2: Summary of Y-axis orientation error results in the house scenario.

		Marker 2_A (Yellow circle)		Marker 2_B (Yellow circle after rotation)		Marker 3 (Green circle)		Marker 4 (Black circle)		Marker 5 (Blue circle)	
Walking Speed	**Method**	**Average distance error (m)**	**Standard deviation (m)**	**Average distance error (m)**	**Standard deviation (m)**	**Average distance error (m)**	**Standard deviation (m)**	**Average distance error (m)**	**Standard deviation (m)**	**Average distance error (m)**	**Standard deviation (m)**
Slow	**Steady**	0.07	0.01	0.08	0.01	0.24	0.17	0.42	0.28	0.33	0.36
Slow	**Side**	0.04	0.02	0.03	0.01	0.15	0.19	0.52	0.26	0.96	0.42
Slow	**Obstruction**	0.25	0.30	0.21	0.27	0.87	0.54	14.31	7.94	15.48	7.58
Fast	**Steady**	0.08	0.06	0.10	0.02	0.32	0.07	0.48	0.11	0.51	0.35
Fast	**Side**	1.34	1.85	1.30	1.81	1.50	1.37	2.01	2.11	0.51	0.38
Fast	**Obstruction**	0.22	0.21	0.18	0.18	1.78	1.33	8.21	5.13	8.61	6.07

Table 3.3: Summary of the results of the walked distance error in the IRIS Lab soccer field scenario.

		Marker 2_A (Yellow circle)		Marker 2_B (Yellow circle after rotation)		Marker 3 (Green circle)		Marker 4 (Black circle)		Marker 5 (Blue circle)	
Walking Speed	**Method**	**Average orientation error (Y axis) (º)**	**Standard deviation (º)**	**Average orientation error (Y axis) (º)**	**Standard deviation (º)**	**Average orientation error (Y axis) (º)**	**Standard deviation (º)**	**Average orientation error (Y axis) (º)**	**Standard deviation (º)**	**Average orientation error (Y axis) (º)**	**Standard deviation (º)**
Slow	**Steady**	1.17	0.29	5.93	3.63	6.79	5.76	5.93	4.28	0.38	0.22
Slow	**Side**	1.72	0.11	7.07	5.83	9.34	9.85	5.49	6.23	7.13	3.32
Slow	**Obstruction**	2.37	1.27	13.40	8.99	15.34	4.01	104.66	53.18	120.21	62.80
Fast	**Steady**	0.78	0.48	8.27	7.38	11.82	8.34	8.38	7.89	7.35	7.57
Fast	**Side**	1.09	0.58	2.41	0.82	2.62	0.60	3.56	1.90	2.14	2.51
Fast	**Obstruction**	1.38	0.22	12.99	5.52	43.91	41.26	78.10	58.46	43.81	60.19

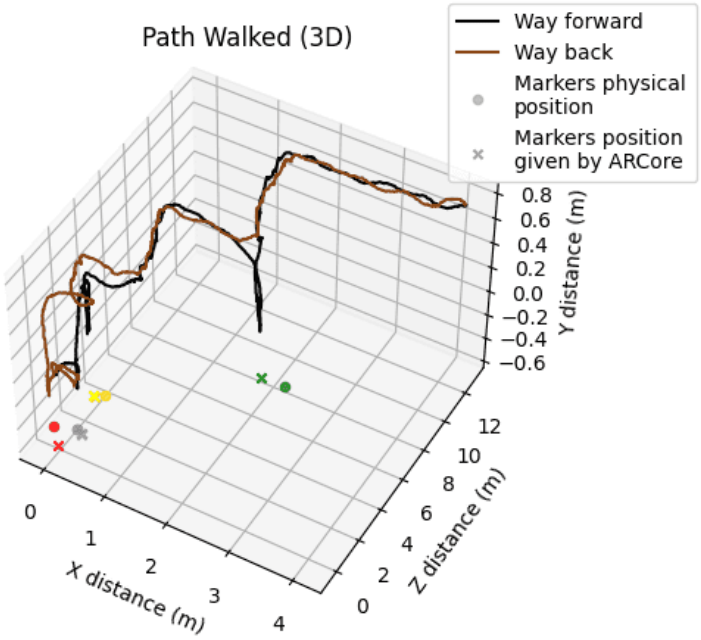
Table 3.4: Summary of Y-axis orientation error results in the IRIS Lab soccer field scenario.



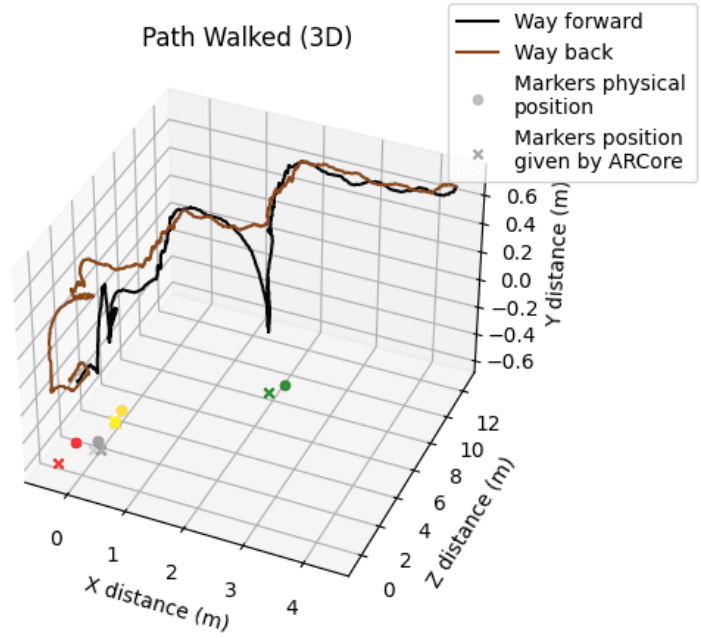
(a) 1º run: Steady method and slow walking speed.

(b) 10º run: Steady method and fast walking speed.

Figure 3.4: Two plot examples built with the device's camera coordinates over each frame in a run, using the steady method in the house scenario.

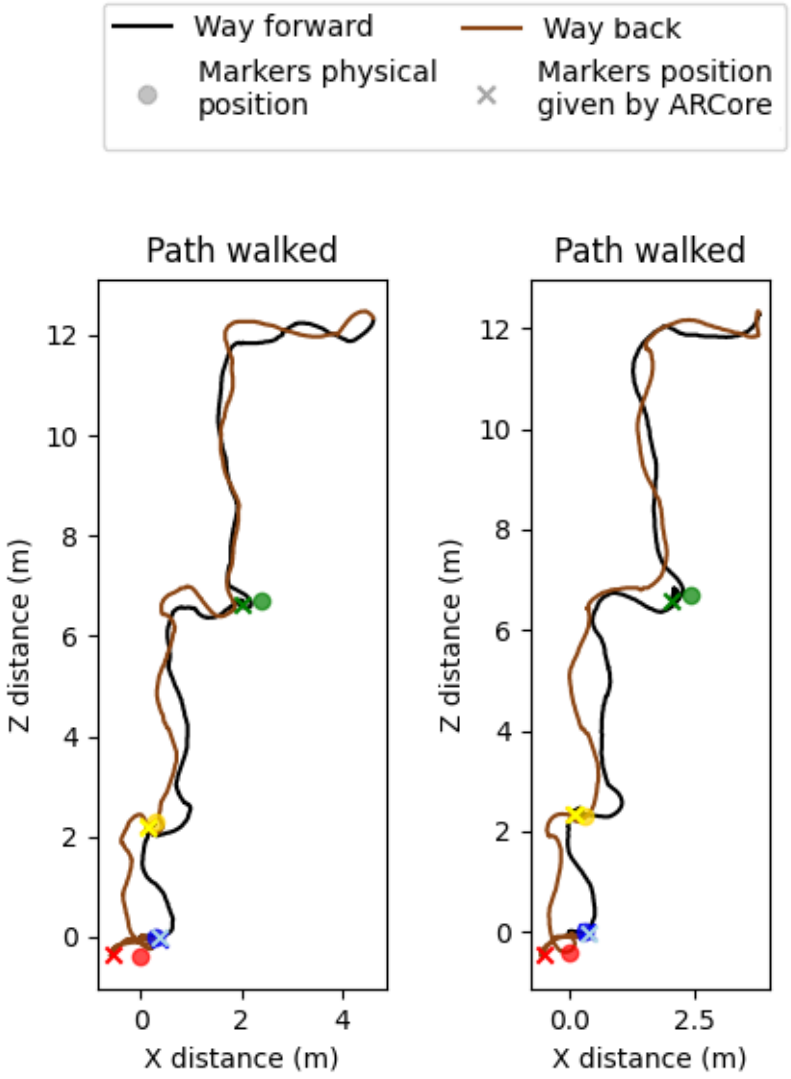


(a) 1º run: Steady method and slow walking speed.



(b) 10º run: Steady method and fast walking speed.

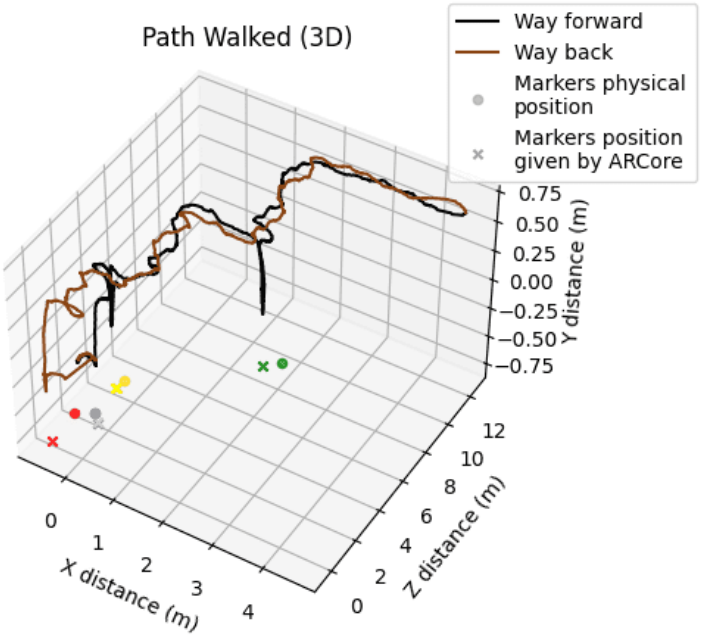
Figure 3.5: Two 3D plot examples built with the device's camera coordinates over each frame in a run, using the steady method in the house scenario.



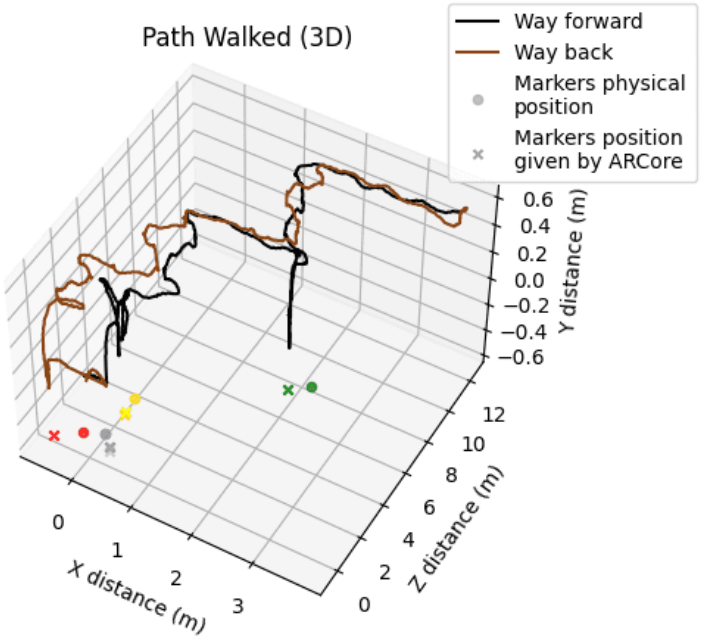
(a) 4º run: Side method and slow walking speed.

(b) 13º run: Side method and fast walking speed.

Figure 3.6: Two plot examples built with the device's camera coordinates over each frame in a run, using the side method in the house scenario.

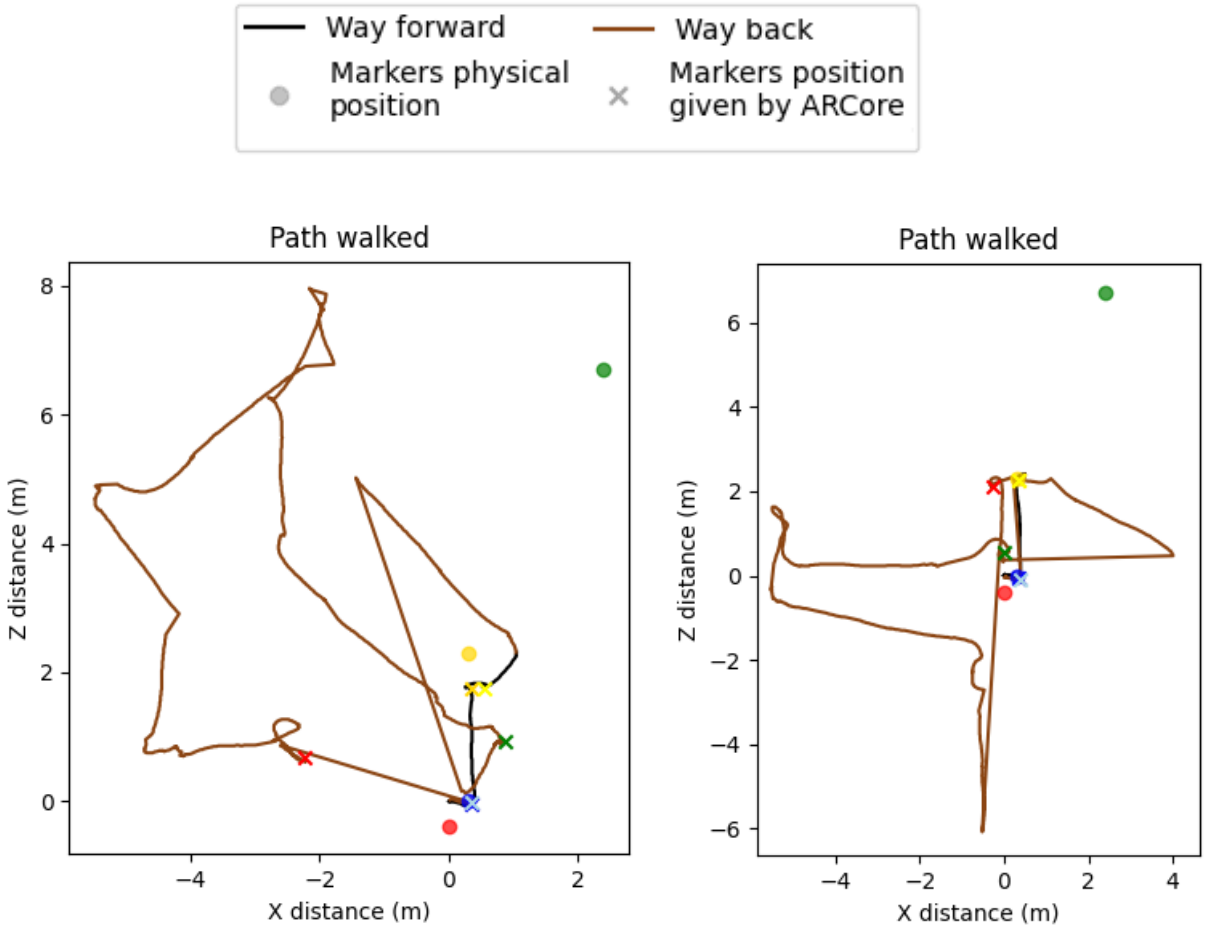


(a) 4º run: Side method and slow walking speed.



(b) 13º run: Side method and fast walking speed.

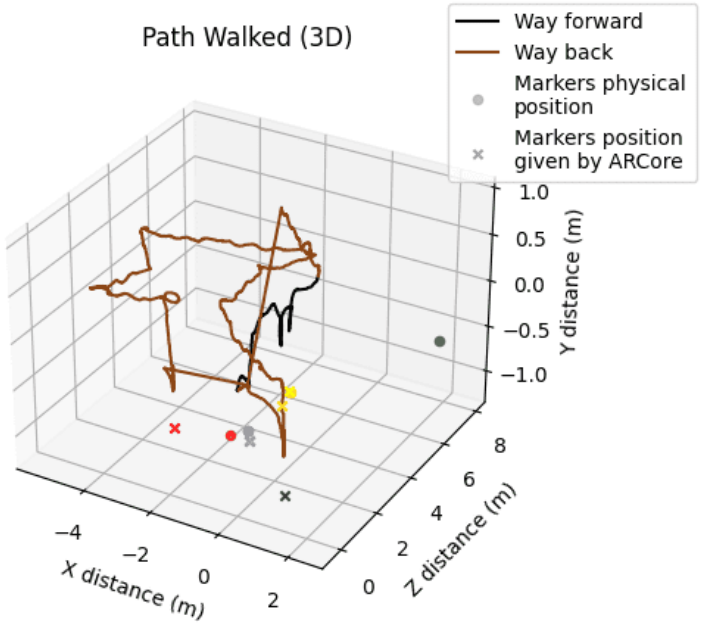
Figure 3.7: Two 3D plot examples built with the device's camera coordinates over each frame in a run, using the side method in the house scenario.



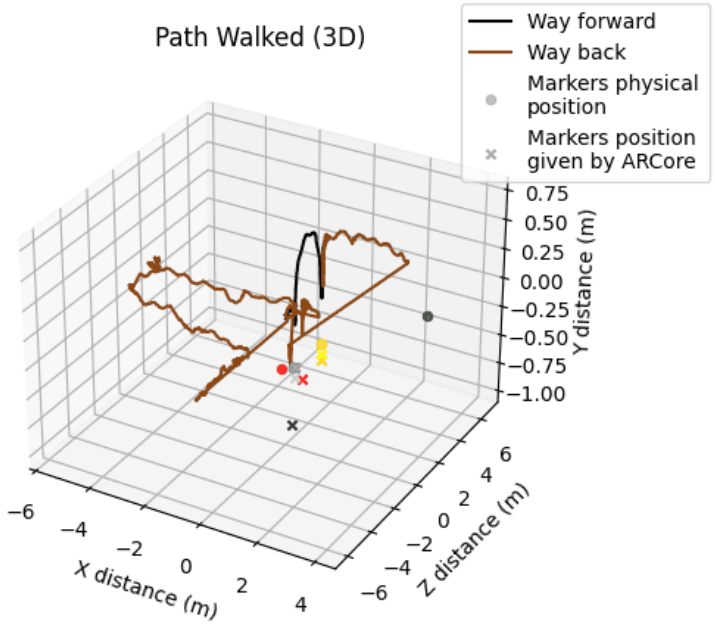
(a) 7º run: Obstruction method and slow walking speed.

(b) 16º run: Obstruction method and fast walking speed.

Figure 3.8: Two plot examples built with the device's camera coordinates over each frame in a run, using the obstruction method in the house scenario.

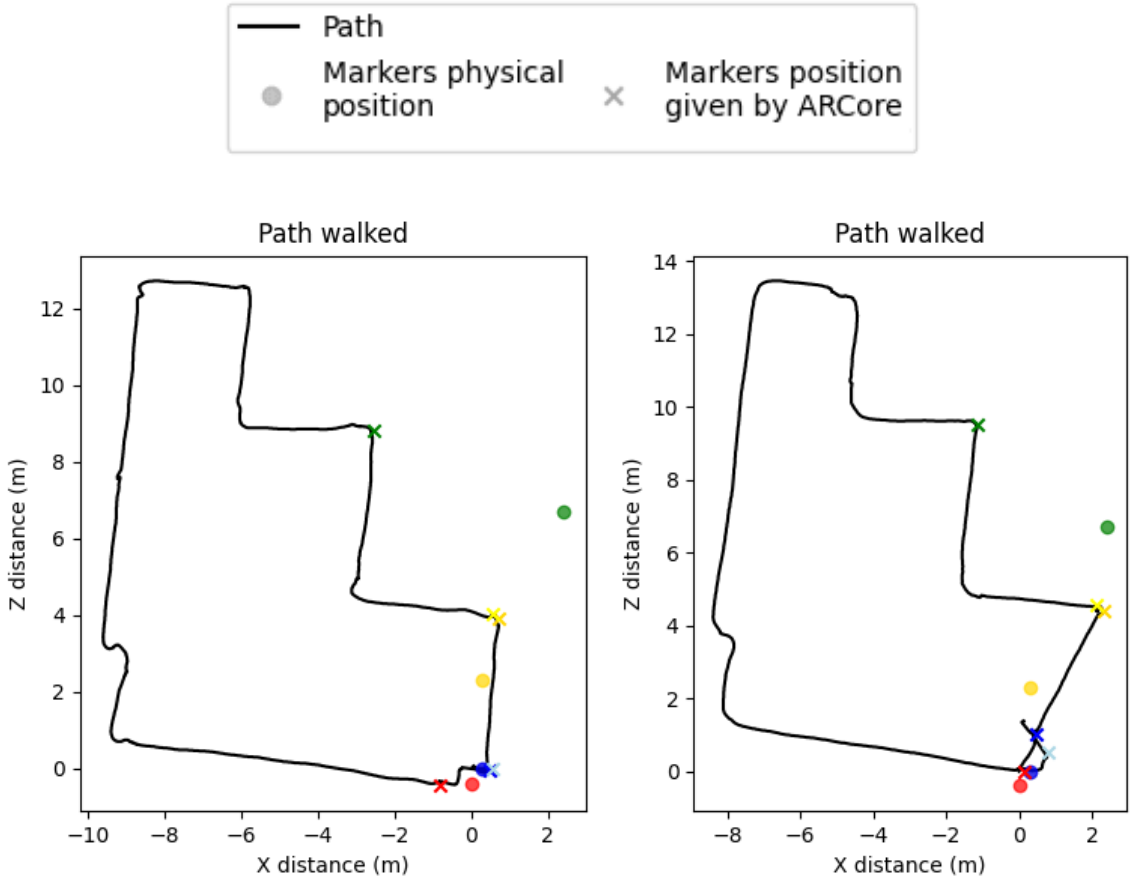


(a) 7º run: Obstruction method and slow walking speed.



(b) 16º run: Obstruction method and fast walking speed.

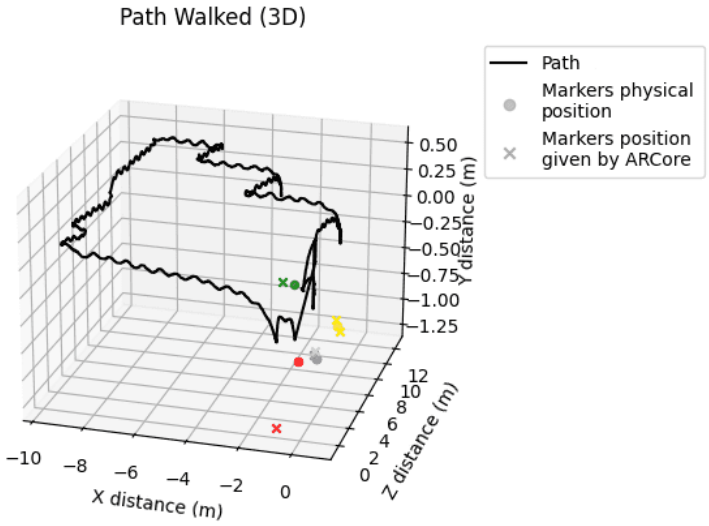
Figure 3.9: Two 3D plot examples built with the device's camera coordinates over each frame in a run, using the obstruction method in the house scenario.



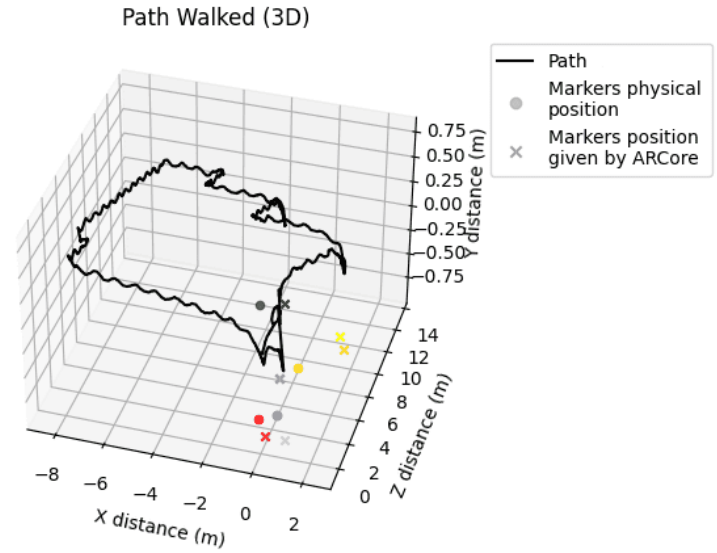
(a) 1º run: Steady method and slow walking speed.

(b) 10º run: Steady method and fast walking speed.

Figure 3.10: Two plot examples built with the device's camera coordinates over each frame in a run, using the steady method in the IRIS Lab soccer field scenario.

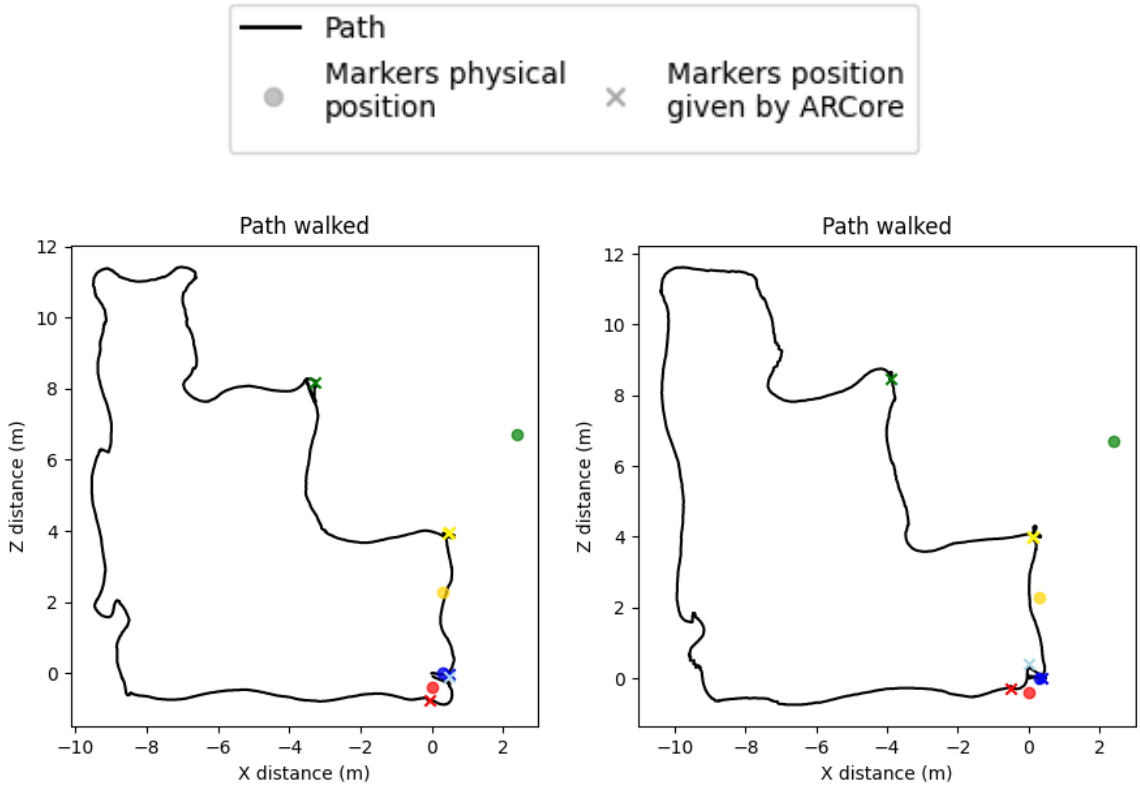


(a) 1º run: Steady method and slow walking speed.



(b) 10º run: Steady method and fast walking speed.

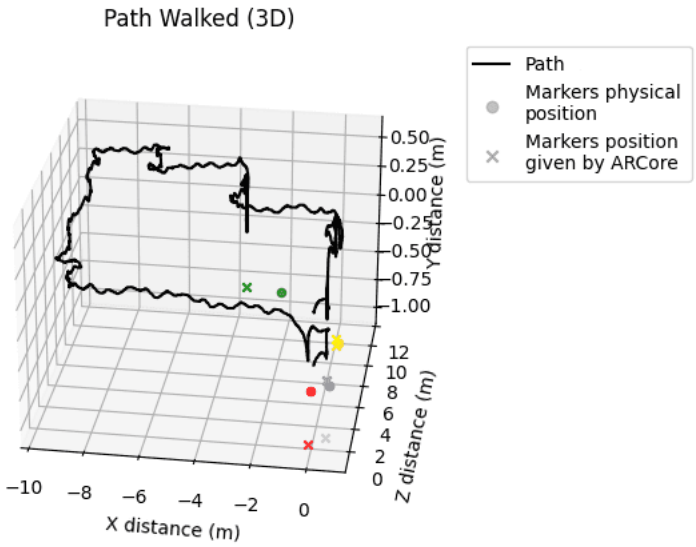
Figure 3.11: Two 3D plot examples built with the device's camera coordinates over each frame in a run, using the steady method in the IRIS Lab soccer field scenario.



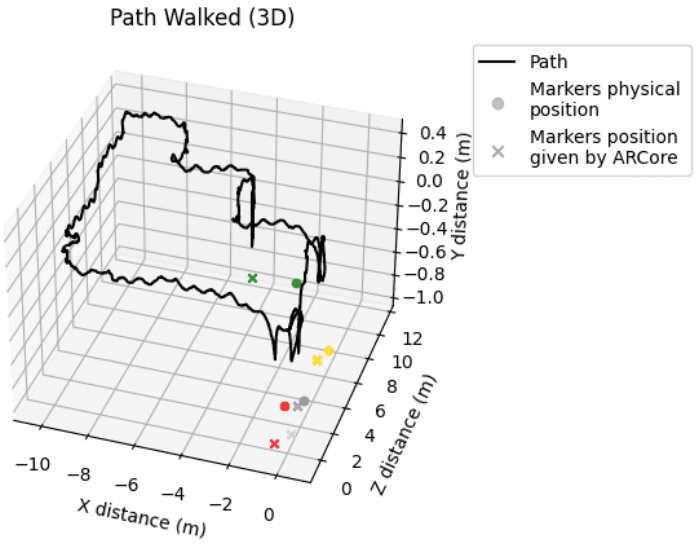
(a) 4º run: Side method and slow walking speed.

(b) 13º run: Side method and fast walking speed.

Figure 3.12: Two plot examples built with the device's camera coordinates over each frame in a run, using the side method in the IRIS Lab soccer field scenario.

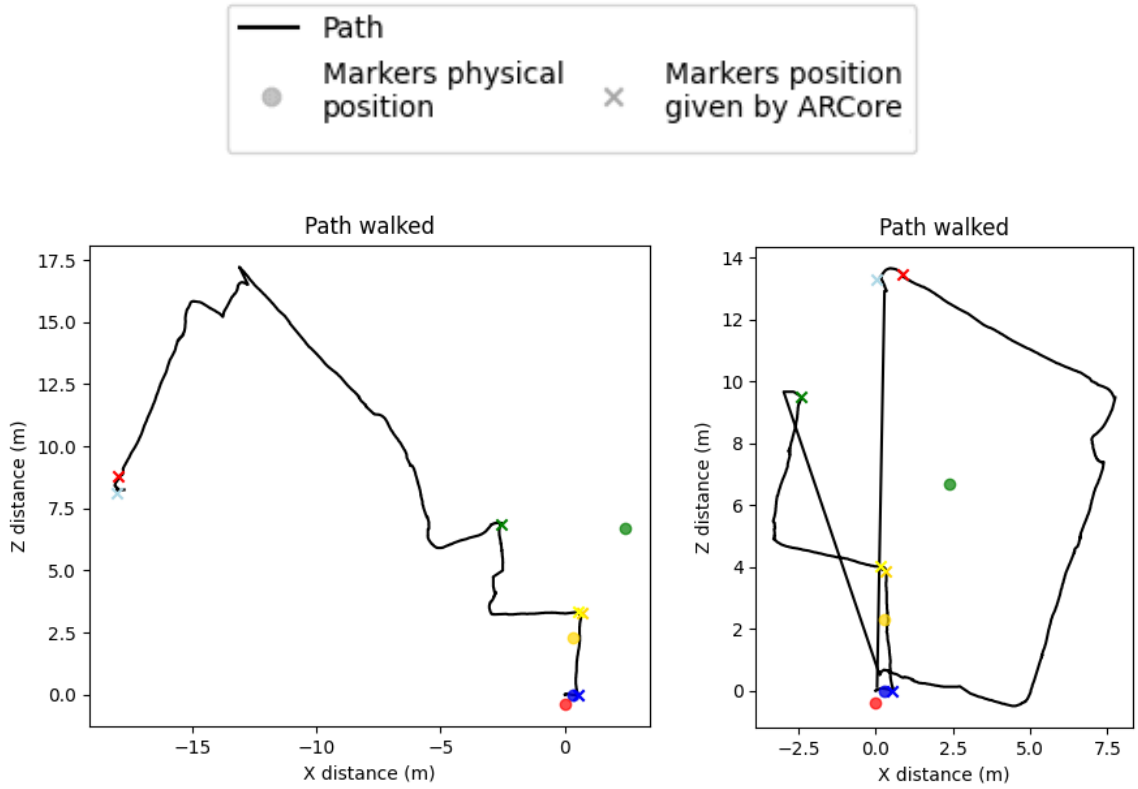


(a) 4º run: Side method and slow walking speed.



(b) 13º run: Side method and fast walking speed.

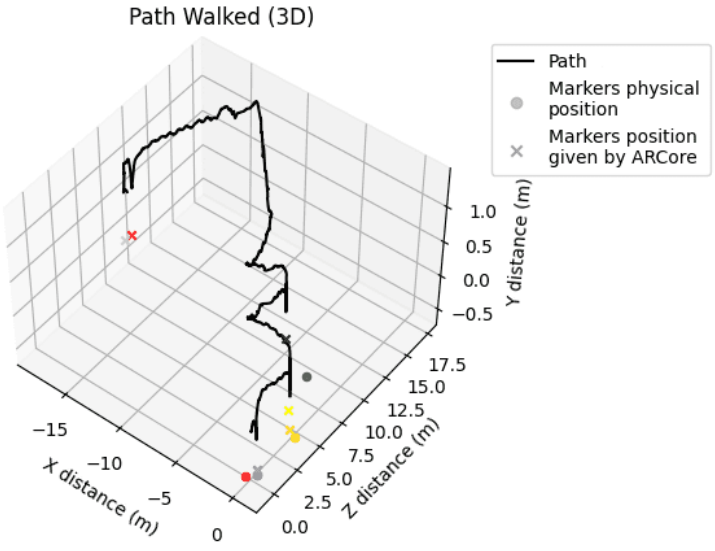
Figure 3.13: Two 3D plot examples built with the device's camera coordinates over each frame in a run, using the side method in the IRIS Lab soccer field scenario.



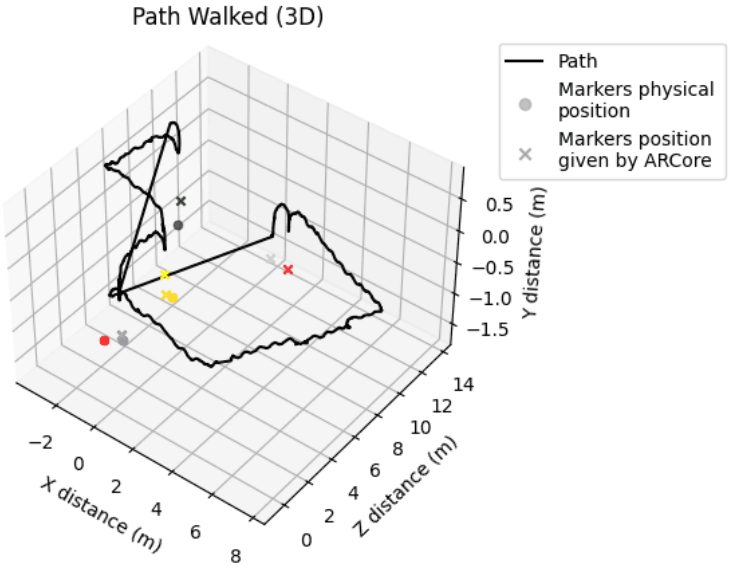
(a) 7º run: Obstruction method and slow walking speed.

(b) 16º run: Obstruction method and fast walking speed.

Figure 3.14: Two plot examples built with the device's camera coordinates over each frame in a run, using the obstruction method in the IRIS Lab soccer field scenario.



(a) 7º run: Obstruction method and slow walking speed.



(b) 16º run: Obstruction method and fast walking speed.

Figure 3.15: Two 3D plot examples built with the device's camera coordinates over each frame in a run, using the obstruction method in the IRIS Lab soccer field scenario.

3.2.3 Discussion

In order to draw some conclusions from this experiment, the obtained results, in both scenarios, had to be statistically analyzed to verify the produced drift in the motion tracking estimation during the application execution, as well as to verify which method stands out as the best and whether the user's walking speed has any notable influence on this ARCore technology module. Some charts were built to facilitate the analysis and support the results presented in the above section. For each intended verification, we started by analyzing the output variable for the position error and then the one for the orientation error around the Y axis.

Looking to the 3d charts (figures 3.5, 3.7, 3.11 and 3.13), one of the things that stands out right away is that, in the place of the markers, there is a rise and fall in the position of the device's camera relative to the Y axis. This is due to the ARCore Augmented Images technology not being able to recognize the marker only pointing to it upon arrival at its location, which required to move the device towards the marker to collect the corresponding results. This movement leads in lowering the device, resulting to changes in the Y axis. Although, the image recognition not having the intended range level, it seemed stable and was a great help in the developed experiment.

Obstruction Method Analysis

The first evidence that was quickly observable, was the poor motion tracking performance using the obstruction method. In both scenarios, the two output variables had terrible results, the average distance error varying between 0.18m and 0.69m on the second marker, reaching, in the fourth marker, 6.60m in the house scenario and 14.31m in the IRIS Lab soccer field scenario and the average orientation error not having a big error until the first device rotation around the Y axis in the experiment run, however, after a rotation obstructing the device's camera, the orientation error starts to be considerable, going up to more than 75º in both scenarios after all path rotations being performed. Analyzing figures 3.8 and 3.14, showing the path drawn using the device's camera coordinates using the obstruction method, it can be observed that in a certain point, the motion tracking technology is completely lost, starting to write a totally different path and positioning the markers far away from their real positions. In figures 3.9 and 3.15, we can observe the same, but also that these positioning and orientation losses can result in errors at the height of the device (Y distance).

This behavior is justifiable because the ARCore motion tracking technology uses visually distinct features to compute its change in location and in the occasions where the camera is obstructed, it stops detecting the scene feature points, which leads to a momentary pause of tracking. Looking at figure 3.16 which corresponds to the same run as figures 3.8 and 3.9, it is possible to verify that the application, in 15 points, wrote over a significant number of frames the equal device's camera pose. In the steady and in the side method the same does not happen, which allows us to claim that these are the moments when the obstruction occurs and where the application gets effectively lost.

Since this method presents unfavorable results and it is not possible to draw the planned conclusions applying this method, it will be discarded from the remaining evaluation analysis.

Steady and Side Methods Analysis

Despite the obstruction method presenting poor results, the steady method and the side method, in both walking speeds, offer favorable results. Analyzing figures 3.4 and 3.5, relating to the experiment in the house scenario, and comparing this plot examples with the real traveled path, figure 3.3, we can observe that using the steady method, the motion tracking technology perceived the route that was being taken, finding enough feature points in the scenario texture. Still in the steady method, observing

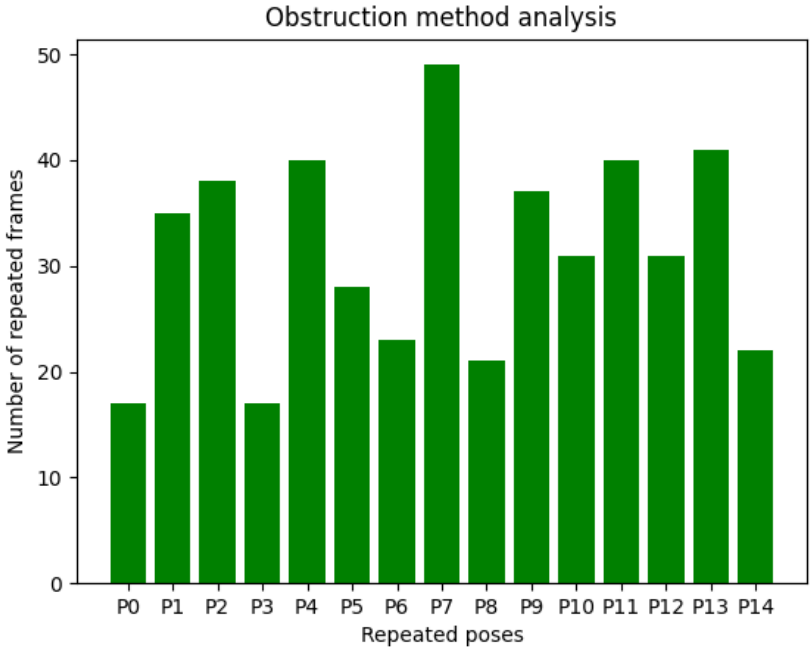


Figure 3.16: 7º run in the house scenario: Example of the number of sequentially repeated frames displaying the exact equal pose over the entire run.

---

and comparing figures 3.10 and 3.11 with figure 3.2, now over the IRIS Lab soccer field scenario, it is also possible to declare that the drawn path line by the ARCore application is identical to the walked one. However, in contrast with the house scenario, on the IRIS Lab soccer field in the 3D plots (figure 3.11) there is a considerable error in the device's height (Y distance) along the path. Looking now to tables 3.1 and 3.3 relating to the average distance error, and focusing on the steady method, it can be noted that the error is mostly light, having the maximum average error at the fourth marker in the house scenario of 0.35m and a standard deviation of 0.21m and in the IRIS Lab soccer field scenario a maximum average error at the fifth marker of 0.51m and a standard deviation of 0.35m, both practicing the fast walking speed. Relatively to the orientation error, presented in tables 3.2 and 3.4, the error is also not highly significant, in general not reaching an average of 10º, with the maximum average orientation error, at the house scenario, in the second marker after the rotation, of 14.93º but a standard deviation of 14.11º, which can indicate that the average result comes from a bad run compared to the remaining two runs.

Proceeding now to the side method we analyzed it in the same way as the steady method, firstly looking to figures 3.6 and 3.7 in comparison to figure 3.3, we can observe that the drawn path with the coordinates given by ARCore is relatively equivalent to the one performed by foot. The same applies when comparing figures 3.12 and 3.13 with figure 3.2. The main visual difference between the side method and the steady method is the wavy lines using the side method, consequence of the method execution, that implied the motion of the device to the sides while walking. This result allows to assert that the motion tracking technology was not just aware of the walking route, but also of the device's movement. Going into more detail and analyzing the average errors using this method, examining the average distance error tables 3.1 and 3.3, we verify a maximum average error, in the house scenario administering the fast walking speed, at the third marker, of 0.44m and a standard deviation of 0.23m

and of 2.1m and a standard deviation of 2.11m at the fifth marker applying the slow walking speed, in the IRIS Lab soccer field scenario. The average distance error in the house scenario is acceptable, however, in the IRIS Lab soccer field, employing the fast walking speed, the error is large. This is due to a one time run with a high error, as can be seen from the also high standard deviation, but since only three runs were performed, this large error cannot be despicable and considered an outlier. Regarding the average orientation error, it does not reach the 10^{o} in both scenarios, which is quite satisfactory.

Hereupon, these two methods seem acceptable for the drift analysis and the walking speed comparison. In addition, we will also check which of the two is better.

Drift Analysis

A common performance parameter in AR applications that use some source of tracking technology is the drift. Drift is the increase of the tracker error with time and distance. As ARCore motion tracking technology is no exception, it also suffers from drift and needs to be analyzed.

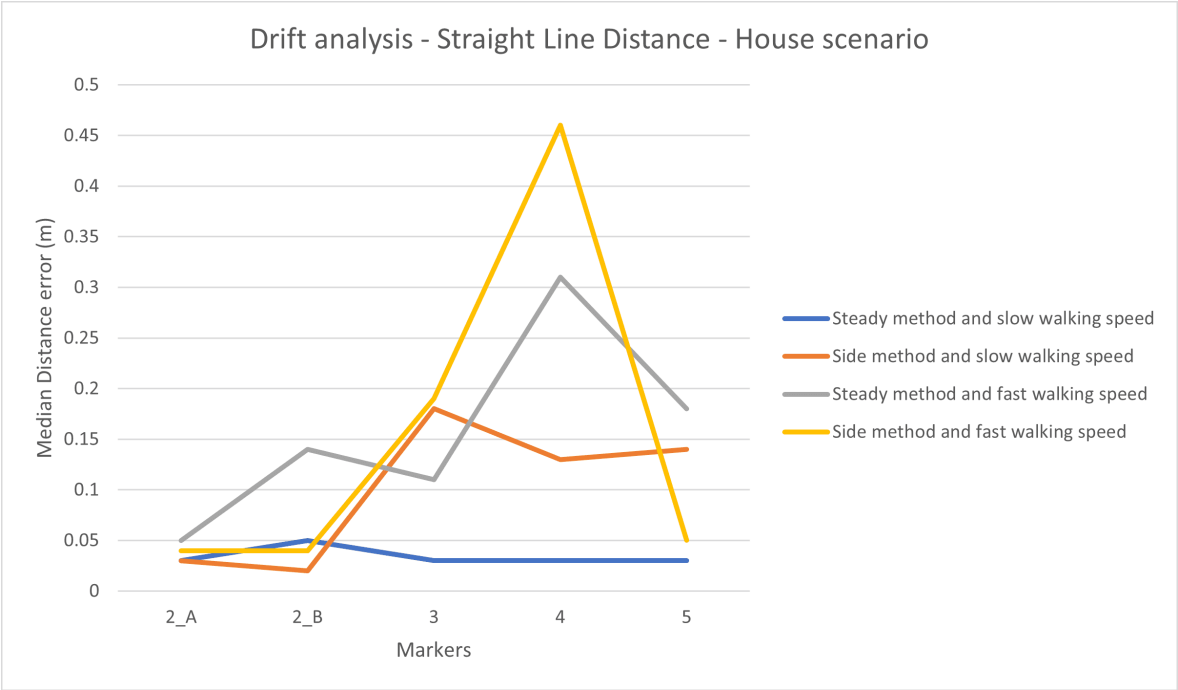
Starting by analyzing the position drift, we draw two charts, one for the House scenario (figure 3.17a) and one for the IRIS Lab soccer field (figure 3.17b). These charts were built using the median result for the distance error for each method in each walking speed in each of the markers.

Focusing on figure 3.17a, it is possible to observe that with the course of the experiment, the distance error tends to increase with the time and walked distance to reach the next marker, however, this is not a norm. As we can note, in the steady method and slow walking speed the error decreases from marker 2_B to marker 3, maintaining this error along marker 4 and 5. In the steady method and fast walking speed there is also an error decrease from marker 2_B to marker 3, increasing the error from it to marker 4. The side method and slow walking speed has a decrease in the error from marker 3 to marker 4. As for the last, the side method and fast walking speed, is the one in which the error tendency to increase is more observable. The transition between the fourth and fifth marker is mostly a decrease, since as referred in section 3.2.1, the ARCore application often automatically corrects its device's camera pose based on a marker position that was previously detected.

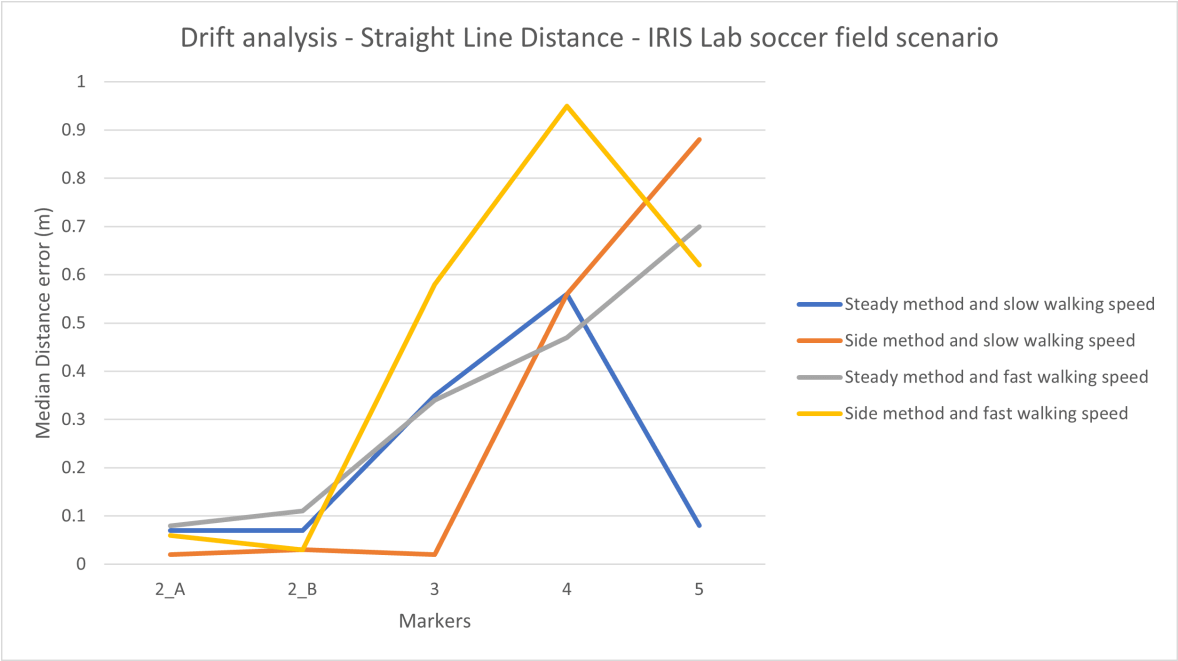
Considering now figure 3.17b, the position drift increase between marker 2_B and marker 4 is more visible as in this scenario, all the median distance errors, using the two methods applying each walking speed, intensify. The only inconsistency here is seen in the side method and slow walking speed, between marker 2_B and marker 3, where the error slightly decreases.

For the study of the orientation drift around the Y axis, two identical charts to those mentioned above were made, with the difference that these were created using the median error in the results of the Y orientation. The orientation drift analysis in the house scenario is seen in figure 3.18a and regarding the IRIS Lab soccer field scenario in figure 3.18b.

Considering first the house scenario, it is verifiable that at the second marker, after the first rotation (transition between marker 2_A and 2_B), the median orientation error slightly increases in all methods and walking speeds. This is also valid between marker 3 and marker 4, although the third marker presents a smaller error than the second marker, before and after the rotation and the fifth marker also presents a smaller orientation error than the fourth one. With respect to the IRIS Lab soccer field scenario, there is also a minor increase in the error after the first rotation occurs in both methods and walking speeds, however for the remaining marker transitions, the drift is not linear. Between marker 2_B and marker 3, the error has a large grow using the steady method and fast walking speed, in contrast with the steady method and slow walking speed that shows a big error reduction, the side method applying both walking speeds keep a similar error. Arriving to the fourth marker, the errors corresponding to the slow walking speed do not experience considerable changes, the steady method and fast walking speed error shows a large decrease and the side method and fast walking speed error



(a) House scenario.

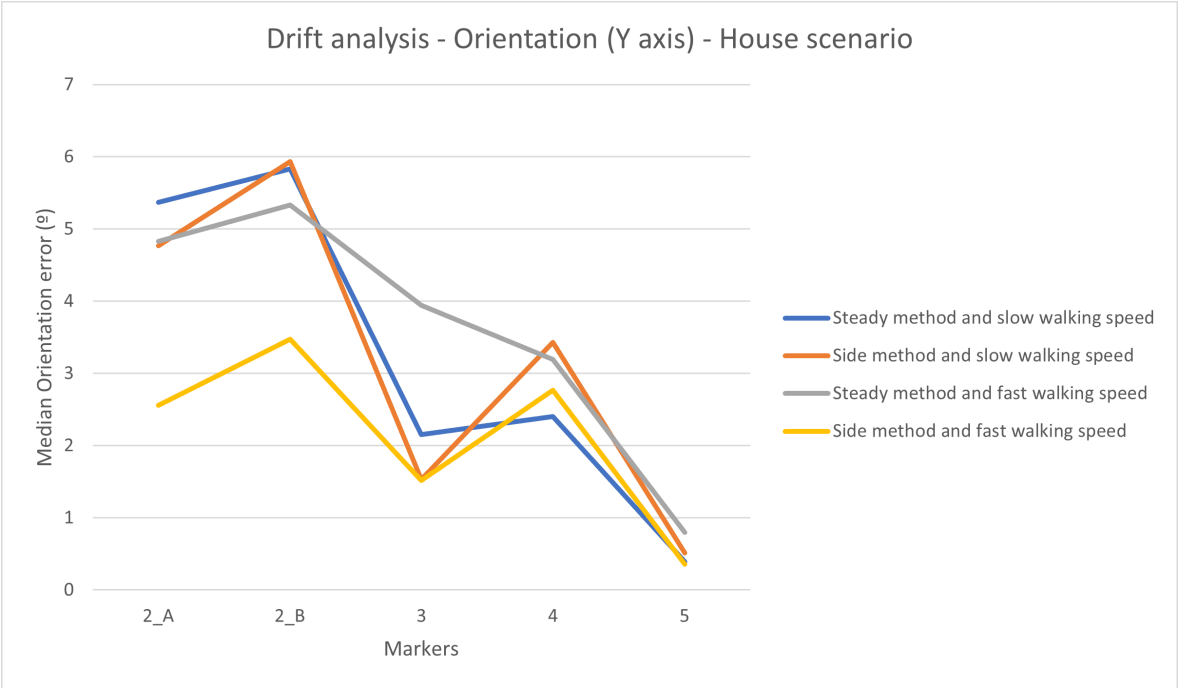


(b) IRIS Lab soccer field scenario.

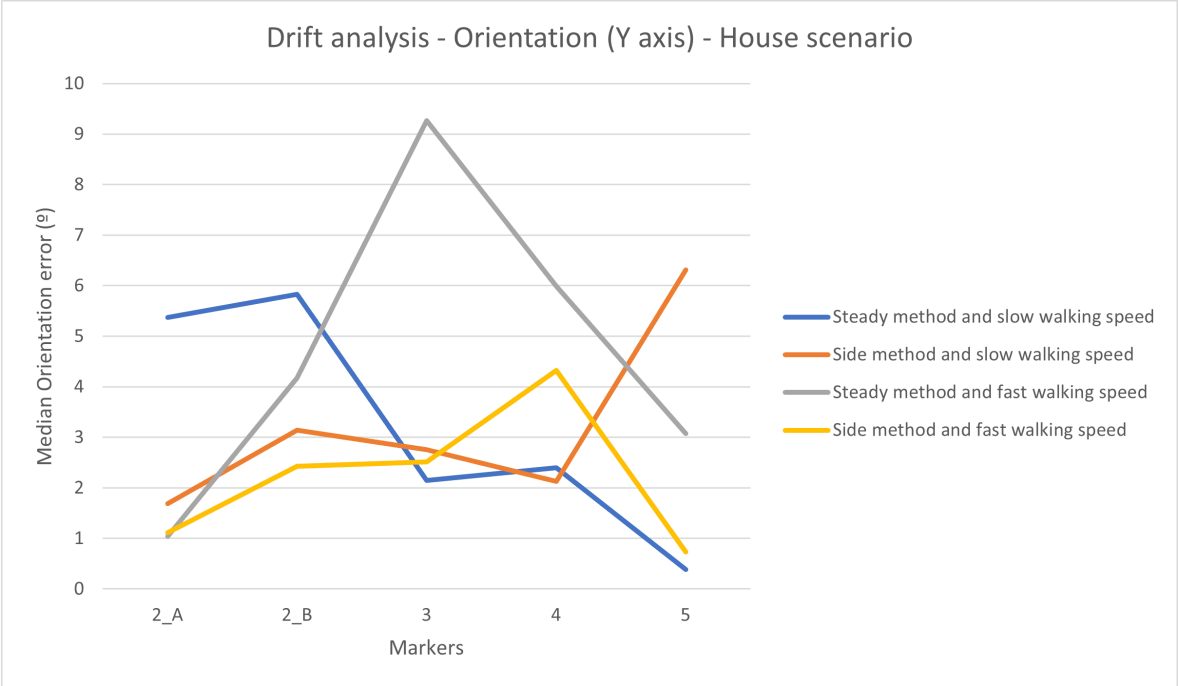
Figure 3.17: Evolution of the median distance error over each marker.

increases. Finally at the last marker, only the side method applying the slow walking speed increases the orientation error, the remaining three approaches decrease it.

In distinction to the position drift, which is accumulated with time and the walked distance and



(a) House scenario.



(b) IRIS Lab soccer field scenario.

Figure 3.18: Evolution of the median orientation error (on the Y axis) over each marker.

can cause inconveniences traveling long distances, the orientation drift may arise, but it probably will not be an issue since rotating within the environment makes the drift increase and decrease with time not reaching a large and significant error.

Walking Speed Analysis

To simplify the walking speed influence analysis, we assembled four bar charts. Figure 3.19a and 3.20a regarding the average position error and the average orientation error, respectively, in the house scenario. The remaining two charts, included in figure 3.19b and 3.20b, refer to the average position error and the average orientation error, respectively, in the IRIS Lab soccer field scenario. To accomplish this study, the slow and the fast walking speed will be compared using the steady method and the side method, in both scenarios.

Beginning by analyzing figure 3.19a, which contains the results in the house scenario, we can observe that markers 2_A, 2_B, 3 and 4 present a larger average distance error applying the fast walking speed than applying the slow walking speed, both using the steady and the side methods. Marker 5 presents an equivalent error using the steady method in the two walking speeds and again a bigger error practicing the side method and fast walking speed.

Subsequently, figure 3.19b, in the IRIS Lab soccer field scenario, suggest similar outcomes. In marker 2_A and 2_B the usage of the steady method demonstrates identical average distance errors and subtly smaller errors administrating the slow walking speed. In the third, fourth and fifth markers, the slow walking speed also had minor errors, but in a more visible way. Regarding the side method, in markers 2, 3 and 4, the difference is massive due to the run with bad results applying the fast walking speed, favoring the slow walking speed approach. At the last marker, using the side method, the fast walking speed presents a lower error, resulting from the ARCore automatic correction when the fifth marker (which is also the first one) is recognized again.

Concerning now the average orientation error in the house scenario, figure 3.20a, the marker 2_A is not highly significant since no rotations occurred. Markers 2_B, 3 and 4 present better results applying the slow walking speed in the steady method (i.e. smaller orientation errors). The fifth marker shows the opposite, result from the ARCore motion tracking adjustment again. Using the side method, marker 2_B reveals a larger error with the slow walking speed, marker 3, 4 and 5 present slightly larger errors in the fast walking speed.

Lastly, we will analyze figure 3.20b about the IRIS Lab soccer field scenario average orientation errors. For the reason referred above, marker 2_A is relatively irrelevant. Regarding the remaining four result points, using the steady method, the slow walking speed demonstrated smaller orientation errors in all markers and by contrast, the side method exhibited better orientation results applying the fast walking speed in every marker.

This enables us to assume that the slow walking speed is better in both methods to decrease the position error and with respect to the orientation error, the slow walking speed stood out as the best speed for the steady method, but for the side method, the fast walking speed probably would not cause any major issues.

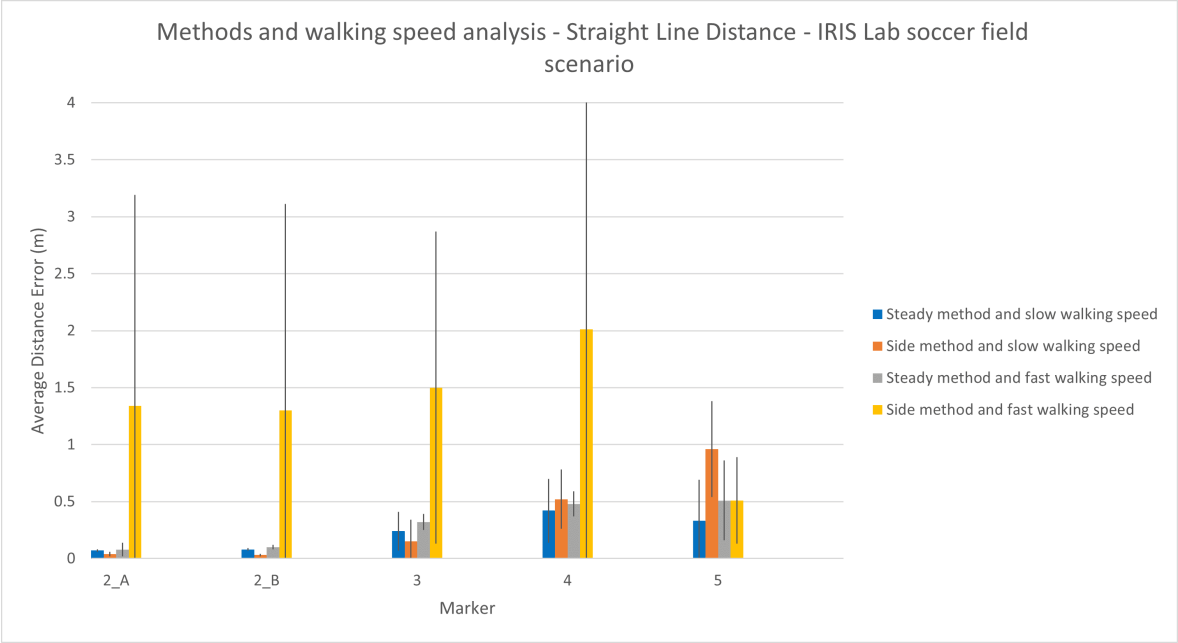
Best Method Analysis

The steady and the side methods seem appropriate, obtaining satisfying results, but which one is better? To answer this question, we will lean over on figures 3.19a, 3.19b, 3.20a and 3.20b again, this time, comparing both methods over the same walking speed.

Interpreting figure 3.19a, we can verify that adopting the slow walking speed, both methods present similar average distance errors in marker 2_A, 2_B and 5, however in marker 3 and 4, the steady method had smaller errors in comparison to the side one. Now for the fast walking speed, with exception of marker 2_B, the steady method revealed better results.

For the average distance error in the IRIS Lab soccer field scenario, figure 3.19b, applying the

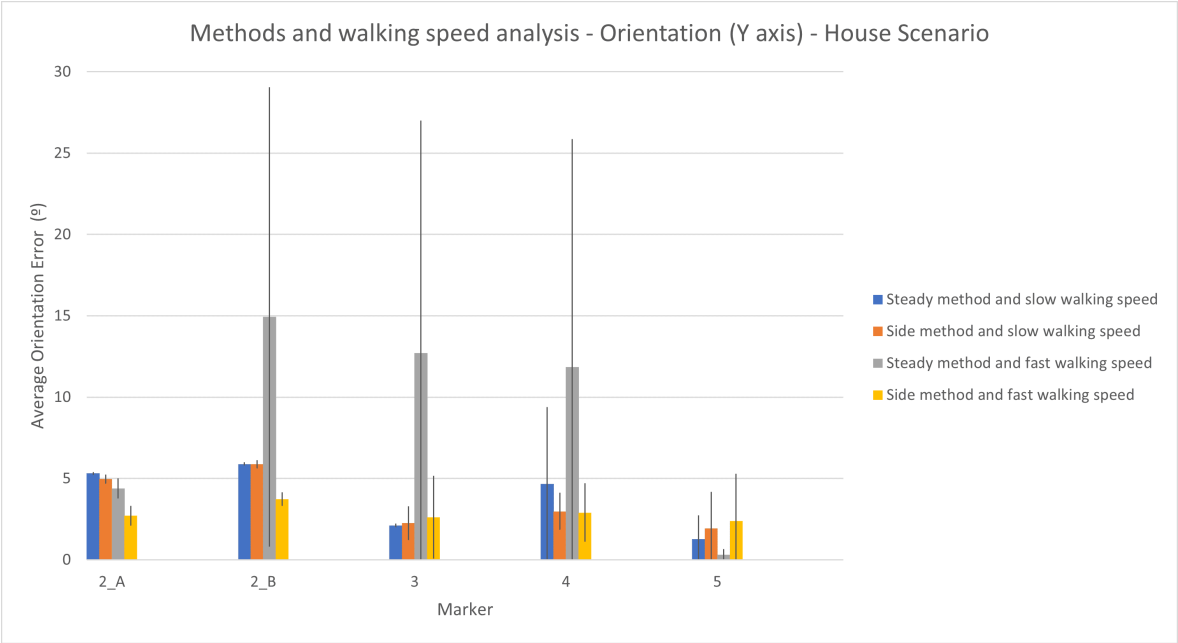
(a) House scenario.



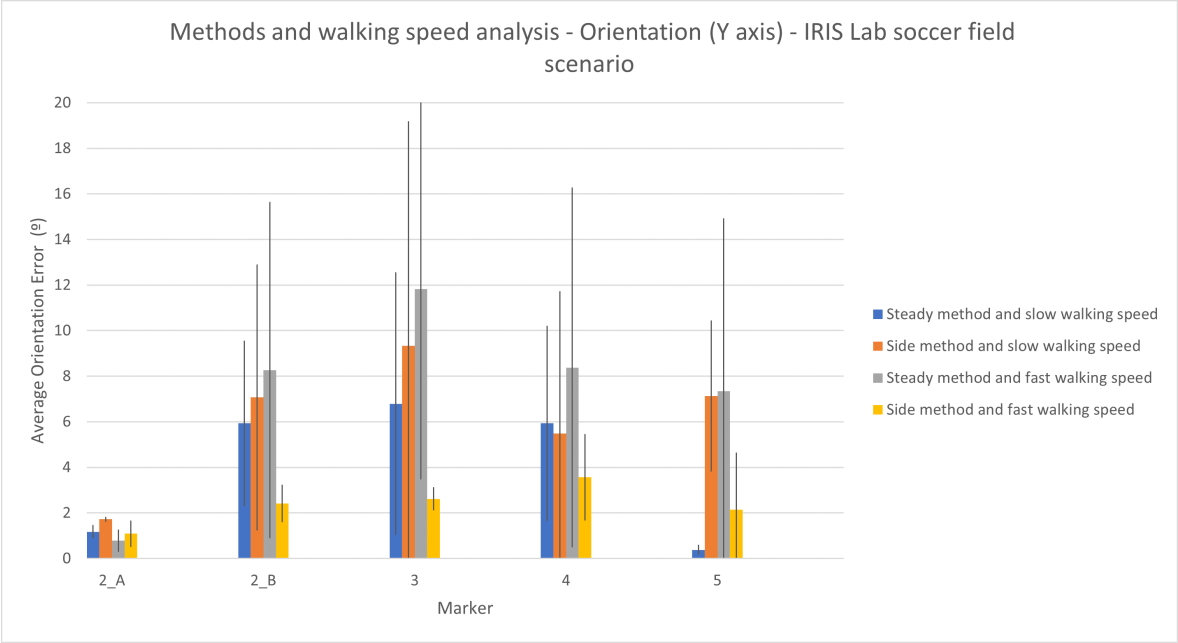
(b) IRIS Lab soccer field scenario.

Figure 3.19: Bar chart containing the average distance error concerning each method and walking speed for every marker.

slow walking speed, we can note that until the fourth marker being reached, the side method presented the smaller average error, however, in the fourth one, the steady method already presents the smallest error. Looking now at the fast walking speed, the steady method shows a largely lower average error in relation to the side method (the enormous difference is due to a bad run using the side method and



(a) House scenario.



(b) IRIS Lab soccer field scenario.

Figure 3.20: Bar chart containing the average orientation error in the Y axis concerning each method and walking speed for every marker.

fast walking speed).

Regarding the average orientation error, starting by analyzing the results in the house scenario, figure 3.20a, we can observe that in marker 2_B and 3, the error is identical between both methods, slightly lower using the steady one and in marker 4 the steady method present a higher orientation

error. For the fast walking speed, the steady method shows worse results than the side method in every marker (except the fifth one, result from the motion tracking correction). Similar results are found in the IRIS Lab soccer field scenario, figure 3.20a, where the application of the slow walking speed result in better average orientation errors using the steady method in markers 2_B and 3, what no longer happens at marker 4, where the side method has a slightly smaller error. Regarding the fast walking speed, the side method presents smaller errors in all markers after one rotation again.

Therefore, we can infer that for shorter distances both methods are similar, however for longer distances the steady method will be preferred, as it results in minor position errors. The side method would probably be a better choice in applications where the orientation is much more important and the walking speed is high. From a usability and interaction standpoint, the steady method is preferred, being more natural and allowing a better visualization of virtual objects along the path (in case these objects are not in the sides of the path).

Considerations

Conducting this experiment allowed us to draw some conclusion about ARCore motion tracking technology, in which cases it can be used and how to improve its use. In general, excluding the obstruction of the camera when the application is executing, ARCore motion is reasonably robust, although not entirely accurate, but it might be used in applications where exact precision is not required. In large scenarios where a user needs to travel extensive distances, it may not be an adequate technology since the drift increases with the walked distance. However, if the distance to be walked is performed in a circular way, meaning that the user passes through the same place, ARCore offers an automatic way to correct its device's camera pose, that was affected over time.

For our application use case, we decided to use a folding smartphone arm support attached to the robotic wheelchair, close to the left driving wheel, in a way that the smartphone device would rest in a stable position in front of the user, being able to adjust the smartphone position depending on the user's height and preference (figure 3.21). Thus, the user does not need to hold the smartphone, preventing him from obstructing the camera with his hand or fingers. Since the steady method stood out as better than the side method, it also helped with the device stability and device rotations following the wheelchair rotation. The application goal is for people with no experience in controlling robotic wheelchairs, so we assume that the user will apply a slow motion speed, corresponding to the first two speed levels of the power wheelchair (level 1 around 1.4km/h and level 2 around 4.2km/h), hence, the motion tracking error will be reduced as the slow walking speed excelled in relation to the fast one. Even after the user gains some control and comfort with the wheelchair maneuvers, that can make him increase the wheelchair speed to the next levels (7.2km/h or higher), the application can also be used, offering less accuracy but offering enough robustness anyway.

Hereupon, the motion tracking technology emerged as a possible alternative and junction to the cloud anchors technology for our application, removing the limitations induced by the latter one, such as the need to previously configure every anchor and the one-year cloud anchor host time. It is noteworthy, that it is required to have texture along the usage scenario and the more available texture, the better the robustness of the motion tracking technology.

Figure 3.21: TA IQ MWD wheelchair with the folding smartphone arm support attached.

Chapter 4

Augmented Reality Serious Game for Learning Robotic Wheelchair Maneuvers

In this chapter we will describe the ARSG concept, its architecture and specific modules, finishing this chapter with the conceived library for other designed Unity games to be played using ARCore in an AR environment.

4.1 Concept

As previously referred, people with reduced mobility can benefit from all the advantages that a robotic wheelchair may bring to them, but learning to use these wheelchairs may not be easy. Using the cross-platform game engine Unity, we developed highly adaptable configuration tools to set up an AR environment. Using these configuration tools and developing all the logic behind a serious game, we can use this application to visualize virtual objects in a real world scenario and play the game following and interacting with them. Furthermore, the user game performance and trajectory may be visualized in a desktop PC Unity application, that can help the user and health professionals to perceive which actions demand to be improved.

The developed serious game has the gameplay of a racing game, the AR field is composed of a curved track with one or many passage points that need to be passed by along the race. Two versions of this game were created, one for following a line fixed on the middle of the road with the support of directional arrows (figure 4.14a). In the second one, it is intended to follow a moving object, with the format of a car, that travels the path to its end (figure 4.14b). Both game versions have additional features, described below, to create a more enjoyable game. The work performed by (Liarokapis et al., 2005), described in section 2.2.2, indicated that most users preferred the more serious and educative game in relation to the one for entertaining purposes only, so every implemented feature, forces the users to interact with the virtual objects through wheelchair movements, which might also improve its learning, hence, making the game more pleasurable. These features are:

- Obstacles, that can be positioned close to or/and on the road. If the user fails to dodge one, the obstacle will be destroyed and an explosion will occur in the screen. This forces users to deflect the robotic wheelchair from its original course as they would have to proceed in the case of a real obstacle.

- The road area restriction, which limits the user to that space, preventing him from leaving

the road, unless an obstacle demands to exit it. Has the objective of learning to control the wheelchair in order to be within a certain path.

- A stop sign, which aims to obligate the user to quickly react and immobilize the wheelchair. The sign appears lying down, in less than two meters from the user if the device is pointing at the spot where it is located. Then the stop sign starts to rise, taking 5 seconds until being in its vertical position and this is the time that the user has to react and to stop the wheelchair.

- A spotlight that makes the user to speed up or to rotate the wheelchair to its opposite direction, training the users reaction and their decision-making. The spotlight virtually blinds the user, rendering a white image on the device's screen, if the user keeps the smartphone pointed at the spotlight.

The game can be configured in many and different scenarios, but as the power wheelchair is in the Robotic Laboratory of University of Aveiro it was only tested inside this scenario, more specifically in its soccer field.

4.2 Architecture

The developed application has two operating modes: using the ARCore cloud anchors technology (figure 4.1) and using the ARCore motion tracking technology (figure 4.2). These operating modes were implemented to have a more robust option (cloud anchors) and a second option with less limitations, more easy to use and that maintains a better relative position between virtual objects (motion tracking). Besides this, each operating mode is also divided into three modules, configuration, game and performance visualization, targeting distinct users (user X, Y and Z).

The configuration module is responsible for the placement of virtual objects around the real environment and it may be performed using the smartphone device or using the desktop PC, implying a calibration process. If using the desktop PC and the cloud anchors operating mode, it is necessary to initially place the anchors in the intended positions in the environment, in order to generate a file, which can be used later to associate and attach the virtual objects, configured via desktop PC, to these anchors.

On the other hand, the game stage, typically used by a different user (user Y in figures 4.1 and 4.2) from the one that uses the configuration component (user X in figures 4.1 and 4.2), will hold and access the configured content, stored in the user device application, so it may position the virtual objects (and the anchors, if the cloud anchors mode is chosen) around the already configured environment. This allows to play the game many times as long as the configured scenario does not change substantially.

Finally the last module, the performance visualization is responsible for storing the various user actions while playing, so after one or more game completions, the user (user Z in figures 4.1 and 4.2, which can be the player or a health professional) is able to watch his performance. To store the actions along the game, a message exchange system was elaborated adopting a client-server model, where the client is the smartphone, which in turn is responsible to send the various messages, via Wi-Fi, to the desktop PC application, acting as a server. These messages will be stored so that they can be read and processed by the Unity performance visualization application. Treating these messages it is possible to display the game practiced by the user.

Therefore, the system using the cloud anchors operating mode, schematized in figure 4.1, consists in the following process: User $\langle$X$\rangle$ starts by placing the desired cloud anchor(s) over the real environment and associate the virtual object(s) to it (them), using the Android device [A]. The cloud anchor(s) data

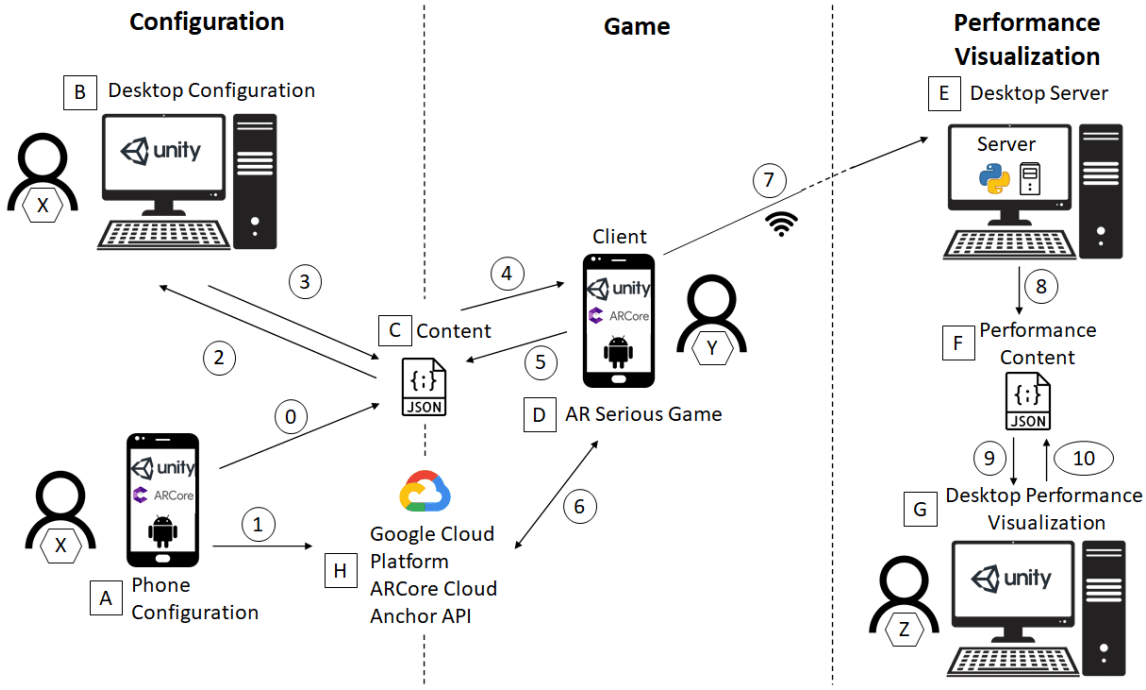


Figure 4.1: Application architectural schematic using ARCore cloud anchors technology.

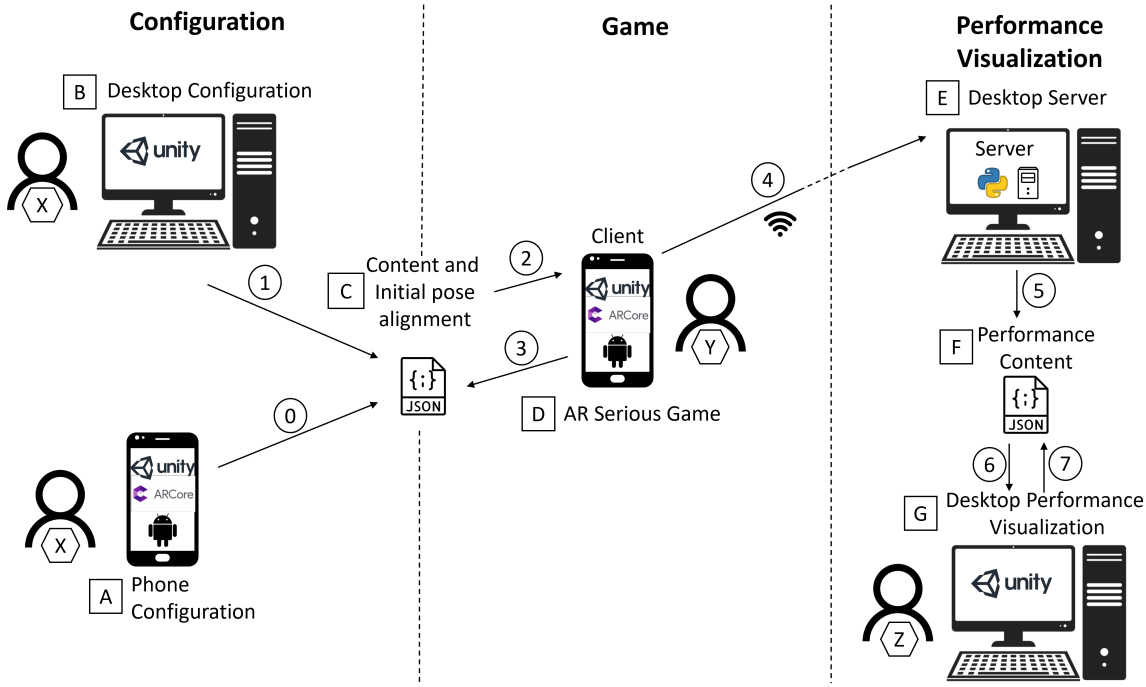


Figure 4.2: Application architectural schematic using ARCore motion tracking technology.

and its attached virtual object(s) information will be written (0) in a JSON file [C]. In addition, the

cloud anchor(s) will be hosted (1) in the GCP ARCore Cloud Anchor API [H]. This user can also use the desktop PC configuration to position the virtual object(s). To use the desktop PC configuration [B], user ⟨X⟩ has to transfer the content file (2), which stores the cloud anchor(s) data to the desktop machine application. Then, the cloud anchor(s) data is read and after the configuration of the virtual object(s), the cloud anchor(s) data together with its (their) associated virtual object(s) is written in a new file (3) so it can be transferred to the Android device application (4). Now, the user who will play the game (user ⟨Y⟩) can choose an AR scenario configured by user ⟨X⟩. The game module [D] will read the corresponding content file (5) in order to request the resolving of the anchor(s) to the GCP ARCore Cloud Anchor API, which will return the anchor(s) pose(s) if the resolving request succeeds (6). The anchor(s) will be placed in the corresponding pose(s) with the attached virtual object(s). As the game is played, the Android application is sending messages through Wi-Fi (7) to a desktop PC server [E], which in turn, stores these messages (8) in a JSON file [F]. Lastly, user ⟨Z⟩ can transfer the file with the messages stored (9) to the desktop PC performance visualization application [G] which will read and process the massages (10), in order to display the user's performance throughout the game.

Although the motion tracking operating mode has some similarities to the one described above, it also has differences. Its process, schematized in figure 4.2 is the following: User ⟨X⟩ will configure the virtual object(s) using the Android device [A] (0) or the desktop PC [B] (1), which will generate a file [C] containing the virtual object(s) information. If using the desktop PC for the configuration, the file has to be transferred to the Android application (2). Then, user ⟨Y⟩ can choose the configured AR scenario and align its Android starting position [C] to make it match with the configuration starting position. Next, the game module [D] will place the virtual object(s) over the real environment (3). Equally to the cloud anchors operating mode, the game will send messages (4) to the desktop PC server [E] which will treat them the same way for the same purpose (5) (6) (7).

4.3 Development

In this section, each module and component module will be described, detailing their behavior, technical aspects and communication with the associated components. The complete UIs for the smartphone application and for the desktop PC application are present in appendix sections C.1 and C.2, respectively.

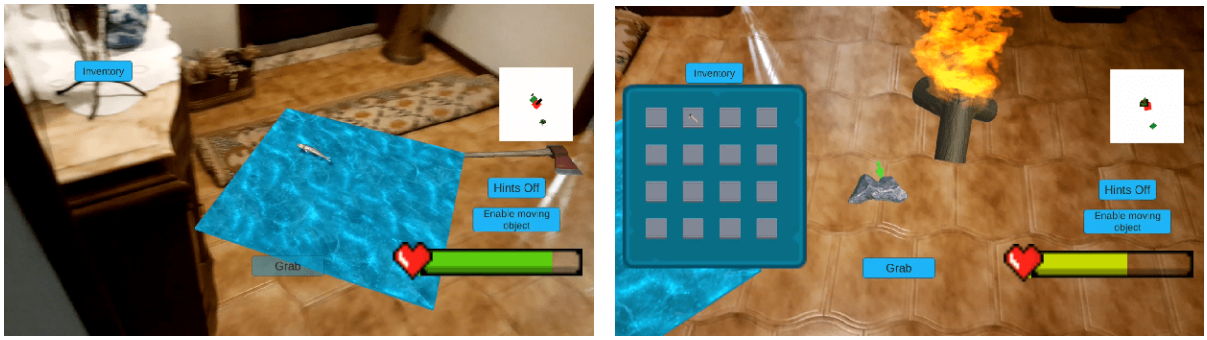
4.3.1 Configuration

The configuration module is the first application module and it is essential for its functioning. It is responsible for saving the AR environment after it is assembled. We will first introduce a preliminary version of the developed configuration tool used for other game scenario and then explore in detail its final version.

Preliminary Version

In the context of Realidade Virtual e Aumentada (RVA) course, a non-serious AR survival game was developed. The game configuration was achieved importing the required virtual objects into the Unity application editor and programming the game behavior. In figure 4.3, examples of images of this game are presented. Outside the gaming area, this configuration tool is also useful, almost every use case where the user wants to show or visualize something in a wide space with enough texture,

this application may be helpful and practical. Examples of these use cases are: A real estate agent who wants to show how an empty house/room may look like when furnished or an event promoter who may want to present an obstacle course in a school sports hall, without transporting and setting the real equipment. These and other tasks can be facilitated using the developed configuration tool and the application game window for the visualization of the created AR environment.



(a) Image example 1. Visualizing a river with a fish.

(b) Image example 2. Lighting up a fire.

Figure 4.3: Non-serious game developed in the RVA course using an earlier version of the configuration module.

Final Version

Four alternatives, supported by the ARCore SDK, are available to achieve the configuration goal and can be enabled changing the mode option, using the settings menu, in the smartphone application (figure 4.4) or using the configuration menu in the desktop PC application (figure 4.5). These approaches are (the way how each approach operates will be detailed below, in this section):

- Using ARCore cloud anchors technology and the placement of the virtual objects on the smartphone device (phone configuration - cloud anchors approach).
- Recurring to ARCore cloud anchors technology on the smartphone device to position the anchors and after all cloud anchors being configured, positioning the virtual objects on the desktop PC (desktop PC configuration - cloud anchors approach).
- Taking advantage of ARCore motion tracking technology robustness and placing the virtual objects using the smartphone (phone configuration - motion tracking approach).
- Setting the virtual objects position on the desktop PC, so it may be visualized using the ARCore motion tracking technology on the Android device. Requires the calibration of the starting position (desktop PC configuration - motion tracking approach).

The configuration in the desktop PC was achieved using the architectural plan as a representation of the physical scenario. In figure 4.7b we present the desktop PC configuration UI using the IRIS Lab architectural plan. The architectural plan image has to be scaled, in Unity, to match the real proportions of the physical scenario, so the virtual objects position and scale in the plan corresponds to the real world position and to the size in relation to the real physical space. The process for using the desktop PC configuration and the architectural plan will be detailed later in this section.

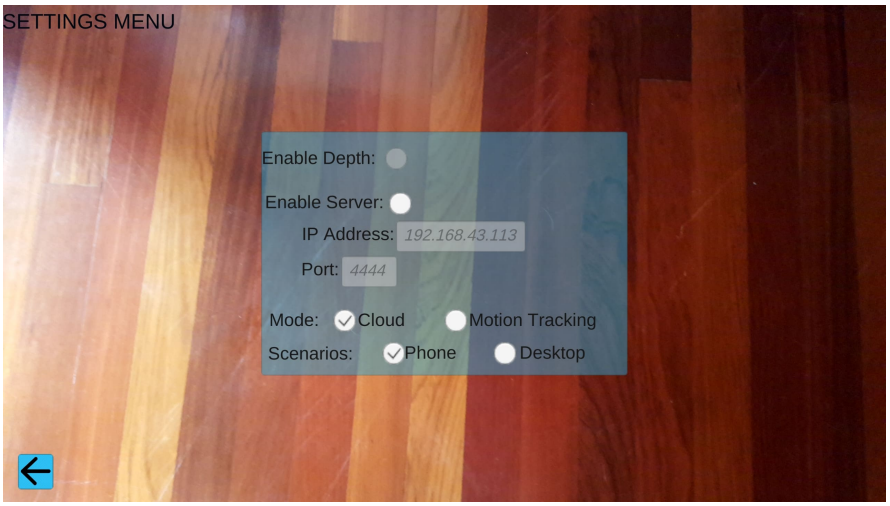


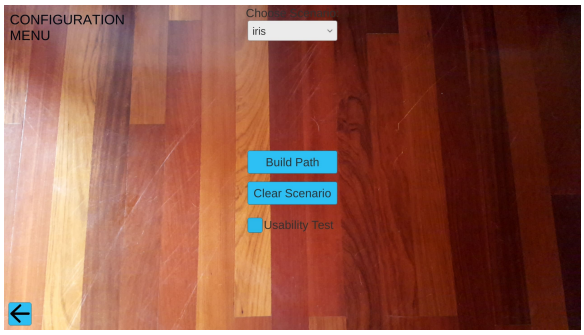
Figure 4.4: Smartphone application settings menu.

Figure 4.5: Desktop PC configuration menu. If the configuration mode is *Motion Tracking* instead of *Cloud* the *Choose File* drop-down list becomes an input field.

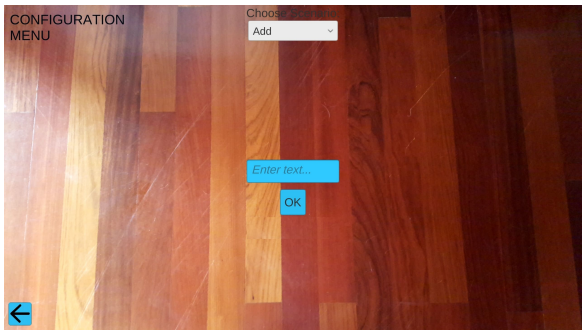
To employ the cloud anchors technology in the application and to benefit from its persistence, it was necessary to apply the Keyless authentication, allowing the host of a cloud anchor for one year, since the API Key authentication only provides a maximum hosting time of one day, as mentioned in the cloud anchors limitations, in section 3.1.3. In order to use the Keyless authentication we created and set an OAuth 2.0 client ID in the GCP console, using the credentials page. Then, using the Unity's application package name and fingerprinting the app's certificate, we were able to associate the GCP project, which contains the OAuth client, with the application.

We implemented the configuration module to be able to create and save several and distinct AR scenarios. Supposing that a user may want to create new AR scenarios in his application, in the same or in a different physical space, he can create them without overwriting the previously created scenarios. It is also possible to erase the content of a specific scenario or even to delete it from the scenario list. The UI for these actions is shown in figure 4.6.

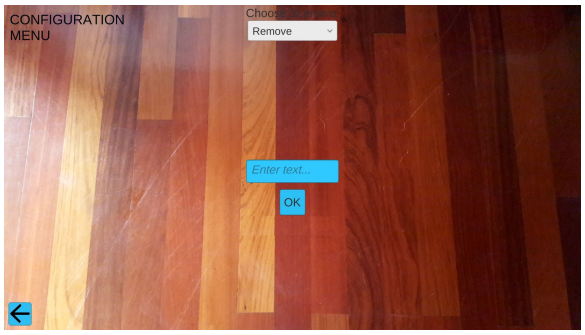
After opening a new or previously created AR scenario, it is possible to configure the AR environment. To accomplish this configuration, the intended virtual objects are placed in the real world (using the desktop PC approaches, an architectural plan is used) and interacting with these, it is possible to



(a) Smartphone configuration menu.



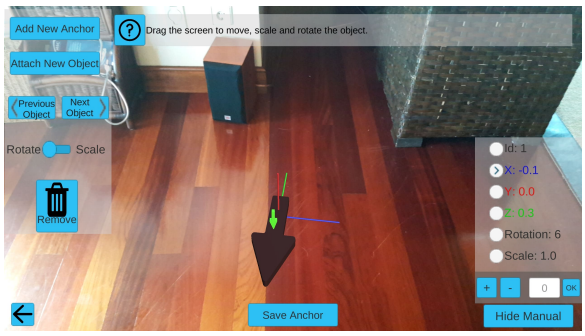
(b) Create a new scenario.



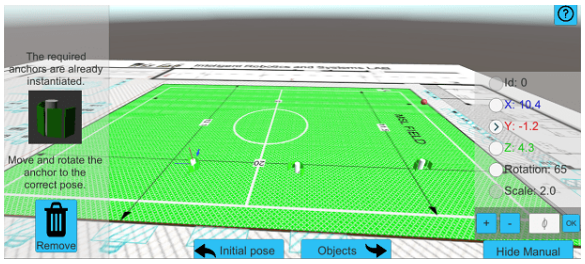
(c) Remove a created scenario.

Figure 4.6: UI for the smartphone configuration menu and UI for creating/removing scenarios.

apply geometric transformations to them. After selecting the intended object, it can be moved, rotated, scaled (applying translation, rotation and scaling transformations, respectively), changed in its format (modifying the virtual model) or removed from the scene. When using the smartphone device the transformations can be applied in an interactive way, dragging the screen. Regarding the desktop PC, interaction takes place through the mouse and the control key. Both, the smartphone and the desktop PC applications, have a side menu for manually insert the intended position, rotation and scale for the selected virtual object. The UIs for interacting with virtual objects are presented in figure 4.7.



(a) Phone configuration - Virtual object selected (directional arrow) and available to be moved, rotated and scaled.



(b) Desktop PC configuration - Virtual object selected (anchor object) and available to be moved, rotated and scaled.

Figure 4.7: Configuration UI for interacting with virtual objects in both applications (smartphone and desktop PC).

In order to save the configured AR scenario, all of its relevant data is stored in a JSON file on the application persistent data path (which points to */storage/emulated/0/Android/data/ <packagename>/file* in Android devices, where *<packagename>* corresponds to *com.Persistent* in the developed application). An example of this file is presented in listing 4.1.

```
{
   "anchorInfo":{
      "anchorX":0.0,
      "anchorY":0.0,
      "anchorZ":0.0,
      "anchorRotX":0.0,
      "anchorRotY":0.0,
      "anchorRotZ":0.0,
      "anchorScaleX":0.0,
      "anchorScaleY":0.0,
      "anchorScaleZ":0.0
   },
   "Name":"CloudAnchor0",
   "Scenario":"scenario_name",
   "Id":"anchor_id",
   "ListAnchorObjects":[
      {
         "Prefab_name":"virtual_object0_model_name",
         "X":0.0,
         "Y":0.0,
         "Z":0.0,
         "Rotation":0.0,
         "Scale_X":0.0,
         "Scale_Y":0.0,
         "Scale_Z":0.0,
         "BezierNumber":0
      },
      <...>
    ]
}
{
<...>
}
```

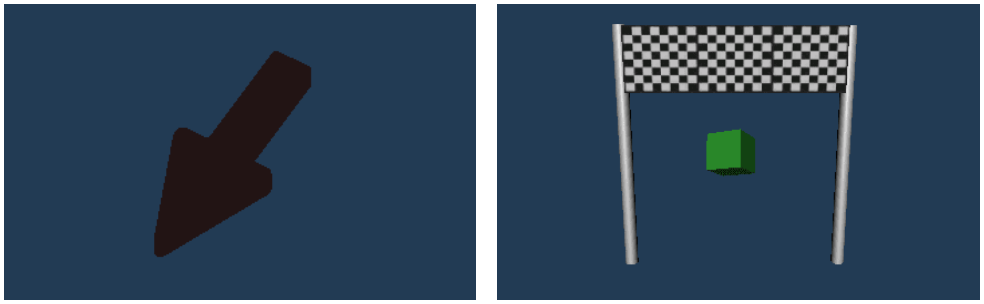
Listing 4.1: JSON file example for storing the configured AR environment.

This configuration tool is highly adaptable for other use cases, as seen in the preliminary version. Importing one or a set of virtual object models to our Unity editor application, completely distinct scenarios can be configured using the four configuration approaches developed. However, some limitations are acknowledged:

- The models must be imported in Unity editor and they cannot be imported in the final application at runtime.
- If the configuration is accomplished using the desktop PC, the models have to be imported to both, desktop PC and smartphone, applications.
- All the operating logic for the desired application must be programmed. Allowing only visualization without extra work.

Specifying the configuration module for the desired serious game, we integrated into the application the virtual models found in appendix section B.2, from the Unity Asset Store. Additionally, we also

developed a way to dynamically build a road, using the Unity's component *LineRenderer*. Adopting the logic of a Bézier curve of degree n and using the device's camera initial position (excluding the Y coordinate, which is swapped with the Y coordinate of the plane where the configuration is taking place), the passage point virtual object (represented with the model presented in figure 4.8b) and the directional arrows virtual object (represented with the model presented in figure 4.8a) as control points, representing the first endpoint, the remaining endpoints and the intermediate control points (if any), correspondingly.



(a) Model representing the directional arrow. (b) Model representing the passage point.

Figure 4.8: Virtual object models used to build the road.

At the end, k Bézier curves will be drawn, where k is the number of endpoints added (the number of passage points). Each control point has an associated integer identifier, (excluding the initial position, which is always the first one, the remaining control points ids, can be changed in the manual insertion menu), so that, between each endpoint and the next one, a Bézier curve with fifty points is drawn using the directional arrows, that have their ids comprising between the ids of these endpoints, as intermediate control points. An example of an AR road can be visualized in figure 4.9.

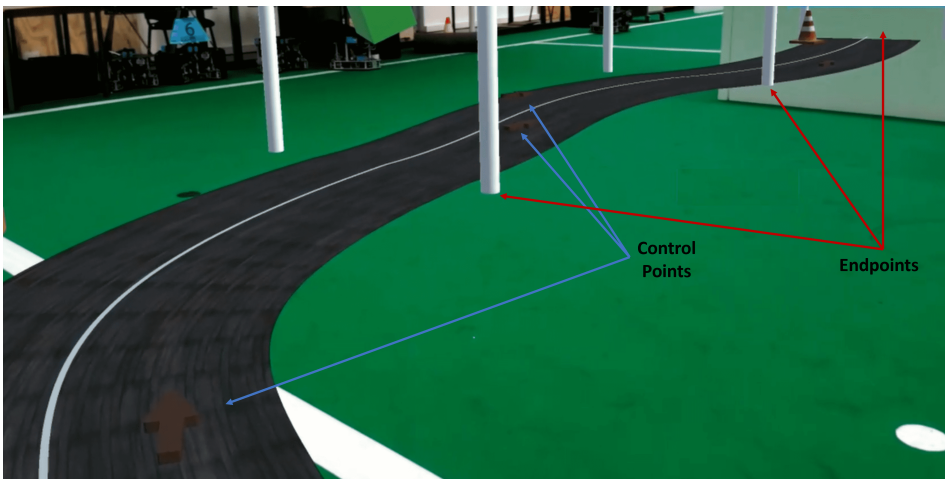


Figure 4.9: Example of a road. The directional arrows are control points and the passage points are endpoints.

Smartphone Configuration - Cloud Anchors Approach

For the smartphone device configuration, using the ARCore cloud anchors technology, it is necessary to place one or more anchors around the environment, near the intended virtual object positions and verify its feature points quality, as described in section 3.1.3. The UI and the user manual instruction to verify the anchor quality, for hosting purposes, is shown in figure 4.10.

(a) UI with an anchor placed.

(b) User manual to obtain the cloud anchor quality. *"Point the device to the anchor and move around it, until all the walls turn yellow and then green."*

Figure 4.10: Smartphone configuration UI and user manual instruction when a cloud anchor is placed.

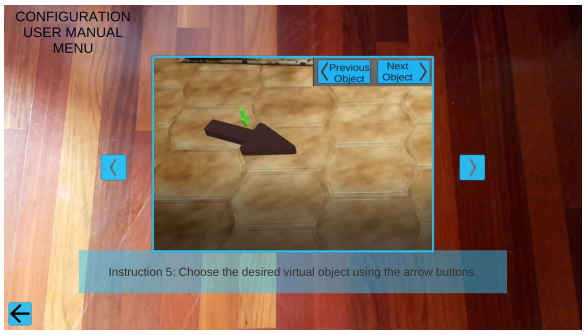
For each anchor, it is supposed to attach the desired virtual objects that stand close to it, so these virtual object poses are stored, relatively to the associated cloud anchor. Virtual objects transformations are applied, using the fingers, to drag and touch the device screen or using the manual insertion menu. The user manual instructions for applying transformations and change the virtual object are present in figure 4.11. The remaining configuration user manual is presented in appendix section D.2.

Smartphone Configuration - Motion Tracking Approach

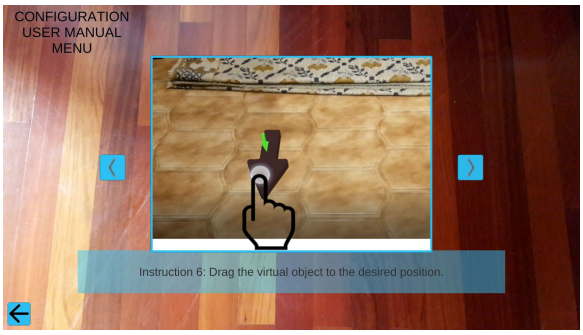
This approach is similar to the described above, but it presents some decisive differences. The first one is that, unlike the cloud anchors approach, the application does not use this technology, using only regular anchors during the configuration stage, so it does no longer need to verify the anchor quality (step 4 in the configuration user manual, figure 4.10b) and host it in the GCP. The latter anchors are only used during the configuration stage, so even if ARCore motion tracking technology suffers from minor miscalculations, the virtual objects can adjust their pose, resting in the same position and orientation, in relation to the physical space. During the game, these anchors are reduced to only one anchor as will be described in section 4.3.2. Another difference is the need to launch the application in a known physical position and rotation, so after completing the configuration, to play the game, the application must be started with the device situated in the same pose, for the AR environment to be aligned, having the virtual objects with the same coordinates as when configured.

Using this approach, it is important to store the anchors pose, so when placing the virtual objects, their poses can be computed based on the pose of the corresponding anchor.

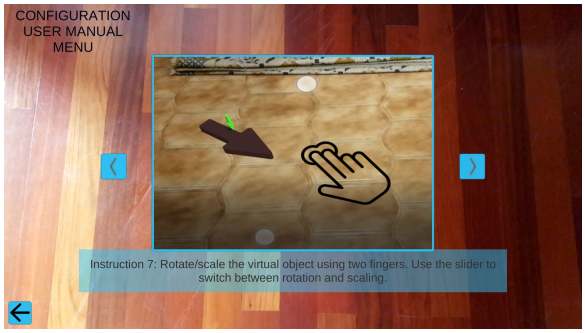
In case the ARCore motion tracking gets lost, providing the same device's camera pose over consecutive frames, such as when pointing the camera to a white wall while walking, a red warning message appears, informing the user that the part of the scene where the device is point at, does not contain enough texture for the proper functioning of the application.



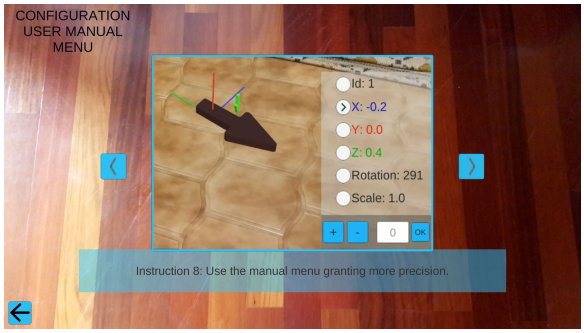
(a) Change the virtual object model. *"Choose the desired virtual object using the arrow buttons."*



(b) Move the selected virtual object. *"Drag the virtual object to the desired position."*



(c) Rotate/Scale a virtual object. *"Rotate/scale the virtual object using two fingers. Use the slider to switch between rotation and scaling."*



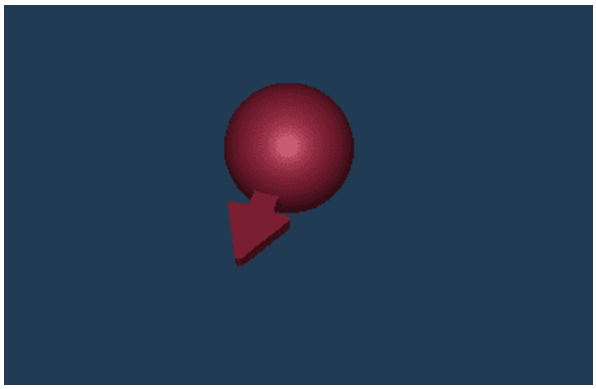
(d) Move, rotate and scale the moving object manually inserting the desired numeric values. *"Use the manual menu granting more precision."*

Figure 4.11: User manual instructions to interact with virtual objects.

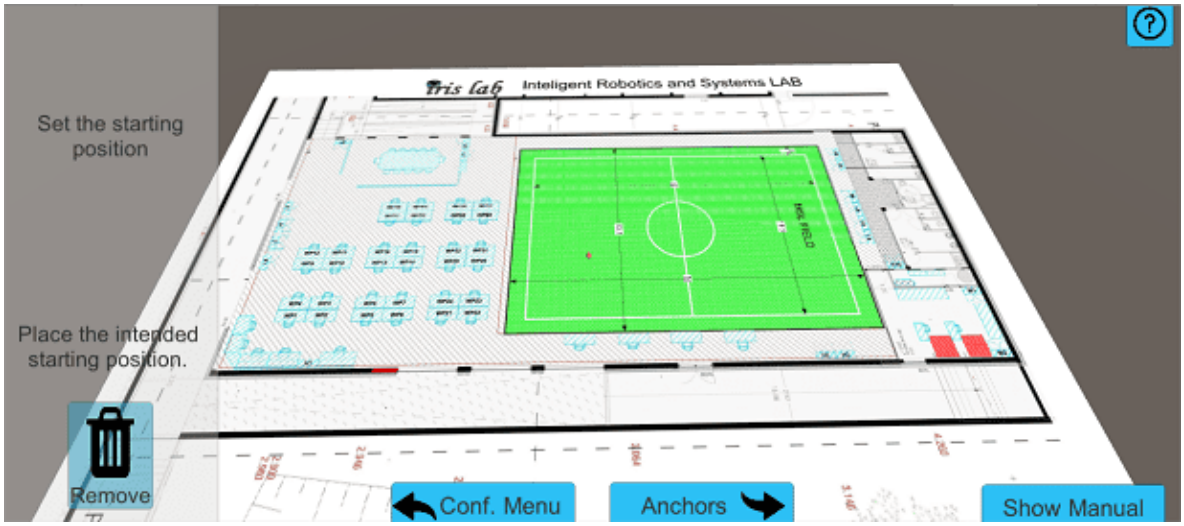
Desktop PC Configuration - Cloud Anchors Approach

The configuration can also be performed with the support of a desktop PC. For this approach to work, first it is essential to place and configure all the required cloud anchors, using the Android device, in the physical environment. Then, the generated file, containing the number of cloud anchors created and their corresponding poses, relatively to the device initial pose, is transferred to the desktop PC application. Having this file and the architectural plan of the physical scenario, objects representing anchors (*anchor objects*) can be positioned in the virtual plan, relatively to the physical scenario. It is possible to align the starting object (represented with model 4.12a) pose, in relation to the plan, with the Android device starting pose. This way, the *anchor objects* will be automatically placed on their correct poses on the virtual plan. The automatic placement is achieved computing the *anchor object* pose using its pose stored in the file, obtained from ARCore motion tracking technology, relatively to the starting object pose. It may be necessary to adjust the *anchor objects* pose, if any motion tracking error occurs. Figure 4.12b presents the desktop PC UI for the starting position alignment.

After all *anchor objects* being correctly positioned in the plan, the desired virtual objects can be placed on top of the plan, building the AR environment. The virtual objects will always be associated to the closest *anchor object*. The UI with every *anchor object* placed as well as calibrated and with an example of disposition of the virtual objects over the architectural plan is show in figure 4.13. The transformations on the various virtual objects are applied using the mouse and the keyboard. The Unity camera responsible for showing the environment can be zoomed, moved and rotated to add precision



(a) Model representing the smartphone starting position object.



(b) Desktop PC configuration UI for positioning the virtual object representing the smartphone starting position.

Figure 4.12: Alignment of the Android starting position, represented with a virtual object model, in relation to the virtual architectural plan.

---

and user control to the configuration. At the end, the virtual scenario can be saved, re-writing the file, with the list of virtual objects for each *anchor object* and transferred back to the Android application persistent data path to the *Output_Desktop_Configuration* folder. The desktop PC configuration user manual can be observed in appendix section D.3 and the completed configuration interface in appendix section C.2.1

Desktop PC Configuration - Motion Tracking Approach

Finally, the fourth approach, configuration using the desktop PC for the motion tracking technology, in contrast to the one for the cloud anchor technology, does not require the placement of the cloud anchors in the physical environment using the smartphone device. However, it requires to position the starting virtual object (figure 4.12a), which illustrates the smartphone starting pose, in relation to the architectural plan, so the configured environments, using this approach, can be used, aligning the smartphone starting position with the one represented by the starting virtual object. This application

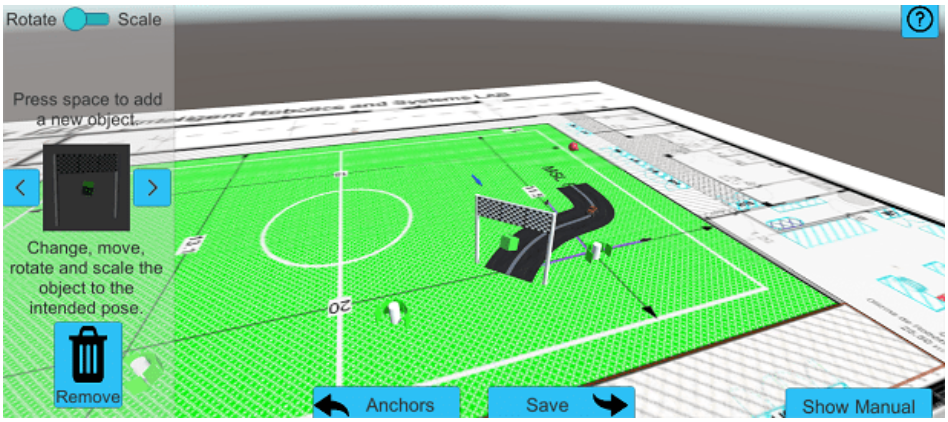


Figure 4.13: Desktop PC configuration UI while building the desired environment, setting the virtual objects.

does not require the placement of *anchor objects*, but to keep consistency over the output file, at least one *anchor object* has to be placed in a non-relevant position, so the JSON file can have the same format as the previous approaches.

4.3.2 Augmented Reality Serious Game

The second module is the developed ARSG to be played in the motorized wheelchair (described in section 2.5) with the folding smartphone arm support. Here, the two operating modes (ARCore cloud anchors and ARCore motion tracking) to build the AR environment and play the game were also implemented. For this module to work, a previous AR scenario needs to be configured and saved in the data JSON file, which has to be stored in the application persistent data path.

Adopting the ARCore cloud anchors technology, as the wheelchair travels through the physical environment, this technology searches for the locations where the cloud anchors with the ids, stored in the data file, have been hosted in the GCP ARCore Cloud Anchor API. When the smartphone points to the locations of the cloud anchors, these will be resolved and accessing the configuration data file, each of the virtual objects associated to the specific cloud anchor will be instantiated, translated, rotated and scaled in relation to the anchor, emerging in the game with the poses that have been configured.

For the operating mode using the ARCore motion tracking technology, the virtual objects will be instantiated when in sight of the smartphone's FoV. Their position and rotation is automatically set in their instantiation, as their pose can be computed at the beginning of the application, based on their associated anchors poses, but without creating the anchors. The anchors are not created since this would imply the instatiation of the anchors and their respective virtual objects too close to their corresponding places, emerging almost on top of the user and with little time for maneuvers. Instead, one anchor close to the starting position is created, so that every virtual object, when instatiated, is associated with this anchor. If the player returns to the starting position, this anchor will bring all virtual objects to their initial position, erasing eventual motion tracking errors. Furthermore, the AR immersion is greater as each pose of a virtual object is relative to the poses of the remaining virtual objects.

In figure 4.14 we present the gaming module interface, in figure 4.14a we can observe the static objects game and in figure 4.14b the moving object game.

For the game where the moving object is followed, the moving object trajectory has to be drawn.

(a) Playing a game with static objects.

(b) Playing a game with the moving object.

Figure 4.14: ARSG UI.

This is accomplished making it travel through the *LineRenderer* represented by the middle of the road. Therefore, its rotation is the angle of the vector calculated between the current line point and the next one and its position would be equivalent to each point present in the line (composed of fifty points, in this concrete case). But, in order to generate a constant motion speed, in each road section (between two endpoints), which can have different line lengths, the road section length is calculated (in meters) and rounded down for creating a "new line" between each point in the corresponding road section and the next point. This new line contains the calculated road section length as its number of points. This way, the moving object will move to the position of every point of each new line in the main line. Four possible speeds are available for the moving object, making it jump every 0, 1, 2 or 4 points in the new lines, moving 2cm, 4cm, 8cm or 16cm, respectively, each frame.

The game scoring, regarding the static objects game, is calculated based on three conditions:

- The time, subtracting a point into the player score for each second that passes.
- The additional features described in section 4.1, which subtracts from the score 50 or 100 points, for leaving the road or not complying with the remaining virtual objects, respectively.
- The 2D distance (calculated using the X and Z coordinates) from each point in the *LineRenderer* to the closest projection on the plane of the device's camera position in the traveled path to that point. Therefore, the number of 2D distances calculated is equal to the number of points that constitute the *LineRenderer*. Adding a score point for each decimeter in the computed 2D distance, to a maximum of 10 points (if the distance is smaller than 0.05 meters).

For the moving object game score, the first two conditions are kept, but the third condition is different. The 2D distance is from each point in the *LineRenderer* that the moving object passes to the device's camera position in the current frame. The perspective of the device's camera in the folding arm support only allows to visualize the moving object about 40 centimeters ahead. This was taken into consideration when calculating the score, subtracting the 40cm distance to the calculated ones when the moving object is in the device FoV.

The logic behind the additional features is the following:

- **Obstacles**: Triggers an visual explosion animation when a Unity's event capable of detecting collisions is called. A similar behavior happens picking the cubes on the passage points but triggering fireworks, instead of an explosion.
- **Leave the road**: When the current 2D distance (using X and Z coordinates) from the device's camera position to the closest point in the middle of the road exceeds half of the road width, the wheelchair is acknowledged as being out of road. An exception is activated when the device is near to an obstacle, allowing to leave the road without losing points on the score.
- **Stop sign**: The stop signs appear when in sight of the smartphone's FoV and are less than 2 meters away from the smartphone. Then, after 5 seconds, it is verified if the device is immobilized, not changing its position for a second and a half (a small interval in the immobilization is determined, contradicting minor motion tracking mistakes, in millimeters).
- **Spotlight**: Their activation is identical to the stop sign, but at a distance of 4 meters. After 3 seconds, it will be checked if the spotlight is still in the smartphone FoV.

At the end of the game, the user can turn around and verify the traveled path by the wheelchair while he was playing. For this, at the end, when the cube of the last passage point is caught, a red line is drawn, filling a new *LineRenderer* with every position that ARCore motion tracking technology kept track off, translating each point to the ground (where the road is drawn). Additionally, it is also possible to watch a replay of the game, as we will describe in the next section.

The application user manual for the gameplay, is found in appendix section D.4.

4.3.3 Game Performance Visualization

The last application module is responsible for displaying, in a desktop PC, the wheelchair driving during the games, so any difficulty or defect can be discovered and corrected more easily. This was achieved creating a client-server model, using a Transmission Control Protocol/Internet Protocol (TCP/IP) communication, implementing the client in the Android game application and conceiving a

new desktop PC application to act as a server. When executing the server application, it will associate the socket with a specific network interface and port number and starts listening for connections. On the other side, when starting the game on the Android device (the client), this will connect to the server. In order, to accomplish this connection, both devices have to be connected to the same network. In addition, the client must have access to the server's IP address and port and configure them in the application's settings menu (figure 4.4), also enabling the sending of messages over the network. Then, while the game is being played, the smartphone application continuously sends distinct messages to the server.

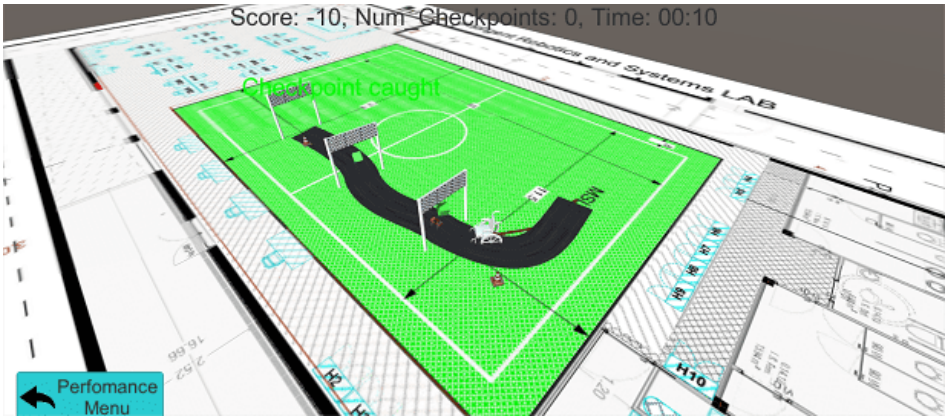
These messages have a one-way pattern (client to server), are formatted in JSON and can have multiple parameters, depending on the message type. The messages types and corresponding functions are the following:

- *REGISTRATION*: The first message, sent when the connection is acknowledged, has the purpose to create a new file for the corresponding user, using its username.
- *STATIC* or *MOVING*: Indicates the game type (game with static objects or game with the moving object).
- *CAMERA*: Has the device's camera pose of the current frame.
- *MOVING_OBJECT*: Has the moving object pose of the current frame.
- *LINE*: Holds the points of the line representing the road. These messages are received when a new road section is detected.
- **_OBJ*: Carries the instantiated virtual object pose. * can represent the directional arrow, the barrier, the cone, the spotlight, the stop and the passage point.
- *CHECKPOINT*: Signals that a cube on the passage point was caught, identifying the corresponding passage point.
- *LOOKWAY_SUCC* or *LOOKWAY_FAIL*: Indicates if the user succeeded or failed deflecting the spotlight virtual object.
- *STOP_SUCC* or *STOP_FAIL*: Indicates if the user succeeded or failed stopping the wheelchair.
- *DODGE_FAIL*: Designates a collision with an obstacle, identifying the corresponding obstacle.
- *OUT_OF_ROAD*: Indicates that the wheelchair left the road.
- *END*: The last message received, when the cube of the last passage point is caught, representing the end of the game.

Within each message, there are also fields for the current message timestamp, score and number of passage points achieved. Since the messages length can be long, due to the *LINE* message type, a message end code is sent at their end. A message illustration is shown in listing 4.2. As messages are sent, they are consequently received by the server, which is responsible for writing each complete message to the corresponding user's file. When the *END* message is received the connection is closed and the file is stored in Unity's desktop PC application. Thereupon, the performance can be displayed in a desktop PC, reading and processing the saved messages, showing what occurred during the game (the model representing the wheelchair can be observed in figure 4.15a, although this model is not

the TA IQ MWD wheelchair model, it makes the visualization more intelligible). Here, similarly to the desktop PC configuration, the architectural plan of the game scenario and the starting smartphone pose, relatively to the plan, can also be chosen. An example of a game performance visualization is shown in figure 4.15b. The user interface and the user manual, for this module, can be seen in appendix sections C.2.2 and D.5, respectively.

(a) Model representing the wheelchair. This model is only a representation for the motorized wheelchair, it does not correspond to its authentic model.



(b) Visualizing the game performance.

Figure 4.15: Game performance visualization using a wheelchair model for representing the player.

```
{
   "user_name":"username",
   "initial_time_s":"01/01/2021 00:00:00",
   "type":"message_type",
   "current_timestamp_s":"00:00:00.0",
   "obj":{
      "position_x":0.0,
      "position_y":0.0,
      "position_z":0.0,
      "rotation_x":0.0,
      "rotation_y":0.0,
      "rotation_z":0.0
   },
```

```
   "gameState":{
      "minutes":0.0,
      "seconds":0.0,
      "num_checkpoints":0,
      "score":0
   },
   "line":[
      {
         "x":0.0,
         "y":0.0,
         "z":0.0
      },
      <...>
   ],
   "messageEnd":"split_by_this_message_end"
}
```

Listing 4.2: Illustration of the JSON message for the game performance visualization, sent by the game application.

4.4 Unity Library

Besides the developed application, a library package was also created to be used in other Unity games. As referred in section 4.3.1, the configuration module, for other use cases, presents the limitation of only allowing the observation of virtual objects in an AR environment, requiring to transfer all the game components, materials and logic to our application and set them up all over again. For overcoming this problem, this library was built, since Unity already presents a scene configuration tool, it establishes a way to easily transform an already created game into an ARG using ARCore motion tracking technology.

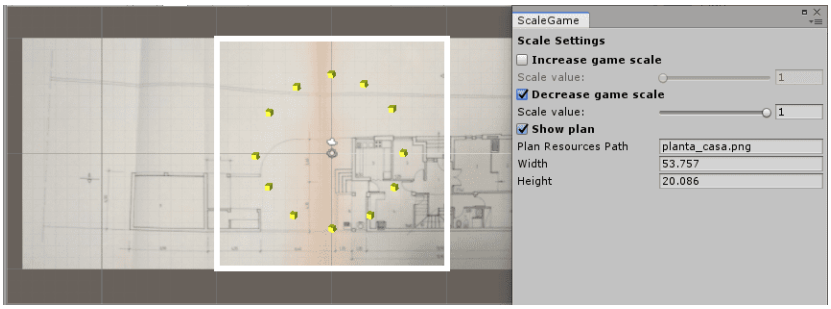
This is attained, disabling the scene camera and creating a new one with all the ARCore components associated with it, as well as the player's game object, making the Android device the player (or an observer if it has no active role in the game). All Unity build settings and player settings are also configured to support ARCore and *Build and Run* the application to the Android device. Now, this new camera object can be positioned, with respect to the rest of the scene (or the other way around, positioning the entire scene in relation to the camera object), so when the application is started on the Android device, the whole scene will show up in relation to the camera starting pose.

The Unity scene can measure unrealistic dimensions when presented in the real world, being able to reach hundreds of meters or kilometers, which would be hard or even impossible to travel, so a method to scale the scene was developed. The scaling process is automated and consist in creating a new empty object and associating every object in the scene to this one, as their children. Then, scaling the new created object will scale the entire scene as a whole and this object can now be erased, after disassociating every object from it. This is automatically done using the designed scaling menu, as presented in figure 4.16. It is possible to add the architectural plan of the physical scenario where it is intended to play the game, thus, making the scaling process more perceptible and exact.

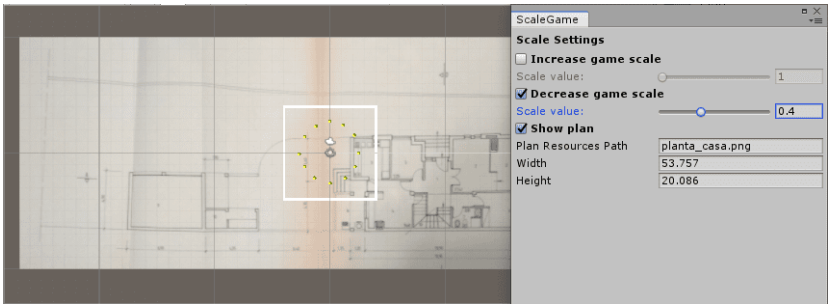
The library was tested in two free Unity games, *Roll-a-ball*[1] and *Tanks*[2]. The first one is a game that consist of a ball that rolls around the scene, picking floating cubes. The second one is a two-player tank shooting game. In the *Roll-a-ball* game, two experiments with different players were formed,

[1]https://learn.unity.com/project/roll-a-ball

[2]https://learn.unity.com/project/tanks-tutorial



(a) Scene with large proportions in relation to the physical scenario.



(b) Scene with appropriated proportions in relation to the physical scenario, after decreasing its scale.

Figure 4.16: Example of the scaling process. Decrease the proportions of the *Roll-a-ball* game scene.

---

one in which the Android device has the function of the ball, to pick the floating cubes and another, in which the smartphone carries a player consisting of a rigid body with an associated box collider and its role is to push the ball, in order to catch the cubes. At the *Tanks* game, an observer player was created, to watch the scene. Figure 4.17 presents images of these games in AR.

In appendix E, we provide the package content and the step-by-step instructions, to use this gaming library.

(a) *Roll-a-ball* game scene in AR, before pushing the ball.

(b) *Roll-a-ball* game scene in AR, after pushing the ball.

(c) Observing *Tanks* game scene in AR.

Figure 4.17: *Roll-a-ball* and *Tanks* games as ARGs, using ARCore SDK motion tracking technology.

Chapter 5

Usability Tests

In this chapter, we present the usability tests conducted to evaluate the developed application. We conceived two tests at different stages of the application, at a preliminary stage and at the end of its development. In the final usability tests, we divided the tests to evaluate the configuration module and the game module (along with the performance visualization module).

5.1 Preliminary Usability Tests

At an earlier stage of the application development, we performed usability tests to evaluate the AR application and its interface. The objective was to evaluate how users, without any experience in the application, behave manipulating it, their difficulties and unintended actions, in addition to verify the application potential and possible limitations.

In the following subsections, we will explain the state of the application at the moment these tests occurred, what the tests consisted of and the required tasks. Then, we present the results obtained in the tests and the changes to the application that derived from the results of these tests.

5.1.1 Application Stage and Test Setup

The preliminary tests were performed while the application was under development and it was still intended to be played by users without physical disabilities, i.e., by users without the need to use a wheelchair to play the games, playing them on foot.

The tests consisted in accomplishing five tasks inside a furnished residence with the AR environment already configured and prepared. The users did not had contact with each other during and before the respective experience, so that there was no influence between users and their first experience with the application was during the test. At the beginning of the test a shorter or longer explanation on the concept of AR, depending on the user's knowledge about it, was given, followed by a contextualization of the state of the application at the moment, its future application area and future goal. After the user understood its usefulness, he was asked to complete each task, sequentially, providing a few more information about the task if needed. Before starting each task, we encouraged the user to browse through the application searching for the menu to perform each task so the navigation through the application could also be evaluated. The requested tasks are:

- **Task 1: Play a game with static virtual objects, on the scenario "*House*", using the cloud services**. This task consisted in choosing the correct scenario and then, follow the direction of

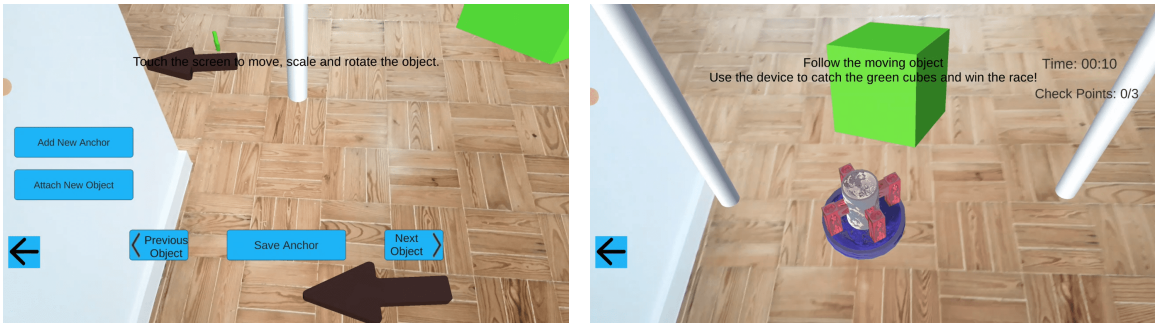
the virtual brown arrows and catch all the three green cubes that appear along the circuit in the shortest possible time.

- **Task 2: Play a game with the moving object, on the scenario "*House*", using the cloud services** (figure 5.1b). Similarly to the previous task, the goal is to catch the three green cubes disposed on the same circuit in the shortest time possible, however, instead of following the arrows direction, it is required to place a moving object on the ground (touching the smartphone screen) and follow it to the end.

- **Task 3: Create a new scenario and configure it, using the cloud services on the smartphone, to have one passage point and one directional arrow, both with a position, rotation and scale at the user's discretion** (figure 5.1a). First, a brief and non-technical explanation of what an anchor is and how to save it was summarily provided. Subsequently, after the user understood and internalized the concept, he had to create his scenario and name it, in the configuration menu. Proceeding with the new and clean scenario, he had to place an anchor on the ground and verify its quality (as described in subsection 3.1.3). Having all the walls green (figure 3.1b), the anchor object instantly disappeared and a virtual game object emerged in the same position and rotation. The user had to swap it to the intended virtual object (the passage point or the arrow) and translate, rotate and scale it to the desired transformation, using his fingers on the screen. After applying the three transformations on the virtual object he had to attach a new object close to the created anchor, choose the missing virtual object and apply the new desired transformations on the new virtual object. At any moment, he could reselect the previous virtual object to adjust it in accordance with the new one. In the end, the anchor had to be saved.

- **Task 4: Clean the created scenario in the previous task**. In the configuration menu, the user had to clear, from the scenario that he created earlier, the virtual objects and the associated anchors.

- **Task 5: Delete the created scenario from the scenarios list**.



(a) Configuration UI.

(b) Moving object game UI.

Figure 5.1: Example images from the application UI state at the preliminary usability tests.

An observer was accompanying the user while performing the tasks to observe and register his actions, difficulties and mistakes, as well as in the event of some kind of help or explanation was needed. After all tasks have been concluded, we asked the user to fill a questionnaire composed by ten items based on the System Usability Scale, divided into the two categories, play and configuration, and give his opinion about the application, the obstacles that he found and what could be improved

to make the application more intuitive and easier for future users. The questionnaire can be found in appendix F.1.

This evaluation had 4 participants with little to no experience and knowledge on AR interfaces. They included two females and two males aged between 20 to 60 years.

5.1.2 Results, Considerations and Changes

Overall, the users completed every task with success, however, their actions, difficulties and commentaries were not the same, what allowed to discover some flaws in the application that could be improved.

The main and more common problem was that the text hint was not being read or/and noticed by the user. To overcome this issue, we adjusted the application as a user suggested, creating a blinking button near the help text, inserting a semi-transparent gray panel behind it and increasing the text font size, to make it more outstanding. Another recommendation on the interface, was to simplify some help messages, texts and button texts, which were too technical for the application content and its final users, making them more user friendly. The settings menu was also created based on this, so that technical information could be found in the same region and in a more convenient place.

Regarding the game tasks, all users accomplished the game with static virtual objects with success and as expected, except for one user, who did not understand how the cube of the passage points were collected and required help. Which probably would not happen if he noticed the help text. Relatively to the game with the moving object, more difficulties were observed, the users looked for the moving object around the environment or started following the route without worrying about it, without realizing that they had to insert it into the floor. This led us to change the logic of positioning the moving object, trying to make this process automatic without user interaction. The first implemented approach was to place the moving object on a detected horizontal plane by ARCore, benefiting from its environment understanding. However, sometimes, the process of detecting the plane takes too long, so later on, it was changed to start at the beginning of the track.

With respect to the configuration, the interaction with virtual objects was considered intuitive and simple, but at the anchor quality check step, two users did not realize what kind of movement they should perform to accomplish this hosting stage. Since users, sometimes, struggled understanding what they should do, just reading the hint text, led to the creation of user manuals, where the instructional texts had visual and graphic support.

Every user agreed with the idea potential, recognizing that the game would be appropriate in the context of learning how to control a robotic wheelchair (question 1 - *"I think that i would like to use this application frequently for its usage scenarios."*). In addition, 75% of the users highly agreed and 25% had a neutral opinion that the game application was simple to use and well integrated (question 3 - *"I thought the application was easy to use."* and question 5 - *I found the various functions in this application were well integrated."*). As for the simplicity and integrity of the configuration application, 50% agreed and 50% had a neutral opinion. This suggested that playing the game was easier than performing the configuration.

5.2 Usability Tests for the Final Application

In order to evaluate the final version of the application, a second study was conducted. As in the preliminary test we intended to analyze the users difficulties, what should be improved and their opinion on the distinct tasks performed. The navigation through the application's menus was no longer

required, restricting the user to perform the tasks, but in addition, since the tasks were particularized, we collected data from the user performance during the task execution. These usability tests occurred in the IRIS robotics laboratory soccer field of University of Aveiro and were divided into configuration and game experiments. At the beginning of each user task, a brief introduction to the application and its technologies was provided. Then, the user performed the proposed tasks, while his actions were observed, and answered a questionnaire at the end. The questionnaire is presented in appendix F.2.

5.2.1 Configuration Usability Tests

In this section, we introduce the configuration usability tests, present the results gathered and discuss them. At the end, some considerations regarding the experiment outcomes will be pointed out.

Methodology

In the configuration usability tests, the main goal was to compare in which device the user had more precision, efficiency and preference while performing the configuration. Moreover, issues and difficulties in each individual task were also being noted. Therefore, two identical tasks were built:

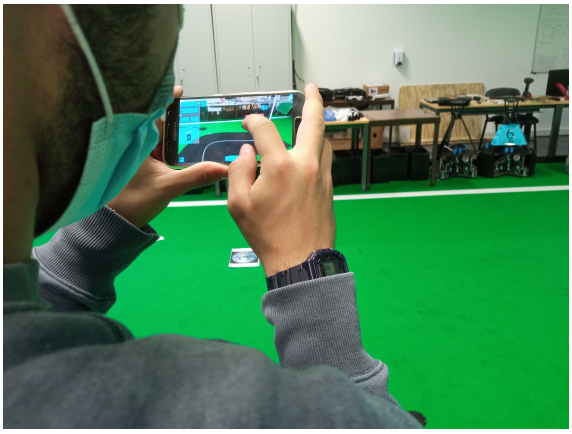
- **Configuration task using the smartphone device** (figure 5.2a): It was required to position four virtual objects in the place of four semi-transparent virtual objects that rest in a predetermined position, rotation and scale, relatively to a marker. The marker used is the one presented in appendix A.1 (figure A.1) and has the objective to fix the semi-transparent virtual objects to it, so their pose does not vary over the various tests. At the test start, the user had to point the Android device at the marker until the application recognized it. Next, the button responsible for attaching new objects becomes interactable. When the user places a new object on the floor, the semi-transparent object appears in relation to the marker, the user had to look for it and when found, the user had to change his placed object model and apply the necessary transformation to overlap the corresponding semi-transparent one. When the user considered that his virtual object was well placed, he could move on to the next one, attaching a new virtual object to the ground. The sequence of semi-transparent objects is the following:

  1. A direction arrow (model 4.8a) requiring translation.
  2. A directional arrow (model 4.8a) requiring translation, rotation and scaling.
  3. A passage point (model 4.8b) requiring translation, rotation and model switching.
  4. A barrier obstacle (model B.2c) requiring translation, rotation, scaling and model switching.

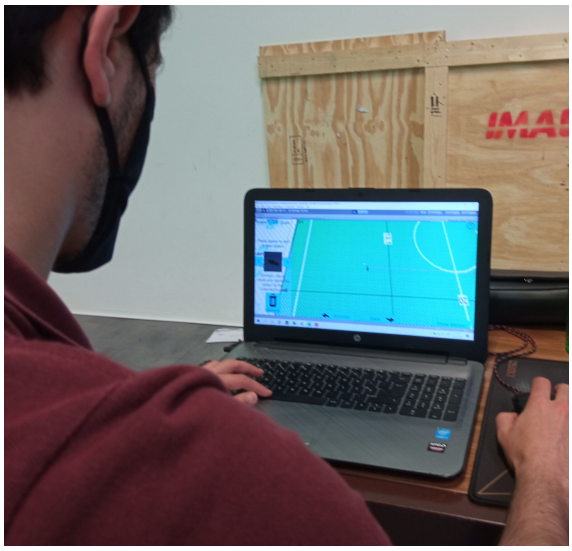
  The placement of anchors or interaction with them was not requested.

- **Configuration task using the desktop PC** (figure 5.2b): Similarly to the previous task, the users needed to position their virtual objects overlapping the semi-transparent ones, following the same sequence. The starting object (represented by 4.12a model) and the anchors are automatically placed by the application, so the semi-transparent virtual objects would appear on the architectural plan, in the same place relatively to the previously described task.

Before the task start, the user manual for the configuration method was presented to the user (user manuals in appendix D.2 and appendix D.3 for the smartphone task and for the desktop PC task, correspondingly). Afterwards, the user can experiment the configuration tool and its interface freely for a few minutes. Then, the usability test was explained and performed.



(a) User fulfilling the configuration task using the smartphone device.



(b) User fulfilling the configuration task using the desktop PC.

Figure 5.2: Users performing the configuration usability tests.

This usability study involved 10 participants, ranging in age from 19 to 34, studying or graduated in distinct areas. 60% of them already had contact with AR applications. The experiment was organized so that, half of the participants performed the smartphone task first, proceeding to the desktop task afterwards and the other half performed the opposite.

During the tests, for each user in each virtual object, the following data was collected: time to position the object, distance from the user object to the semi-transparent object position, the difference between the orientation and scale of the user object and the corresponding goal.

Results

The average results of the collected data from the ten participants for each virtual object in both devices, is found in table 5.1. These results are also presented with boxplots presented in figures 5.3a, 5.3b, 5.4a and 5.4b for the distance error, orientation error, scale error and time for the object placement, correspondingly.

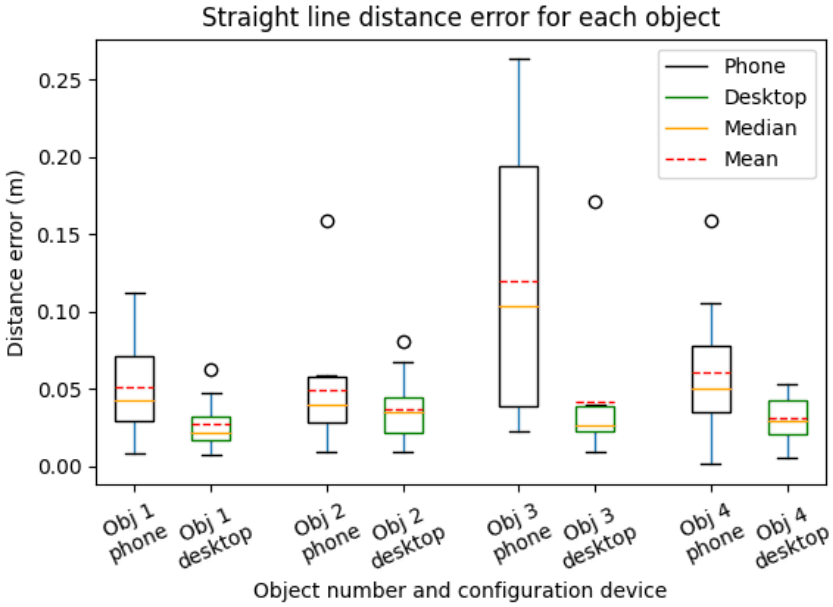
Regarding the distance error, between the user objects and the semi-transparent objects, figure 5.3a, the desktop PC method had better results in every virtual object, but the configuration method using the smartphone did not present bad results, only obtaining an average error greater than 1 decimeter on the third virtual object. Relatively to the orientation error, figure 5.3b, the results do not differ much in the first two objects. At the third object, the desktop PC task has slightly better orientation precision and at object number four, the smartphone configuration, in general, achieved smaller errors. The scale results, shown in figure 5.4a, also present similar results in both tasks. It is noteworthy that, performing the first task, some people scaled the third virtual object unnecessarily. Regarding the time to place the virtual objects, it is possible to observe, in figure 5.4b, that at the beginning of the tasks, the first object was positioned more efficiently using the smartphone, at the second one, most users were faster using the desktop PC, but some users were still considerable slower, finally, the last two objects, demonstrate the opposite, being placed significantly quicker using the desktop PC method. From this, we can deduce, that with training, most users would be more confident and efficient using

the desktop PC application. It was also possible to verify that with the desktop PC, users can be more precise, specially with larger virtual objects, as in the case of the third object.

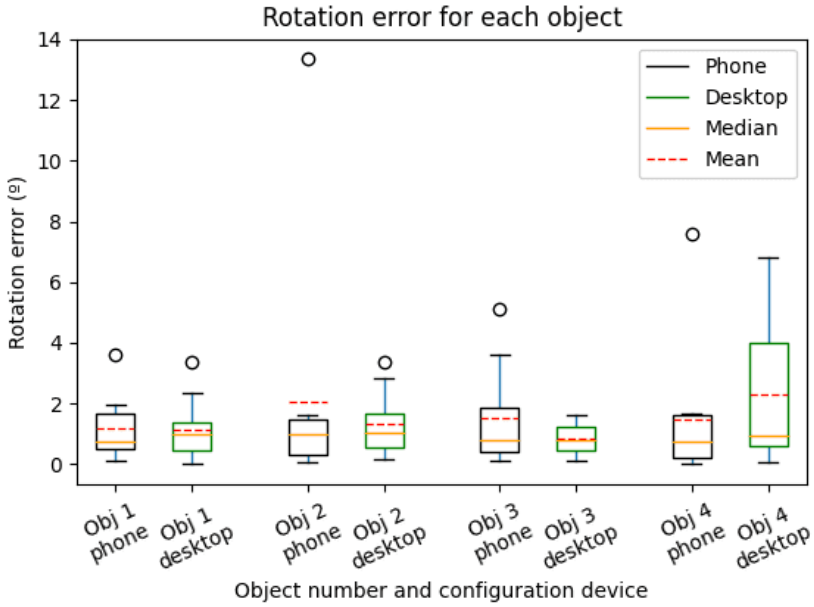
	Object 1		**Object 2**		**Object 3**		**Object 4**	
	Phone	**Desktop**	**Phone**	**Desktop**	**Phone**	**Desktop**	**Phone**	**Desktop**
Average time (mm:ss.ms)	00:54.38	01:12.49	00:46.22	00:55.08	01:38.86	00:47.37	01:10.89	01:00.87
Average X error (m)	0.02	0.02	0.02	0.03	0.08	0.03	0.03	0.02
Average Z error (m)	0.04	0.01	0.04	0.02	0.06	0.02	0.05	0.02
Average straight line distance error (m)	0.05	0.03	0.05	0.04	0.12	0.04	0.06	0.03
Average rotation error(º)	1.16	1.13	2.05	1.31	1.52	0.86	1.45	2.31
Average scale error (m)	0.00	0.00	0.11	0.12	0.01	0.00	0.03	0.06

Table 5.1: Average time and errors, from the ten participants, during the configuration in the final usability tests, for each virtual object positioned by the user in relation to the corresponding semi-transparent virtual object, on both devices.

In addition to the collected data on the user's accuracy and time to execute the tasks, we also asked, in the form of a questionnaire, the user's opinion about the configuration tools and their preference. In both methods, they considered that it was easier to place the virtual objects in the correct position and scale than in the correct orientation (questions 8, 9, 20 and 21 from the questionnaire). They agreed that, in both methods, their performance would improve with training (questions 13 and 26 from the questionnaire). Only one user, in both methods, considered that he would need the help of a technical person to use the application (questions 14 and 27 from the questionnaire). The remaining participants agreed that studying the documentation, would be enough to accomplish the tasks. Regarding the satisfaction of each method (questions 17 and 30 from the questionnaire), in general, the users preferred to use the smartphone, 30% were completely satisfied and 70% were satisfied with the desktop PC configuration tool, and for the smartphone tool, 60% were completely satisfied, 30% were satisfied and 10% were unsatisfied. This is justified by their observations, they found the smartphone tool more interesting and stimulating. Regarding the intuitiveness (question 55 from the questionnaire), 40% preferred the desktop PC tool over the smartphone, 40% the contrary and the remaining 20% showed no preference. Only two users used the manual insertion mode (question 58 and 59 from the questionnaire), considering that could bring advantages in some actions, in relation to the visual interaction mode.

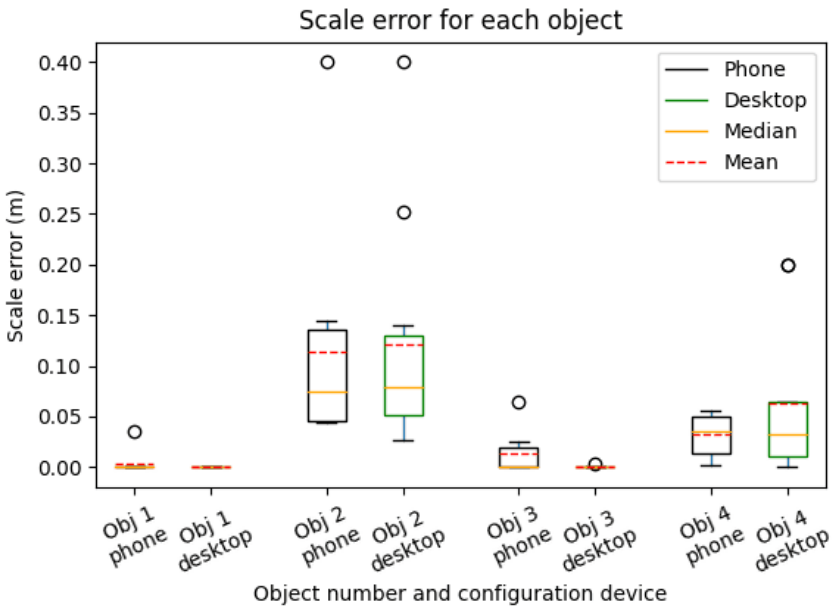


(a) Average straight line distance error.

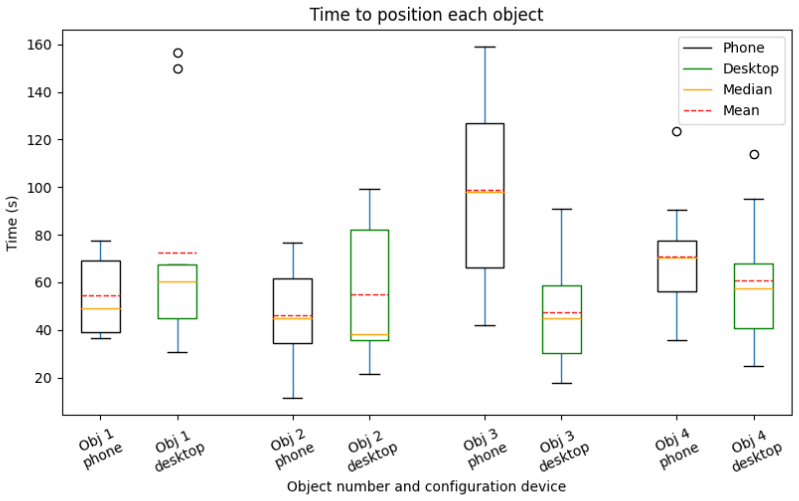


(b) Average rotation error.

Figure 5.3: Boxplots for the results of the ten participants, during the configuration in the final usability tests, for each virtual object positioned by the user in relation to the corresponding semi-transparent virtual object, on both devices.



(a) Average scale error.



(b) Average placement time.

Figure 5.4: Boxplots for the results of the ten participants, during the configuration in the final usability tests, for each virtual object positioned by the user in relation to the corresponding semi-transparent virtual object, on both devices.

Final Considerations

These usability tests allowed to analyze which methods would be better in which use case. The desktop PC method presented superior performance results, being able to bring more precision and efficiency for the user. However, the smartphone, being integrated into an AR environment, can be more engaging and entertaining, as seen by users opinion.

Furthermore, based on user behavior and feedback, we can infer that most people would easily be able to use both configuration tools, fulfilling its objective, but some features could still be improved. We observed that most users executed their second task more easily and confidently than the first one. We also noticed that in the desktop PC configuration, some people had difficulties finding the slider to change between rotation and scaling transformations. On the other hand, this slider on the smartphone interface was easily found, but some users struggled using it because it was too small for thicker fingers. Regarding the desktop PC method, one user suggested to adjust the camera movement depending on its zoom, what we considered to be a relevant improvement. With respect to the smartphone method, a few users had some problems to hold the device with their hands while applying the required transformations. Others thought that a larger screen would improve their performance.

5.2.2 Game Usability Tests

This section is structured as the previous section.

Methodology

The game usability tests had two main objectives. Regarding usability, the objective is to discover and evaluate if the ARSG developed is intuitive, if it supports the interaction with the robotic wheelchair and its control, as well as, to evaluate the game performance visualization, verifying if visualizing the game (that was previously played by the user) and the wheelchair trajectory in the desktop PC, can help the user in finding some failures while he was driving and how he could improve the wheelchair control. The second objective for these tests is to compare the ARCore motion tracking technology with the cloud anchors technology while playing the ARSG. To conceive this comparison and the usability evaluation, we configured the AR environment to contain three cloud anchors with a passage point associated in each cloud anchor. Thus, it is possible to compare the behavior of the virtual objects associated with cloud anchors in relation to their behavior using the motion tracking technology. In addition, the first road section had a barrier obstacle, the second one had a stop sign and a spotlight and the last road section with one cone obstacle close to the passage point. The configuration was performed placing the cloud anchors with the Android device and the virtual objects using the desktop PC. The AR race scenario configuration was implemented for using both, motion tracking and cloud anchors, technologies in the game, calibrating the Android initial position. Now, having the AR scenario built, the following tasks were applied:

- Task 1: Play one game with static objects, on the motorized wheelchair, using ARCore cloud anchors technology.
- Task 2: Play one game with the moving object, on the motorized wheelchair, using ARCore cloud anchors technology.
- Task 3: Play one game with static objects, on the motorized wheelchair, using ARCore motion tracking technology.

- Task 4: Play one game with the moving object objects, on the motorized wheelchair, using ARCore motion tracking technology.

These usability tests involved 20 participants, between 19 and 34 years old, with distinctive professional areas. Only 20% of this group already had contact with applications of navigation in indoor spaces and 30% had no experience with AR before. Half of these participants also participated in the configuration usability tests. This group of participants was divided into two sub-groups, having 10 participants in each sub-group. The first group of 10 participants performed the tasks involving ARCore cloud anchors technology (tasks 1 and 2) and the second group carried out the tasks concerning ARCore motion tracking technology (tasks 3 and 4). As in the configuration usability tests, the order for performing the tasks was alternated.

After explaining the usability tests to the user, he was able to experiment the wheelchair and its control, driving the wheelchair across the robotic soccer field and changing its speed to find which one would make him feel more comfortable during the task execution. Then, the ARSG user manual (appendix section D.4) was presented and the tasks were performed. In figure 5.5 its presented a test participant controlling the robotic wheelchair during the task execution. At the end, before filling the questionnaire, the game performance while completing the tasks was displayed to the user.

Figure 5.5: User performing the game usability test.

For evaluating ARCore technologies, the corresponding technology behavior while users performed each task, was categorized into one of four classifications. These classifications are:

- **Successful game run**: Every virtual object and the road appeared to be correctly placed, any errors were minimal or not visually noticeable.

- **Game run with ARCore auto correction**: The virtual objects and the road were misplaced, but ARCore corrected their positions, allowing to end the run.

- **Semi-successful game run**: The virtual objects and the road were not in the right place, however, it was clear which was the path and no real obstacles prevented from finishing the circuit.

- **Unsuccessful game run**: It was necessary to restart the game. It was impossible to finish the game because the virtual objects and the road were completely misaligned in relation to the configured AR environment.

User Results

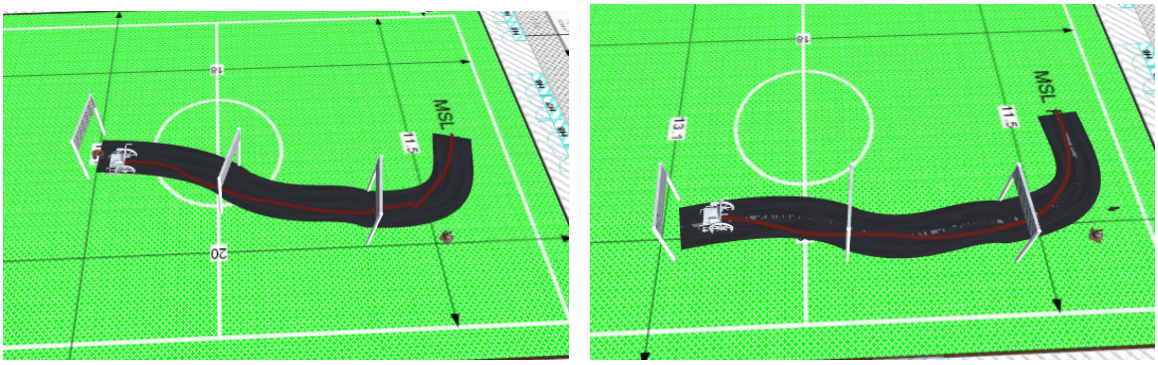
All users were able to accomplish the proposed tasks, passing by the three passage points and catching the green cubes, in each task (excluding unsuccessful game runs caused by ARCore flaws). Nevertheless, not everyone behaved in the same way, there were users who failed in completing some of the game additional features (stop, deflect the spotlight, dodge the obstacles, do not leave the road).

Table 5.2 summarizes the users results for each game version (game with static objects and game with the moving object). Users completed tasks 1 or 3 (static objects game) faster than tasks 2 or 4 (moving object game), due to the choice of a slow speed for the moving object (velocity 2) and as the static objects game allows the user to choose its wheelchair's driving pace, users could finish the corresponding tasks faster. The average distance to the goal (the white line in the static objects game and the car in the moving object game) was smaller in the static objects game. Relatively to the additional features, there were fewer mistakes in stopping at the stop sign and not going off the road, but marginally more collisions while playing the moving object game. In figure 5.6 we can observe user 6 trajectories while performing his tasks.

	Average Time (mm:ss.ms)	**Average nº of virtual object collisions**	**Average nº of successful stops**	**Average nº of unsucessful stops**	**Average nº of spotlights successfully deflected**	**Average nº of spotlights unsucessfully deflected**	**Average nº of exits from the road**	**Average distance to goal (m)**
Static Objects Game	00:30.9	0.25	0.40	0.35	0.35	0.05	1.00	0.33
Moving Object Game	00:35.6	0.30	0.75	0.05	0.25	0.05	0.90	0.40

Table 5.2: Average performance in the game, of the twenty participants, during the game in the final usability tests, following both the static objects path and the moving object.

Regarding the questionnaire results, the users preference, opinion and remarks brought interesting conclusions. 90% of the users found both tasks easy to accomplish (question 34 and 45 from the questionnaire), 5% had more ease in the task with static objects and 5% had a neutral opinion. They also agreed that the ARSG is intuitive (questions 33 and 44 from the questionnaire), 70% highly agree and 30% agree regarding the game with static objects as well as 75% highly agree and 25% agree with respect to the game with the moving object. Most users, who had unsuccessful runs, considered annoying the idea to restart the game. They agreed that their gaming performance would improve with training time in both game versions (questions 37 and 48 from the questionnaire). With respect to the static objects game for helping the wheelchair control (question 39 from the questionnaire), 75% completely agreed that this game version helped in interacting with the wheelchair, 20% agreed and 5% had a neutral opinion. Identical results were obtained in the moving object game to support the wheelchair interaction (question 50 from the questionnaire), where 75% completely agreed that this game version can help with this interaction and 25% agreed. The performance visualization module had acceptable results. For the static objects game (question 40 from the questionnaire) 45% highly



(a) User 6 performance visualization in the game with the moving object using ARCore cloud anchors technology.

(b) User 6 performance visualization in the game with static objects using ARCore cloud anchors technology.

Figure 5.6: Example of user trajectories (red line) drawn by the performance visualization module.

agreed that visualizing their game performance helped in improving the wheelchair control, 35% agreed, 5% highly disagreed and the remaining 15% had a neutral opinion. For the moving object game (question 51 from the questionnaire), 60% highly agreed, 25% agreed, 5% highly disagreed and 10% had a neutral opinion. Comparing both game versions, the moving object game was found as more useful to perceive the game objective (question 60 from the questionnaire), preferred by 25%, where 60% had a neutral opinion. Both games were equally appreciated (question 61 from the questionnaire). The moving object game was found as more useful for learning to control the power wheelchair (question 62 from the questionnaire), 35% preferred the moving object game over the static object game, where 50% of the group had a neutral opinion. The preference over this game version is due to some users learning that the wheelchair joystick had sensitivity (and how to use it), giving more or less speed to the wheelchair depending on the force applied to it. As the moving object had a slow speed, some users tried to adapt the wheelchair speed to the moving object speed, learning this joystick sensitivity.

ARCore Results

As previously referred, the second objective of this evaluation was to compare both game operating modes while users performed the usability tests. The gathered results are presented in table 5.3. It is observable that the number of runs using the motion tracking technology (26 runs) is larger than the number of runs using ARCore cloud anchors technology (21 runs) as a result of the unsuccessful runs, which were repeated from start. The cloud anchors approach succeeded 61.9% of its runs, therefore having more successful runs than the motion tracking approach, which only succeeded half of its runs (50%). The motion tracking operating mode does not have runs with auto corrections since this behavior only happens using ARCore anchors technology, when the anchor pose is adjusted, adjusting every attached virtual object with it. Using the cloud anchors operating mode 19% of the runs were auto corrected by ARCore. 26.9% of the runs using motion tracking operating mode and 14.3% using cloud anchors technology were semi-successful, allowing users to finish the track, which is much less disturbing for the users than having unsuccessful runs.

	Nº of runs	Nº of successful runs (%)	Nº of runs with ARCore auto corrections(%)	Nº of semi-successful runs (%)	Nº of unsuccessful runs (%)
Cloud anchors	21	61.9	19.0	14.3	4.8
Motion tracking	26	50.0	0.0	26.9	23.1

Table 5.3: Comparison between the two operating modes in the game (ARCore cloud anchors technology and ARCore motion tracking technology). Results from the twenty participants (10 in each operating mode) obtained during the game in the final usability tests.

Final Considerations

The game usability evaluation enabled us to distinguish which game version users preferred the most and why. Most users elected the moving object game as more intuitive and useful for learning. However, the static objects game brings more freedom to the user, which makes it more entertaining for some users and allowed some of them to find a deeper understanding of the spatial notion while using the wheelchair.

In the questionnaires or during the test, some users also pointed out some difficulties and suggested future improvements:

- As a difficulty, the device camera's perspective makes the road curves harder to achieve having the wheelchair aligned with the middle of the road.
- As an improvement, the implementation of the performance visualization in real time, displaying the game actions in a small window at the corner of the smartphone screen, which can facilitate the discovery of problems and mistakes in driving the wheelchair.

It was also verified that the ARSG, despite having some possible improvements, already showed its potential and it is possible to infer that its purpose was fulfilled.

Finally, these tests also allowed to compare the two operating modes, in which, the ARCore cloud anchors operating mode stood out as the most precise and robust. When choosing which of the operating modes will be used, it is necessary to carefully analyze what is intended, in order to verify whether ARCore cloud anchors technology robustness overcomes its imposed limitations.

Chapter 6

Conclusion

The main objective of this work is to explore the application of AR to facilitate the HRI between motor disabled people and robotic wheelchairs. To assist this, AR was combined with serious games, taking advantage of serious games educational capabilities and their entertainment potential that can reinforce learning, together with the visual aid and immersion that AR can bring.

After investigating and exploring several AR technologies, we chose to use the Google ARCore SDK, for its potential markerless tracking capability. Studying and testing the ARCore SDK for Android smartphones on the game engine Unity, we came across with a robust cloud anchors technology, that despite some limitations, also offers a promising motion tracking technology, without the need of anchors. An evaluation was conducted to verify how accurate and robust the ARCore motion tracking technology is, showing its potential for the AR solution that we developed. Besides this, the evaluation also originated some good practices for using ARCore in the serious games with the motorized wheelchair. Therefore, the ARSG was developed considering two operating modes, using the cloud anchors or the motion tracking technologies. The development of the ARSG consisted in three modules: configuration, serious game and performance visualization. The configuration module aims to be an intuitive tool for configuring and saving the AR game environment, but it was also intended to create this tool as highly flexible for other use cases. This adaptability was achieved, but some limitations were found preventing to accomplish the configuration of a distinct use case scenario, with distinct virtual objects, at application runtime, requiring to use Unity editor. Besides this, the configuration tool was built for using the two operating modes (ARCore cloud anchors or ARCore motion tracking) during the serious game in Android devices, as well as in desktop PCs. On the other hand, the serious game module uses the configured AR environments to present a serious racing game to be played in a robotic wheelchair. The operating mode used for configuring the AR scenario should be chosen to play the game, however if the configuration was performed using the cloud anchors technology, both operating modes may be used to play the serious game. The serious game is designed for having a set of virtual objects with certain behaviors, which can be arranged to support a novice user in learning and training to control motorized wheelchairs. Finally, the performance visualization module, communicates with the serious game module to receive and save the distinct actions during the game. Hence, these actions can be processed and displayed in a desktop PC application to the wheelchair user or its health carer, to support and find some issues in the interaction between the user and the robotic wheelchair.

To evaluate the developed application we conducted usability tests. The preliminary usability test, carried out at an earlier application stage, had the objective to improve the navigation through the application and its interface, as well as to verify the user behavior when configuring an AR

scenario using the cloud anchors approach and the gameplay of the serious game on foot. These tests allowed some important improvements in the application, such as the placement of the moving object. Regarding the final usability tests, it was intended to evaluate if the developed application fulfilled its purpose, verify the user preference and performance over the created methods, as well as compare the employed operating modes, using different ARCore technologies. These tests showed that the configuration tool can be easily adopted by users with little AR experience. They also revealed that the developed ARSG supports the learning of the robotic wheelchair control, accomplishing the objective of this work and reinforcing the idea already presented in the literature, that AR technologies are a useful tool for supporting HRI and people with motor disabilities. Besides this, it also demonstrated that the operating mode using ARCore cloud anchors technology is more robust than the one using ARCore motion tracking technology.

In addition to the developed application, we also created a library package for Unity games. As Unity already offers a platform for creating and configuring games, we developed a way to easily and automatically adapt an already created game into an ARG using ARCore motion tracking technology in an Android device. Thus, allowing to play or observe various games in an immersive environment, including other entertainment or serious games for rehabilitation.

Future Work

There is some additional work that can be added in the future in order to improve what was achieved in this dissertation:

- ARCore cloud anchors technology has some limitations. Depending on the type of device's movement and of the texture of where the anchor is fixed, the anchor appeared (through empirical tests) to have distinct behaviors, resting in more or less precise positions and even suffering from small displacement while looking at it. Hence, this might result in inconvenient movements of the virtual objects attached to the anchors spread around the AR environment. It would be interesting to further evaluate ARCore cloud anchor resolving technology, in order to study its accuracy, as well as whether and how it is possible to enhance the anchor recognition and precision.
- Implement a way to serialize Unity *GameObjects* and attached scripts, allowing to import virtual object models to the application in runtime, removing the limitation to make this operation in Unity editor and improving the adaptability of the configuration tool.
- Improve the developed library to support and take advantage of ARCore cloud anchors technology as an option to motion tracking technology.
- Implement in the developed library a way to allow the scaling and the player alignment of the transformed games in runtime. The changes in Unity editor would be further minimized, a clearer vision for the ideal game scale could be achieved, as well as permitting to start the application in any pose and afterwards placing the ARG scenario in relation to a plane, for example, the floor.
- Create a synchronization mechanism in the developed library, which would allow to play the game on a desktop PC, while another user would observe the game's actions in the AR environment.

Bibliography

Billinghurst, M., Clark, A. and Lee, G. (2014), 'A survey of augmented reality', *Foundations and Trends in Human-Computer Interaction* **8**(2-3), 73–272.
URL: *http://dx.doi.org/10.1561/1100000049*

Carmigniani, J. and Furht, B. (2011), Augmented Reality: An Overview, *in* B. Furht, ed., 'Handbook of Augmented Reality', Springer, New York, NY.
URL: *http://dx.doi.org/10.1007/978-1-4614-0064-6_1*

Chacko, S. M. and Kapila, V. (2019), An Augmented Reality Interface for Human-Robot Interaction in Unconstrained Environments, *in* 'IEEE International Conference on Intelligent Robots and Systems', pp. 3222–3228.
URL: *http://dx.doi.org/10.1109/IROS40897.2019.8967973*

Fotaris, P., Pellas, N., Kazanidis, I. and Smith, P. (2017), A systematic review of augmented reality game-based applications in primary education, *in* '11th European Conference on Games Based Learning (ECGBL)', Mini Track on Mixed Reality for Game-Based Learning proceedings, pp. 181–191.

Garzón, J., Pavón, J. and Baldiris, S. (2019), 'Systematic review and meta-analysis of augmented reality in educational settings', *Virtual Reality* **23**(4), 447–459.
URL: *http://dx.doi.org/10.1007/s10055-019-00379-9*

Hanafi, A., Elaachak, L. and Bouhorma, M. (2019), A comparative study of augmented reality sdks to develop an educational application in chemical field, *in* 'Proceedings of the 2nd International Conference on Networking, Information Systems Security', NISS19, Association for Computing Machinery, New York, NY, USA.
URL: *https://doi.org/10.1145/3320326.3320386*

Hedayati, H., Walker, M. and Szafir, D. (2018), Improving collocated robot teleoperation with augmented reality, *in* 'Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction', HRI '18, Association for Computing Machinery, New York, NY, USA, p. 78–86.
URL: *https://doi.org/10.1145/3171221.3171251*

Ismail, N. A., Wen, T. C., Salam, M. S., Nawi, A. M. and Mohamed, S. E. N. (2020), 'A review of visual inertial odometry for object tracking and measurement', *International Journal of Scientific and Technology Research* **9**(2), 355–361.
URL: *http://www.ijstr.org/final-print/feb2020/A-Review-Of-Visual-Inertial-Odometry-For-Object-Tracking-And-Measurement.pdf*

Johnson, A. S. and Sun, Y. (2013), Spatial augmented reality on person: Exploring the most personal medium, *in* R. Shumaker, ed., 'Virtual Augmented and Mixed Reality. Designing and Developing Augmented and Virtual Environments', Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 169–174.
URL: *https://doi.org/10.1007/978-3-642-39405-8_20*

Laamarti, F., Eid, M. and El Saddik, A. (2014), 'An overview of serious games', *International Journal of Computer Games Technology* **2014**.
URL: *10.1155/2014/358152*

Liarokapis, F. (2006), 'An exploration from virtual to augmented reality gaming', *Simulation & Gaming* **37**(4), 507–533.
URL: *https://doi.org/10.1177/1046878106293684*

Liarokapis, F., Greatbatch, I., Mountain, D., Gunesh, A., Brujic-Okretic, V. and Raper, J. (2005), Mobile augmented reality techniques for geovisualisation, *in* 'Ninth International Conference on Information Visualisation (IV'05)', pp. 745–751.
URL: *https://doi.org/10.1109/IV.2005.79*

Makhataeva, Z., Zhakatayev, A. and Varol, H. A. (2019), Safety Aura Visualization for Variable Impedance Actuated Robots, *in* '2019 IEEE/SICE International Symposium on System Integration (SII)', pp. 805–810.
URL: *https://doi.org/10.1109/SII.2019.8700332*

Milgram, P. (1994), 'A Taxonomy of Mixed Reality Visual Displays', *IEICE Transactions on Information Systems* **E77-D**(12), 1321–1329.

Nerurkar, Esha, L. S. and Zhao, S. (2017), 'System and method for concurrent odometry and mapping'.
URL: *https://patents.google.com/patent/US20170336511A1*

Orús, C., Ibáñez-Sánchez, S. and Flavián, C. (2021), 'Enhancing the customer experience with virtual and augmented reality: The impact of content and device type', *International Journal of Hospitality Management* **98**, 103019.
URL: *https://doi.org/10.1016/j.ijhm.2021.103019*

Rashid, Z., Melià-Seguí, J., Pous, R. and Peig, E. (2017), 'Using Augmented Reality and Internet of Things to improve accessibility of people with motor disabilities in the context of Smart Cities', *Future Generation Computer Systems* **76**, 248–261.
URL: *http://dx.doi.org/10.1016/j.future.2016.11.030*

Rolland, J. P. and Cakmakci, O. (2005), The past, present, and future of head-mounted display designs, *in* Y. Wang, Z. Weng, S. Ye and J. M. Sasian, eds, 'Optical Design and Testing II', Vol. 5638, International Society for Optics and Photonics, SPIE, pp. 368 – 377.
URL: *https://doi.org/10.1117/12.575697*

Silva, R., Oliveira, J. C. and Giraldi, G. A. (2003), 'Introduction to augmented reality'.
URL: *https://www.lncc.br/~jauvane/papers/RelatorioTecnicoLNCC-2503.pdf*

Wagner, D. and Schmalstieg, D. (2006), Handheld Augmented Reality Displays, *in* 'IEEE Virtual Reality Conference (VR 2006)', pp. 321–321.
URL: *https://doi.org/10.1109/VR.2006.67*

Zolotas, M. and Demiris, Y. (2019), Towards explainable shared control using augmented reality, *in* '2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', pp. 3020–3026.
URL: *https://doi.org/10.1109/IROS40897.2019.8968117*

Zolotas, M., Elsdon, J. and Demiris, Y. (2018), Head-mounted augmented reality for explainable robotic wheelchair assistance, *in* '2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', pp. 1823–1829.
URL: *https://doi.org/10.1109/IROS.2018.8594002*

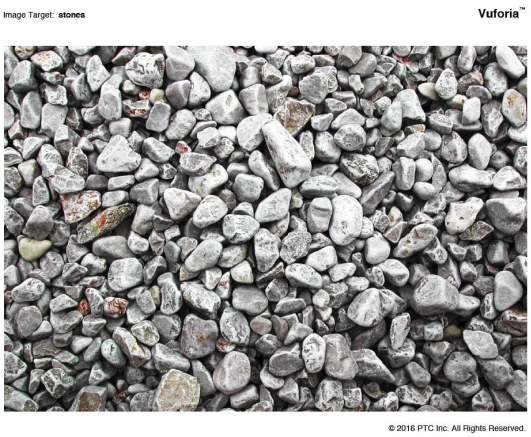
Appendix A

Evaluation of ARCore Motion Tracking Technology - Auxiliary Material

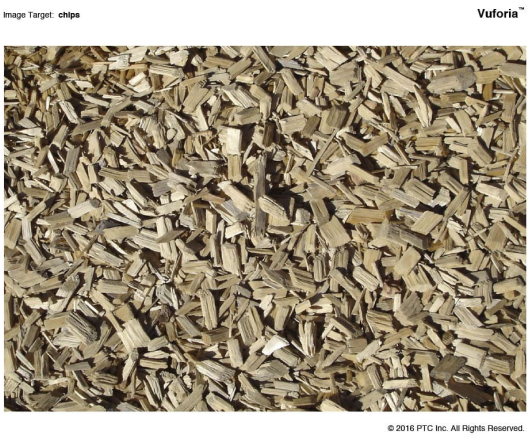
A.1 Images used for the Augmented Images Database

Figure A.1: Marker represented by the red circle in figures 3.2 and 3.3. Obtained from ARCore Augmented Images Webpage[1]. Score by ARCore: 100.

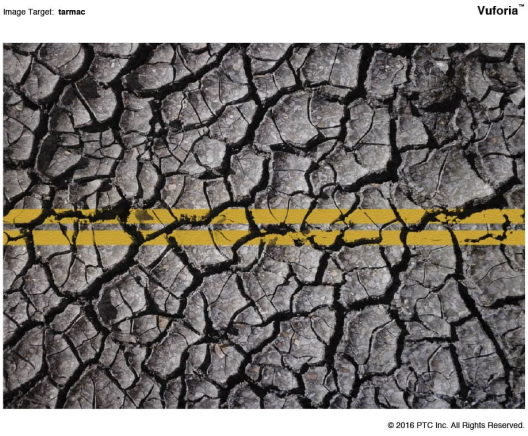
[1]https://developers.google.com/ar/develop/c/augmented-images



(a) Represented by the blue circle. Score by ARCore: 100.



(b) Represented by the yellow circle. Score by ARCore: 100.



(c) Represented by the green circle. Score by ARCore: 75.

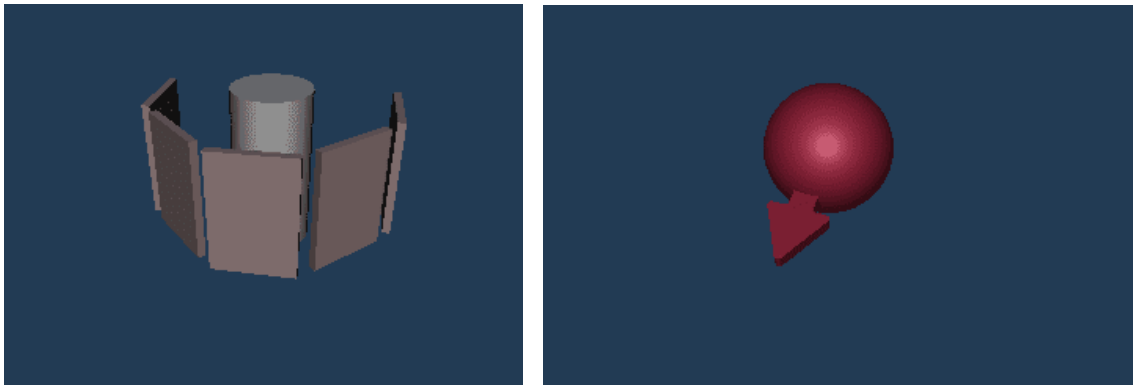
Figure A.2: Markers obtained from Vuforia samples[2]. Represented by circles in figures 3.2 and 3.3.

[2]https://developer.vuforia.com/downloads/samples

Appendix B

Virtual Object Models

B.1 Models for the Configuration



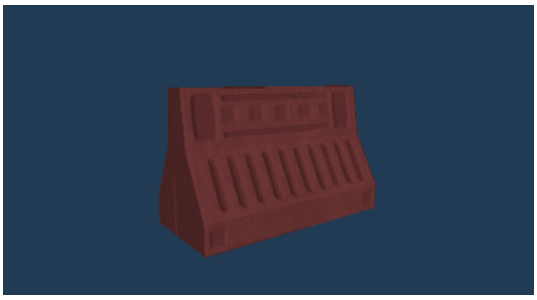
(a) Anchor representation. (b) Phone starting position representation.

Figure B.1: Models for the configuration

B.2 Models for the Augmented Reality Serious Game

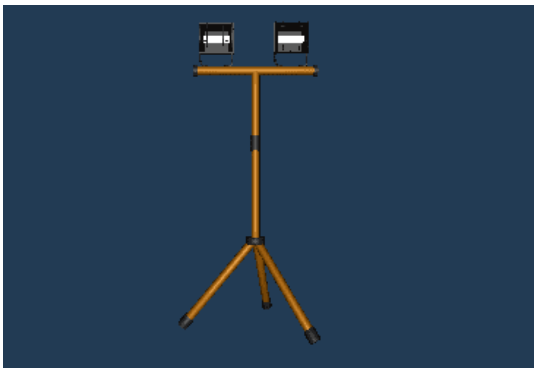
(a) Moving object.

(b) Directional arrow.



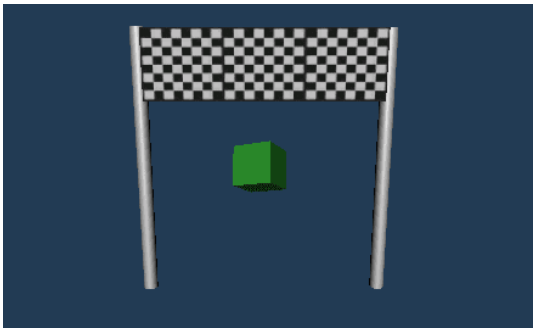
(c) Barrier.

(d) Cone.



(e) Spotlight.

(f) Stop sign.



(g) Passage point.

Figure B.2: Models for the ARSG

B.3 Models for the Game Performance Visualization

Figure B.3: Wheelchair. This model is only a representation for the motorized wheelchair, it does not correspond to its authentic model.

Appendix C

UI

C.1 Smartphone Application UI

Figure C.1: Smartphone application main menu.

Figure C.2: User manuals menu.

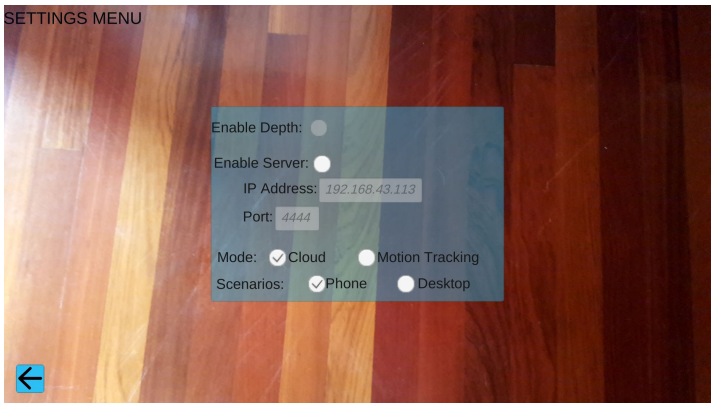


Figure C.3: Settings menu.

C.1.1 Smartphone Application UI - Configuration

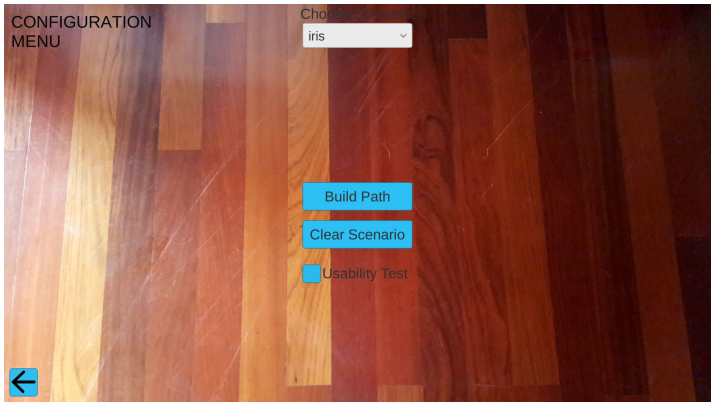


Figure C.4: Smartphone configuration menu.

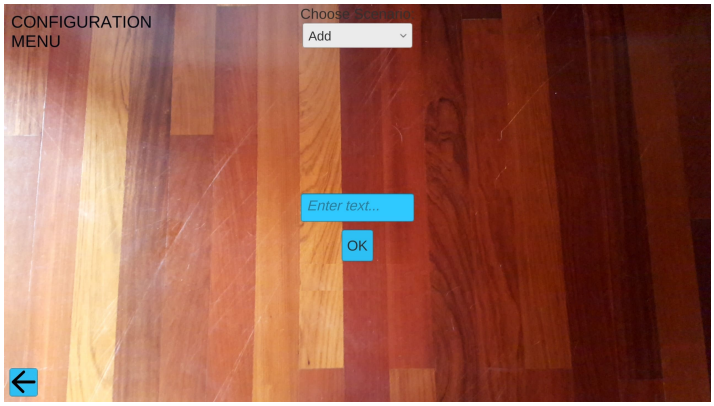


Figure C.5: Create a new scenario.

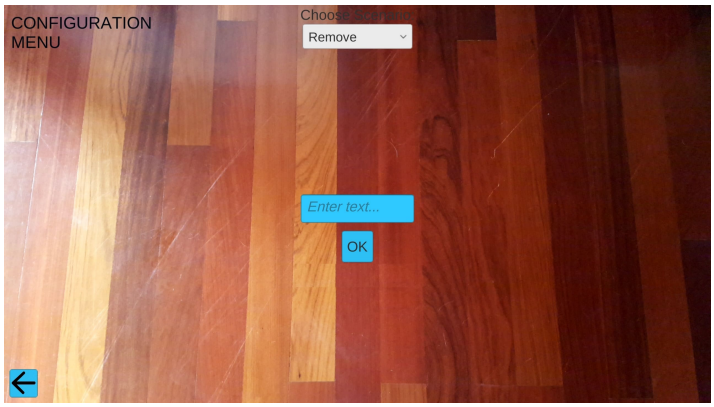


Figure C.6: Remove a created scenario.

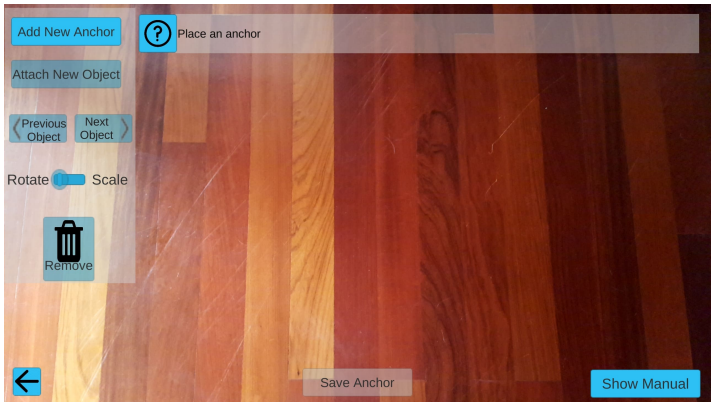


Figure C.7: Configuration starting interface. Specific to the cloud anchor approach.

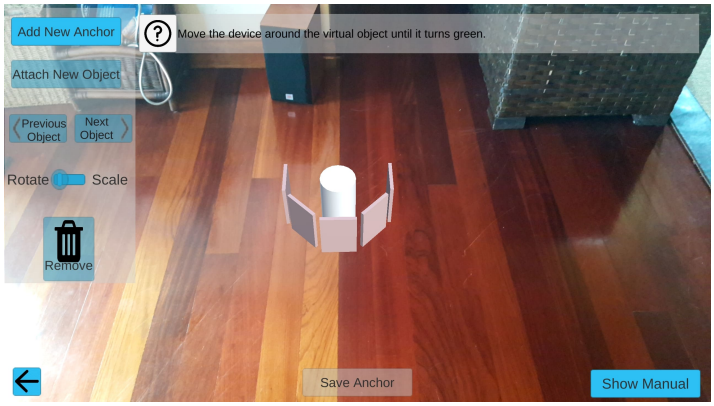


Figure C.8: Interface with an anchor.

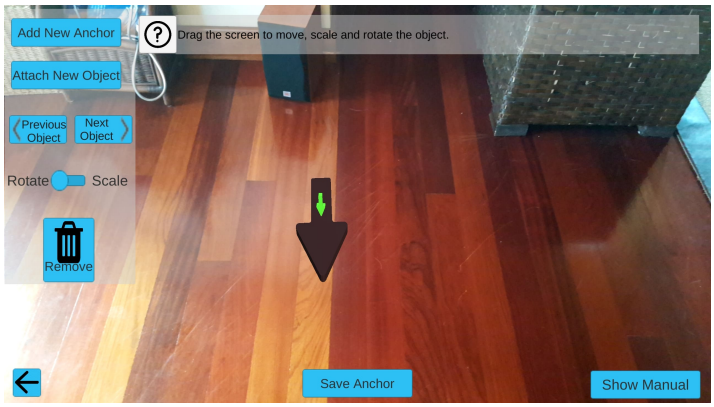


Figure C.9: Interface with a virtual object selected.

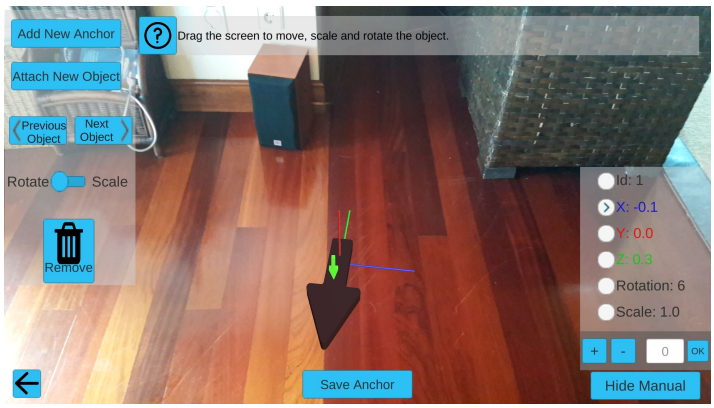


Figure C.10: Virtual object selected and the manual insertion menu open.

C.1.2 Smartphone Application UI - Game

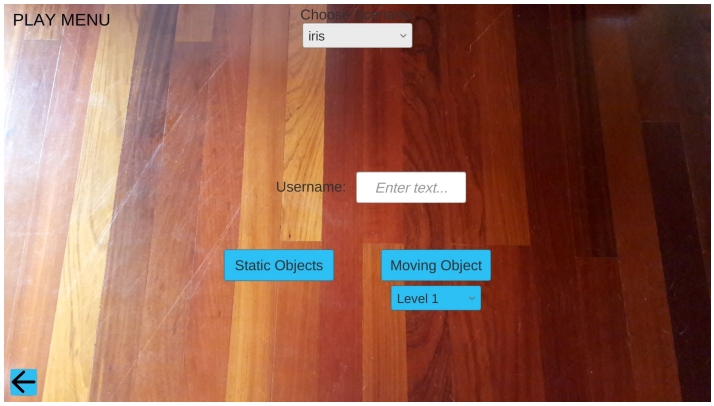


Figure C.11: Smartphone game menu.

Figure C.12: Playing a game with static objects.

Figure C.13: Playing a game with the moving object.

C.2 Desktop PC Application UI

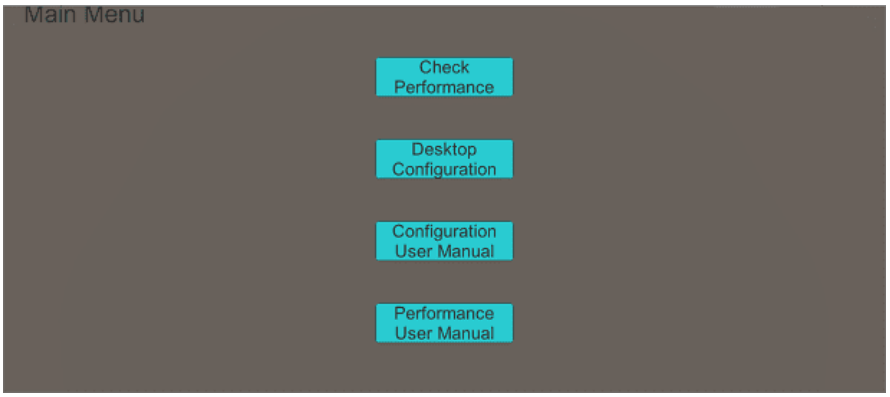


Figure C.14: Desktop PC application main menu.

C.2.1 Desktop PC Application UI - Configuration

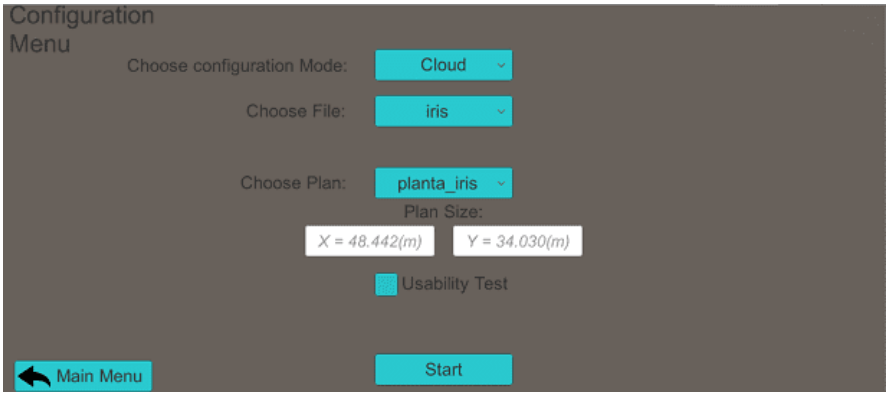


Figure C.15: Desktop PC configuration menu. If the configuration mode is *Motion Tracking* instead of *Cloud* the *Choose File* drop-down list becomes an input field.

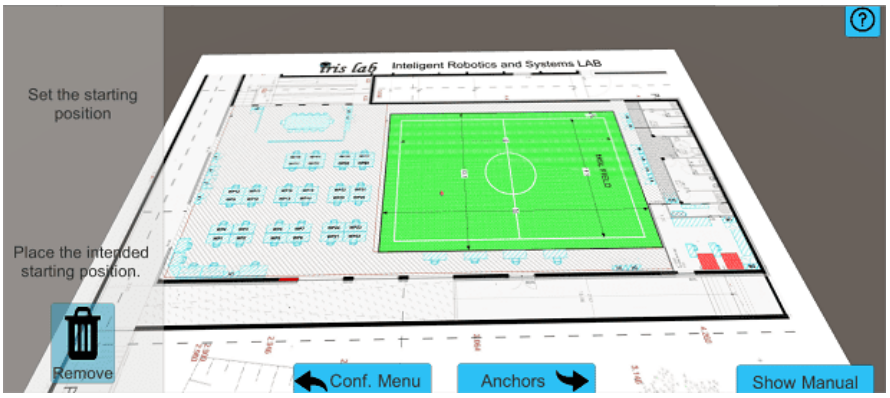


Figure C.16: Positioning the virtual object representing the smartphone starting position.

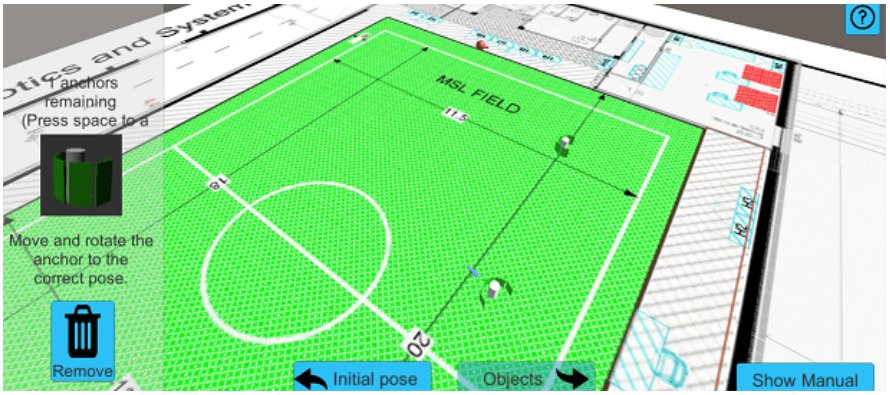


Figure C.17: Anchor placement.

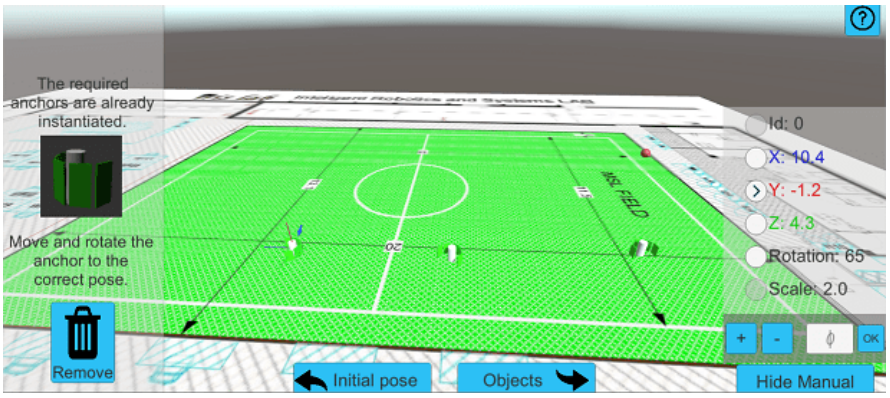


Figure C.18: Interface when every anchor is placed, with the manual insertion menu opened.

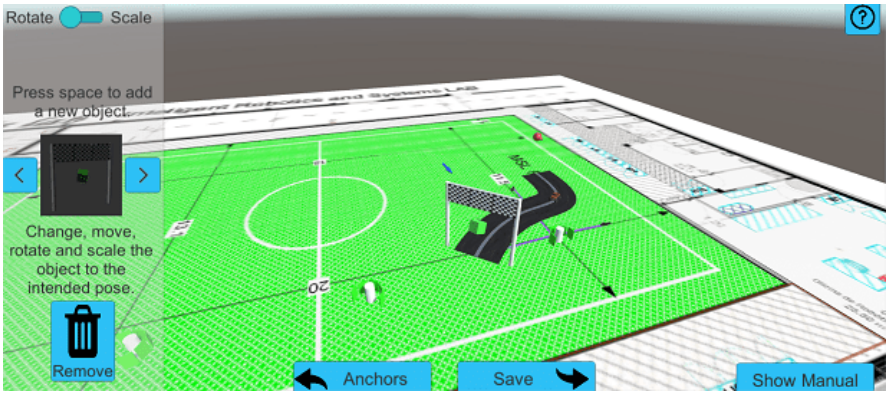


Figure C.19: Virtual object configuration.

C.2.2 Desktop PC Application UI - Performance Visualization

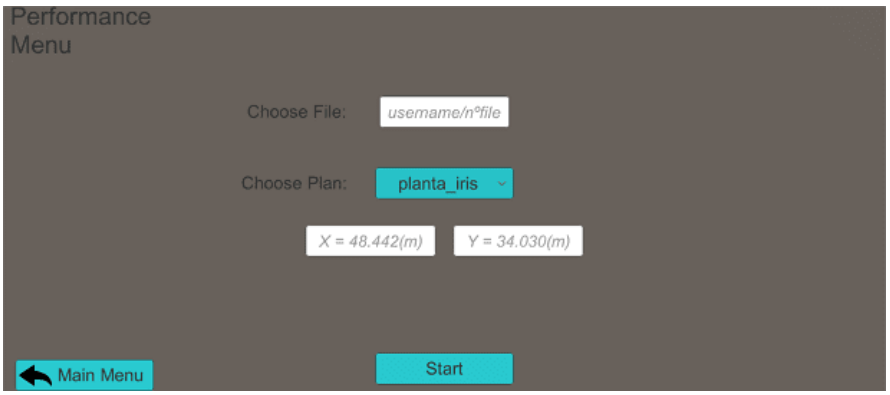


Figure C.20: Desktop PC performance visualization menu.

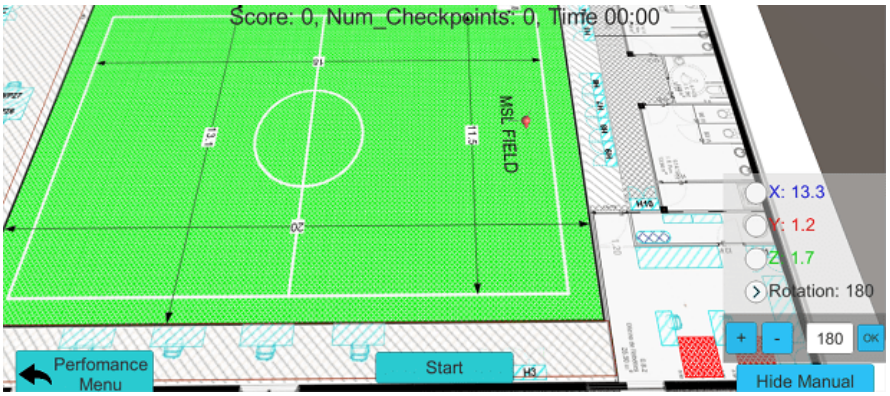


Figure C.21: Positioning the virtual object representing the smartphone starting position in the played game.

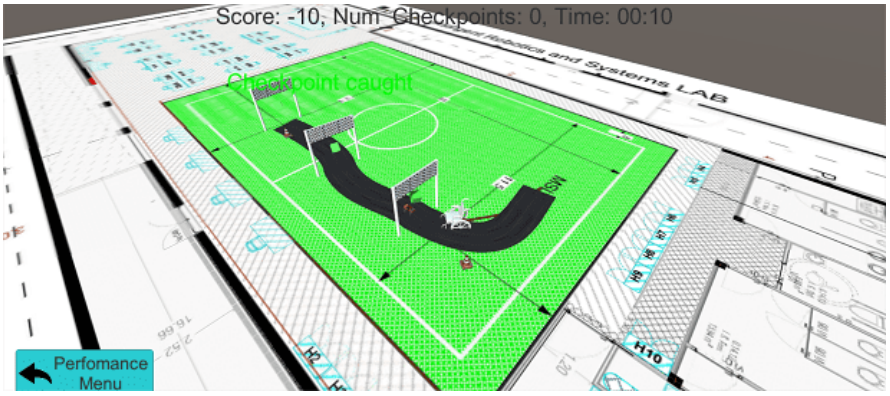


Figure C.22: Visualizing the game performance.

Appendix D

Application User Manual

D.1 Wheelchair User Manual

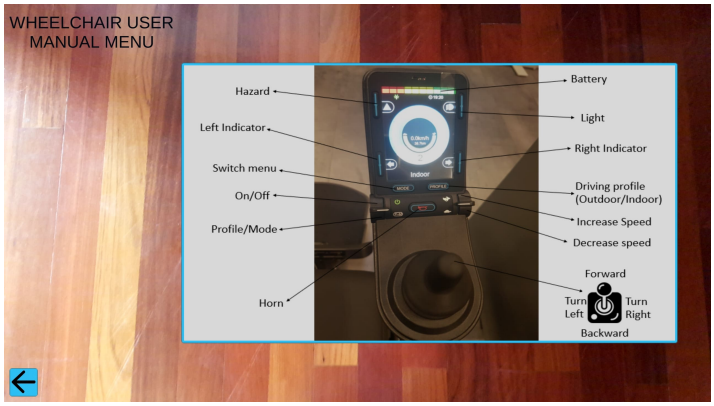


Figure D.1: Wheelchair controller.

D.2 Smartphone Configuration User Manual

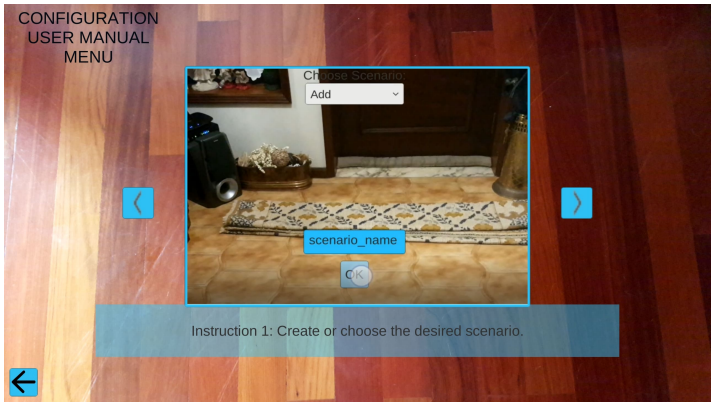


Figure D.2: First instruction. *"Create or choose the desired scenario."*

Figure D.3: Second instruction. Specific for the cloud anchors approach. *"Choose a place with plenty of texture to add a new anchor."*

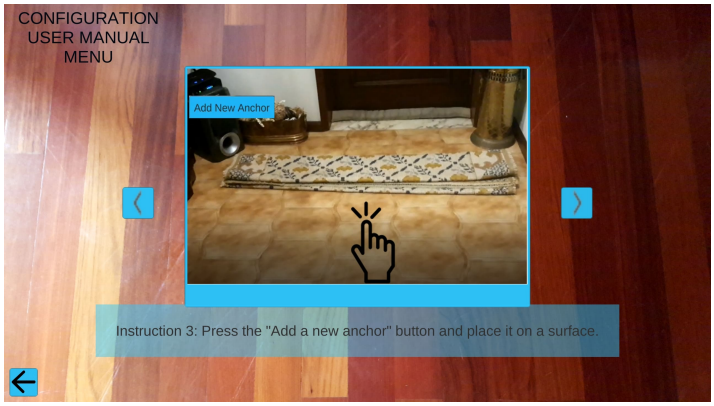


Figure D.4: Third instruction. *"Press the "Add a new anchor" button and place it on a surface."*

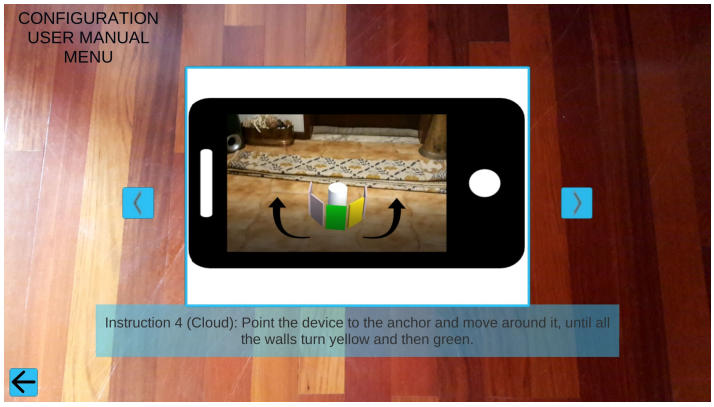


Figure D.5: Fourth instruction. Specific for the cloud anchors approach. *"Point the device to the anchor and move around it, until all the walls turn yellow and then green."*

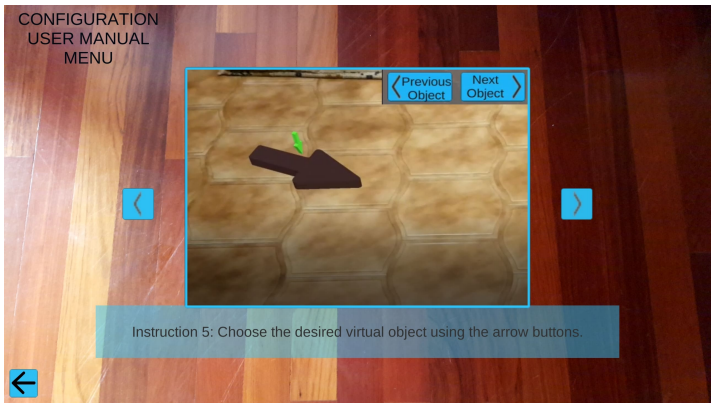


Figure D.6: Fifth instruction. *"Choose the desired virtual object using the arrow buttons."*

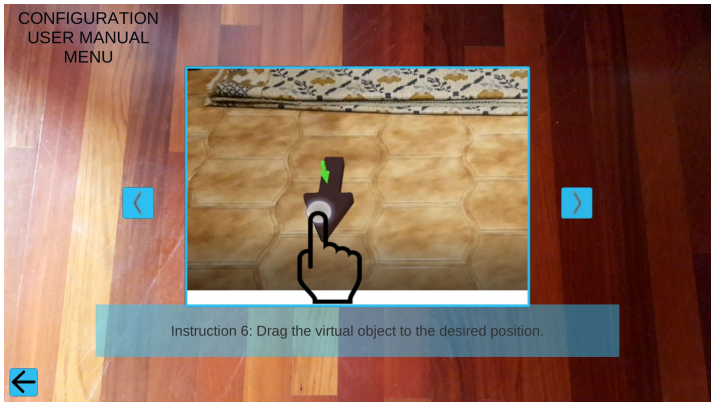


Figure D.7: Sixth instruction. *"Drag the virtual object to the desired position."*

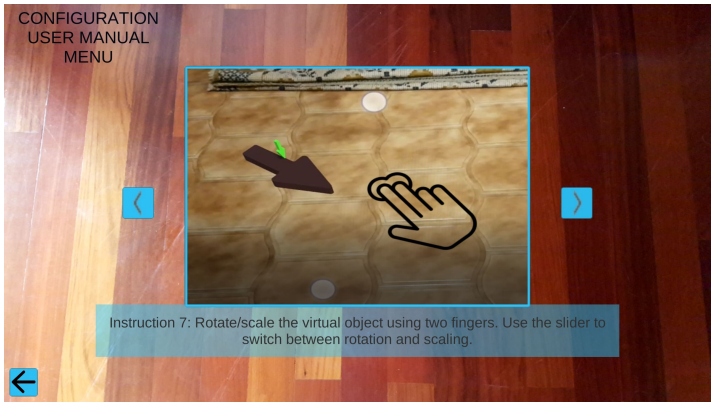


Figure D.8: Seventh instruction. *"Rotate/scale the virtual object using two fingers. Use the slider to switch between rotation and scaling."*

Figure D.9: Eighth instruction. *"Use the manual menu granting more precision."*

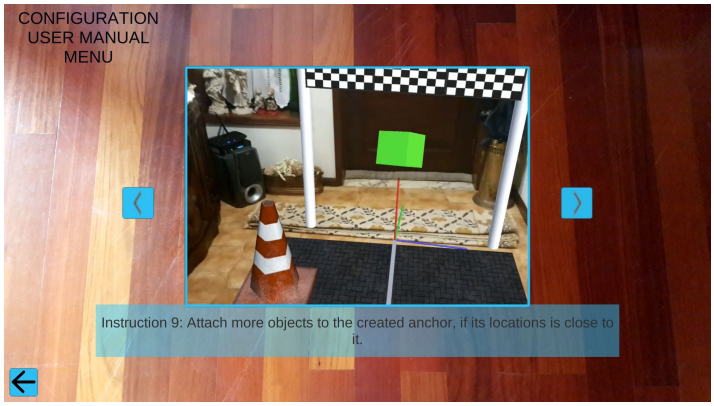


Figure D.10: Ninth instruction. *"Attach more objects to the created anchor, if their locations are close to it."*

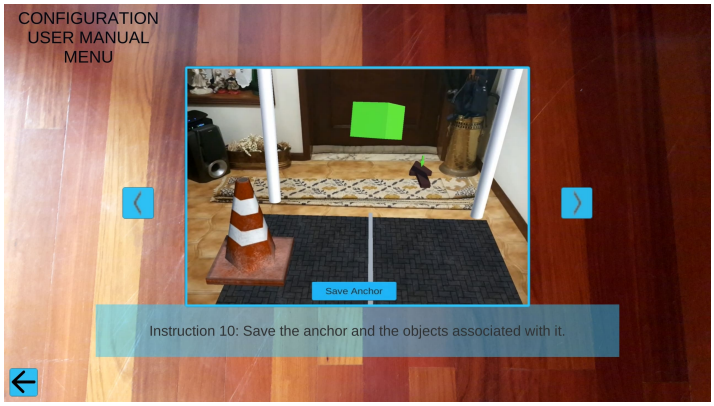


Figure D.11: Tenth instruction. *"Save the anchor and the objects associated with it."*

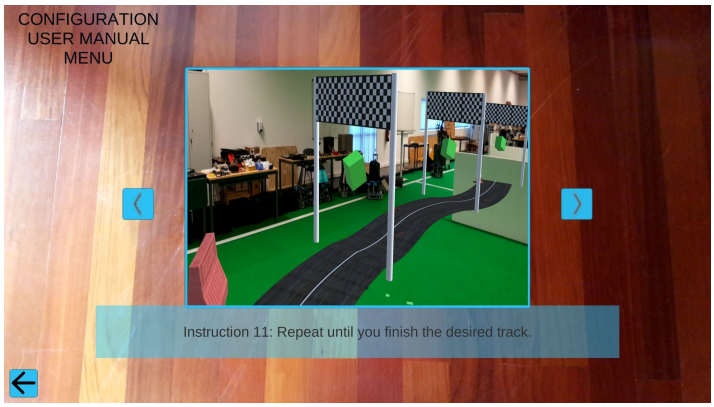


Figure D.12: Eleventh instruction. *"Repeat until you finish the desired track."*

D.3 Desktop PC Configuration User Manual

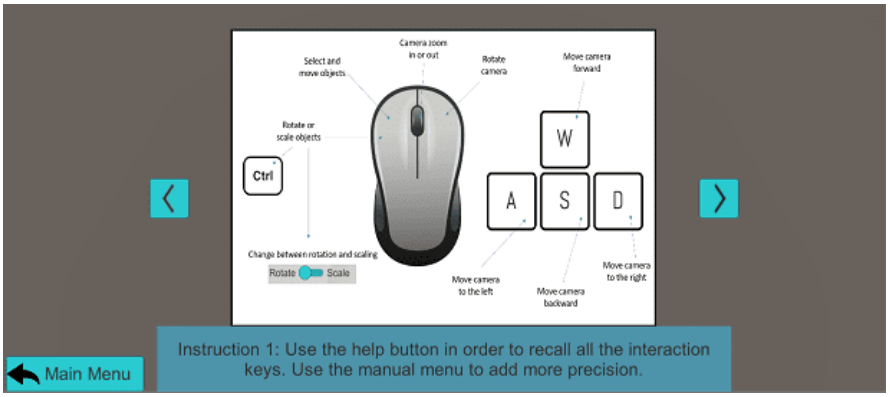


Figure D.13: First instruction. *"Use the help button in order to recall all the interaction keys. Use the manual menu to add more precision."*

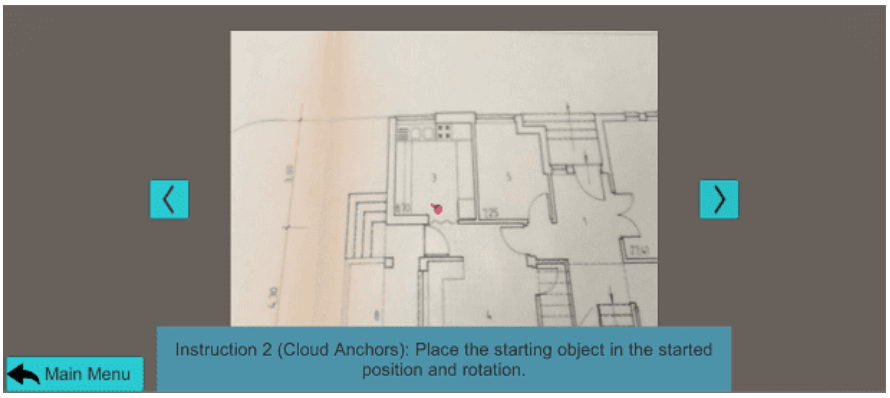


Figure D.14: Second instruction. Specific for the cloud anchors approach. *"Place the starting object in the started position and rotation."*

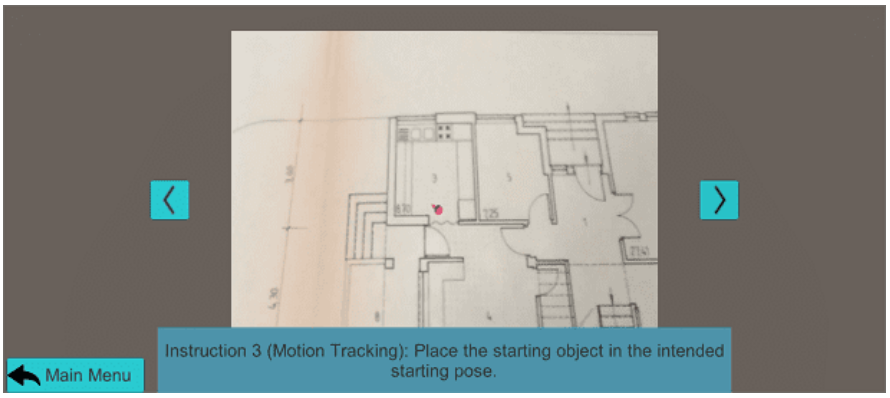


Figure D.15: Third instruction. Specific for the motion tracking approach. *"Place the starting object in the intended starting pose."*

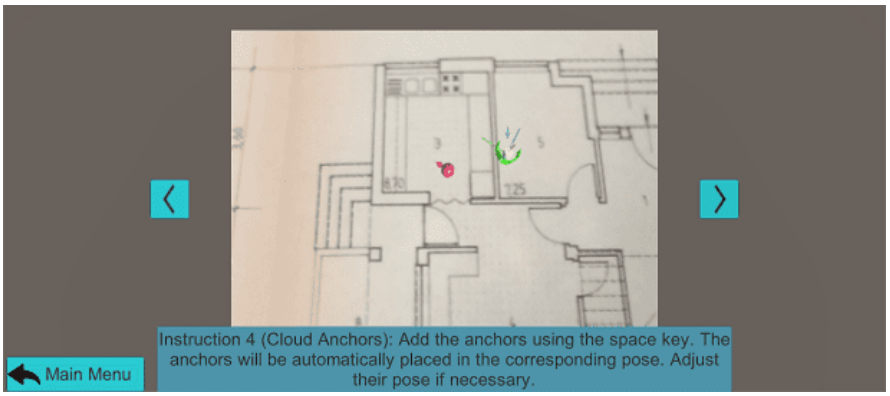


Figure D.16: Fourth instruction. Specific for the cloud anchors approach. *"Add the anchors using the space key. The anchors will be automatically placed in the corresponding pose. Adjust their pose if necessary."*

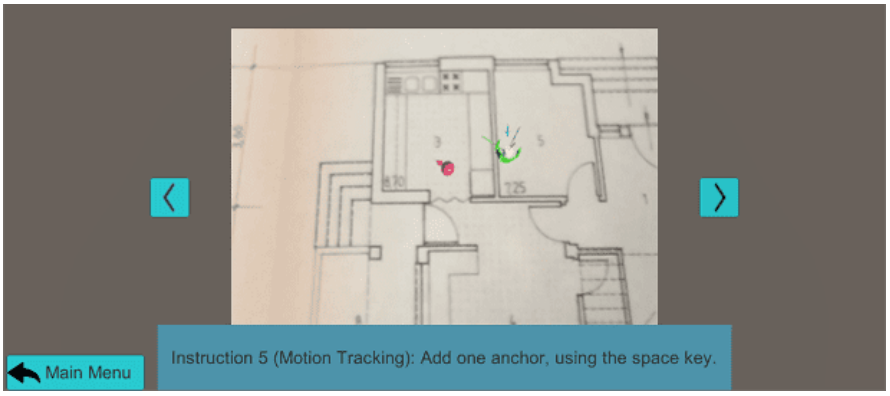


Figure D.17: Fifth instruction. Specific for the motion tracking approach. *"Add one anchor, using the space key."*

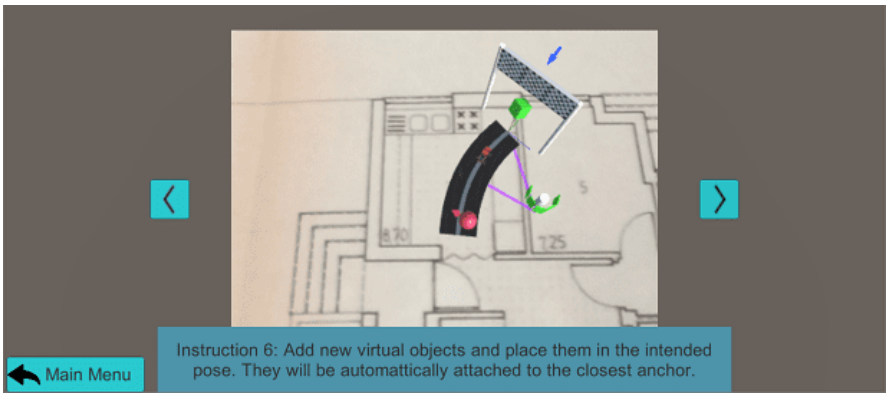


Figure D.18: Sixth instruction. *"Add new virtual objects and place them in the intended pose. They will be automatically attached to the closest anchor."*

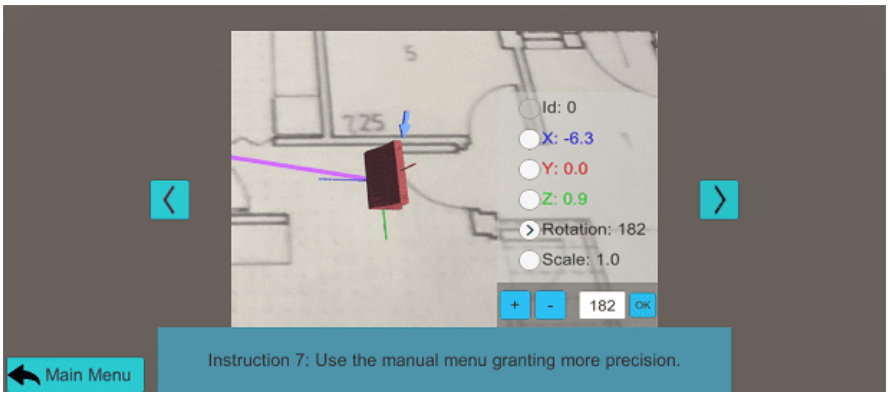


Figure D.19: Seventh instruction. *"Use the manual menu granting more precision."*

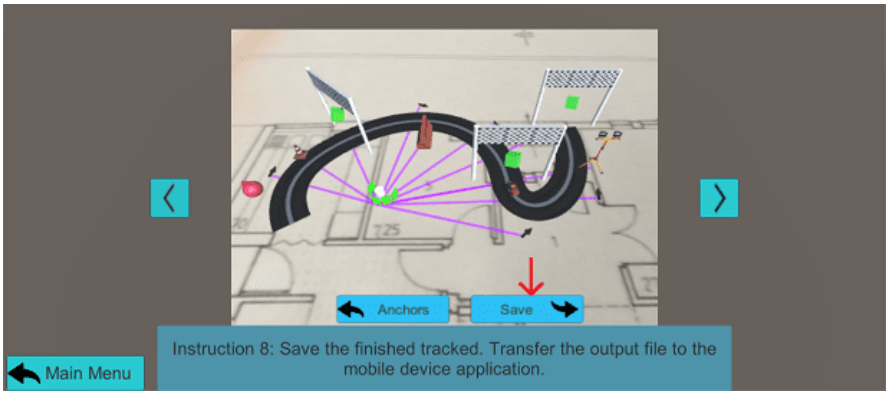


Figure D.20: Eighth instruction. *"Save the finished track. Transfer the output file to the mobile device application."*

D.4 Game User Manual

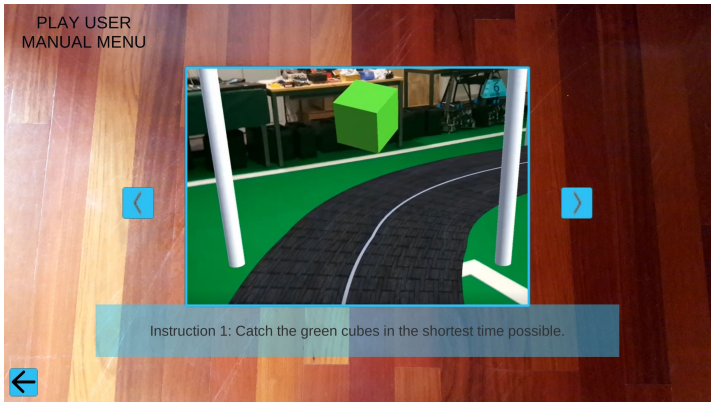


Figure D.21: First instruction. *"Catch the green cubes in the shortest time possible."*

Figure D.22: Second instruction. Specific for the static objects game. *"Follow the direction of the arrows as close as possible to the white line."*

Figure D.23: Third instruction. Specific for the moving object game. *"Follow the car as close to it as possible."*

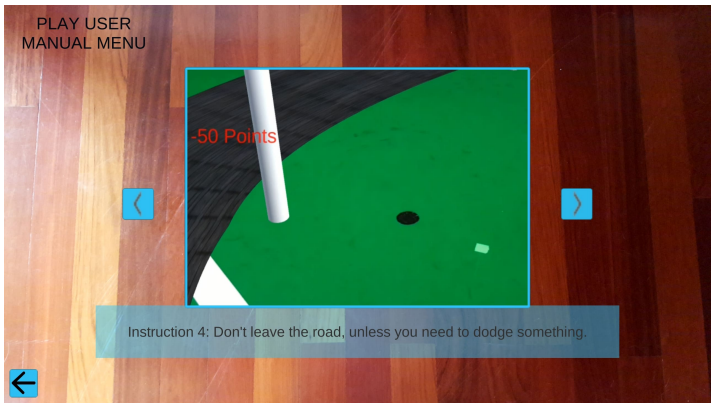


Figure D.24: Fourth instruction. *"Do not leave the road, unless you need to dodge something."*

Figure D.25: Fifth instruction. *"Dodge the road obstacles or they will explode."*

Figure D.26: Sixth instruction. *"Look away from the spotlight or you will get blinded."*

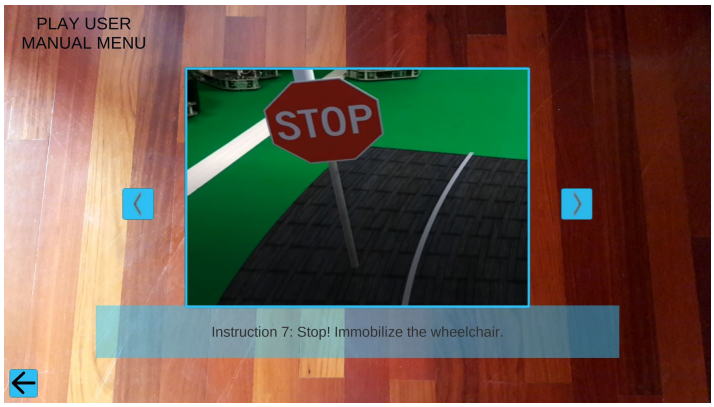


Figure D.27: Seventh instruction. *"Stop! Immobilize the wheelchair."*

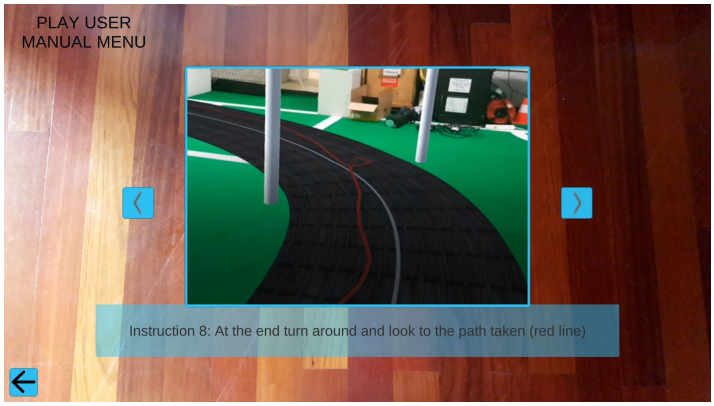


Figure D.28: Eighth instruction. *"At the end turn around and look to the path taken (red line)."*

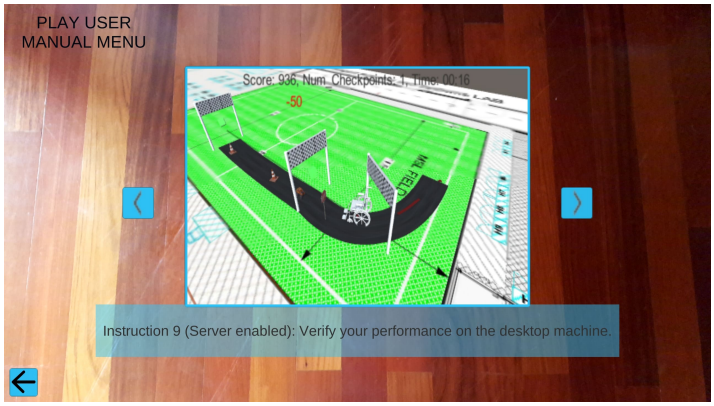


Figure D.29: Ninth instruction. *"Verify your performance on the desktop machine."*

D.5 Performance Visualization User Manual

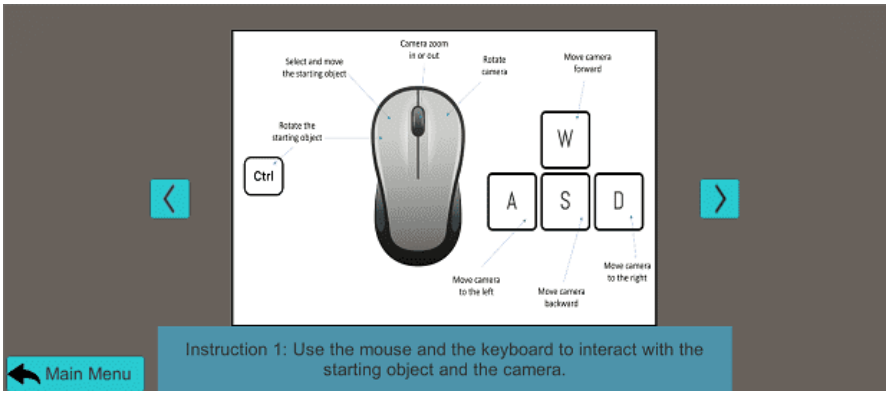


Figure D.30: First instruction. *"Use the mouse and the keyboard to interact with the starting object and the camera."*

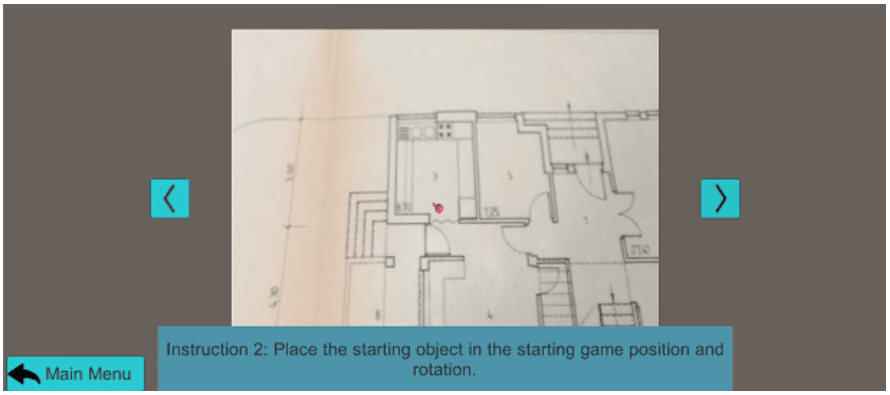


Figure D.31: Second instruction. *"Place the starting object in the starting game position and rotation."*

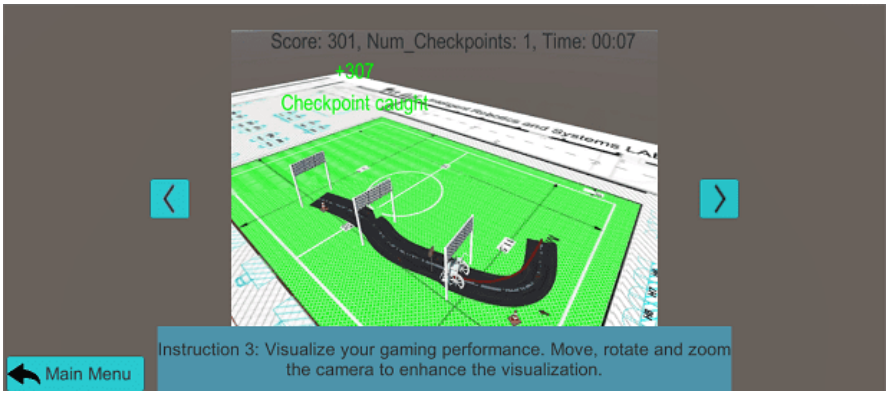


Figure D.32: Third instruction. *"Visualize your gaming performance. Move, rotate and zoom the camera to enhance the visualization."*

Appendix E

Unity Library - User Manual

E.1 Contents

The package includes the following folders:

- GoogleARCore folder: ARCore Unity SDK version 1.20.0.
- *Resources/Plan* folder: Includes the files (images and corresponding width and height) for the scale comparison. Any image with this goal can be added and it will be automatically recognized by the library.
  - *plan_sizes.txt* file: Might contain the image width (x) and height (y) for dynamic purposes. After a new image being added, the file can be updated to contain its real size, adding a new line (*image_name: x=image_real_width,y=image_real_height*).
  - *planta_casa.png*, *planta_iris.png* images: Plan examples for scale comparison.
- Scripts folder: C# scripts to transfer the game for Android an add all the required ARCore components.

E.2 Usage

Instruction to use this package:

1. Import the package "*gameLibraryPackage.unitypackage*". *Assets* → *Import Package* → *Custom Package...* and select the corresponding package (figure E.1).

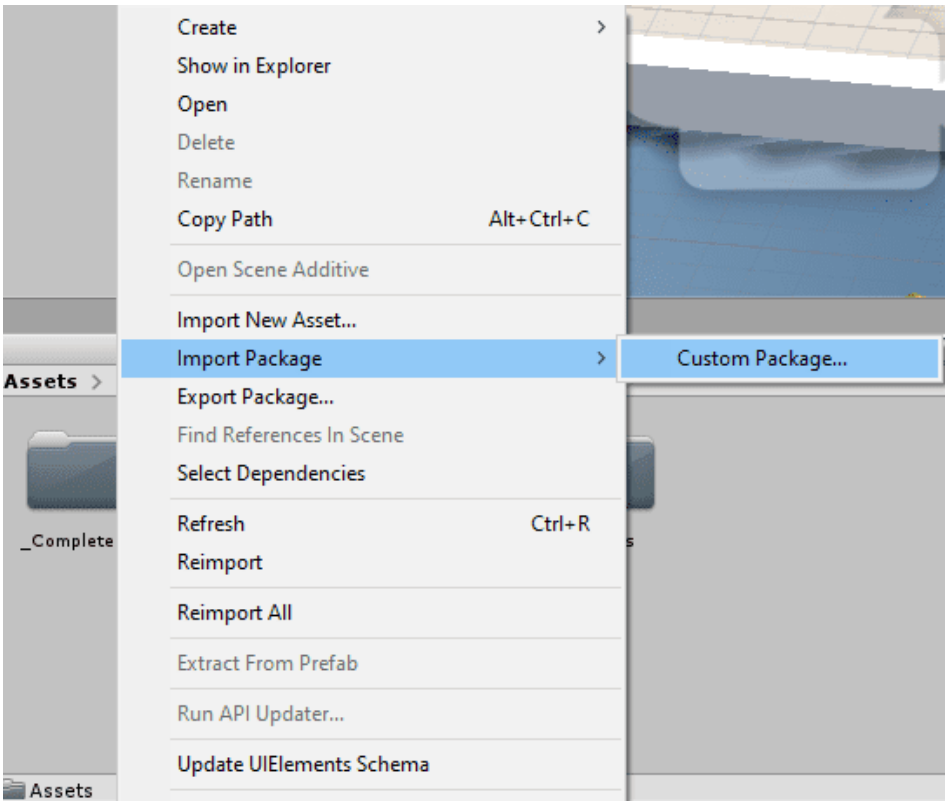


Figure E.1: Import the gaming library package into the Unity application.

2. Tag the intended *GameObject* player with the tag "*Player*", if any (it is possible to be just a spectator) (figure E.2).

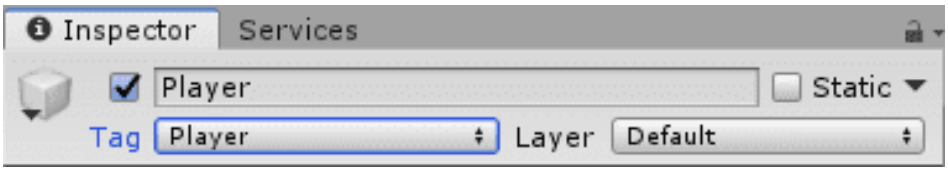


Figure E.2: Tag player *GameObject*.

3. Transfer the game to Android platform and associate the player to ARCore and to the smartphone camera. For this, just use the menu: *ARCore game library* → *Create game for ARCore* (figure E.3).

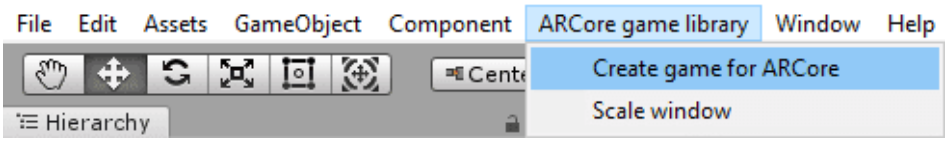


Figure E.3: Transform the game into an ARG.

4. Open the scaling window, if any adjustment to the scene scale must be made to fit in the real

world (figure E.4).

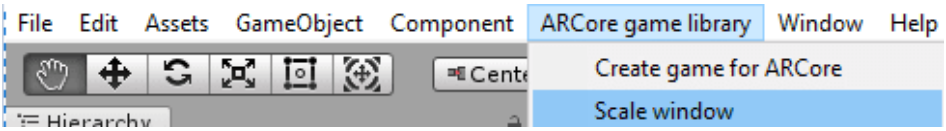


Figure E.4: Open the scale window.

5. Choose the scale comparison image, if any and set its width and height (figure E.5 → *Show plan*).

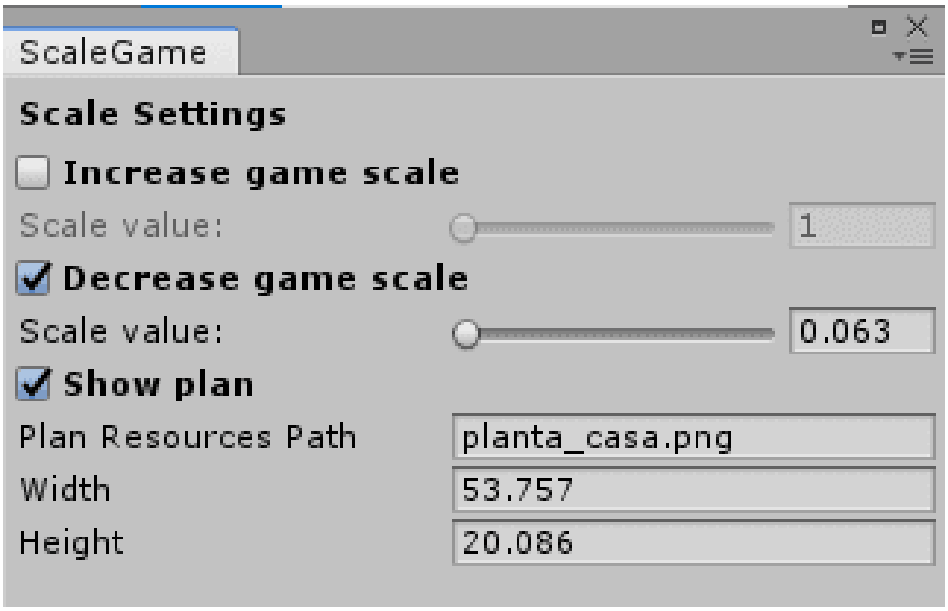


Figure E.5: Scaling window.

6. Increase or decrease the scene scale (figure E.5)
7. Change the "*arcoreHandler*" *GameObject* position and rotation to the intended starting pose on the real world, relatively to the game scene.
8. *Build and Run* and enjoy the game in the AR environment.

Appendix F

Usability Tests

F.1 System Usability Scale Questionnaire for the Preliminary Usability Tests

#	Question	Game					Configuration				
		Strongly disagree				Strongly agree	Strongly disagree				Strongly agree
		1	2	3	4	5	1	2	3	4	5
1	I think that I would like to use this application frequently for its usage scenarios.										
2	I found the application unnecessarily complex.										
3	I thought the application was easy to use.										
4	I think I would need the support of a person to be able to use this application.										
5	I found the various functions in this application were well integrated.										
6	I thought there was too much inconsistency in the application.										
7	I think that most people would learn to use the application very quickly.										
8	I found the application very cumbersome to use.										
9	I felt very confident using the application.										
10	I needed to learn a lot of things before I could get going with this application.										

Figure F.1: Adapted system usability questionnaire to be applied in the preliminary usability tests.

F.2 Questionnaire for the Final Application Usability Tests

Consent statement

Personal Information

All information provided is confidential and will not be distributed or used for any purpose outside of this test.

1. **Participant ID**: _____
2. **Age**: _____
3. **Gender**:
   - □ Male
   - □ Female
4. **Occupation and area**: _____
5. **Have you had contact with augmented reality applications before?**
   - □ Yes
   - □ No
6. **Have you had contact with indoor navigation applications before?**
   - □ Yes
   - □ No

Post-experiment questionnaire - Desktop PC configuration

Complete the questionnaire, based on the experiment you previously completed.

7. **The manipulation and interaction are intuitive.**

   Completely disagree □—□—□—□—□ Completely agree
8. **It is easy to place the virtual object in the correct position.**

   Completely disagree □—□—□—□—□ Completely agree
9. **It is easy to place the virtual object in the correct orientation.**

   Completely disagree □—□—□—□—□ Completely agree
10. **It is easy to place the virtual object in the correct scale.**

    Completely disagree □—□—□—□—□ Completely agree
11. **The manipulation and interaction have irritating characteristics.**

    Completely disagree □—□—□—□—□ Completely agree
12. **If you have found irritating characteristics, please indicate which ones**:

    _____
13. **Manipulation and interaction could improve with training.**

    Completely disagree □—□—□—□—□ Completely agree

14. **Consulting the user manual is enough to be able to use this method of configuration.**

    Completely disagree □—□—□—□—□ Completely agree

15. **Indicate the degree of physical demand you felt using this method of configuration.**

    Low demanding □—□—□—□—□ High demanding

16. **Indicate the degree of mental demand you felt using this method of configuration.**

    Low demanding □—□—□—□—□ High demanding

17. **How satisfied are you relatively to this configuration method?**

    Completely unsatisfied □—□—□—□—□ Completely satisfied

18. **What were the main difficulties?**

    _____

19. **Add any type of observations relatively to this configuration method.**

    _____

Post-experiment questionnaire - Smartphone configuration

Complete the questionnaire, based on the experiment you previously completed.

20. **The manipulation and interaction are intuitive.**

    Completely disagree □—□—□—□—□ Completely agree

21. **It is easy to place the virtual object in the correct position.**

    Completely disagree □—□—□—□—□ Completely agree

22. **It is easy to place the virtual object in the correct orientation.**

    Completely disagree □—□—□—□—□ Completely agree

23. **It is easy to place the virtual object in the correct scale.**

    Completely disagree □—□—□—□—□ Completely agree

24. **The manipulation and interaction have irritating characteristics.**

    Completely disagree □—□—□—□—□ Completely agree

25. **If you have found irritating characteristics, please indicate which ones**:

    _____

26. **Manipulation and interaction could improve with training.**

    Completely disagree □—□—□—□—□ Completely agree

27. **Consulting the user manual is enough to be able to use this method of configuration.**

    Completely disagree □—□—□—□—□ Completely agree

28. **Indicate the degree of physical demand you felt using this method of configuration.**

    Low demanding □—□—□—□—□ High demanding

29. **Indicate the degree of mental demand you felt using this method of configuration.**

Low demanding □—□—□—□—□ High demanding

30. **How satisfied are you relatively to this configuration method?**

    Completely unsatisfied □—□—□—□—□ Completely satisfied

31. **What were the main difficulties?**

    _____

32. **Add any type of observations relatively to this configuration method.**

    _____

Post-experiment questionnaire - Game with static objects.

Complete the questionnaire, based on the experiment you previously completed.

33. **The interaction with virtual objects in the game is intuitive.**

    Completely disagree □—□—□—□—□ Completely agree

34. **It is easy to complete the game.**

    Completely disagree □—□—□—□—□ Completely agree

35. **The game has irritating characteristics.**

    Completely disagree □—□—□—□—□ Completely agree

36. **If you have found irritating characteristics, please indicate which ones**:

    _____

37. **The game performance can improve with time.**

    Completely disagree □—□—□—□—□ Completely agree

38. **How satisfied are you relatively to this game mode?**

    Completely unsatisfied □—□—□—□—□ Completely satisfied

39. **The game helped in the interaction with the robotic wheelchair.**

    Completely disagree □—□—□—□—□ Completely agree

40. **Visualizing your game performance helped in understanding how to improve the wheelchair control.**

    Completely disagree □—□—□—□—□ Completely agree

41. **Consulting the user manual is enough to play this game mode.**

    Completely disagree □—□—□—□—□ Completely agree

42. **What were the main difficulties?**

    _____

43. **Add any type of observations relatively to this configuration method.**

    _____

Post-experiment questionnaire - Game with the moving object.

Complete the questionnaire, based on the experiment you previously completed.

44. The interaction with virtual objects in the game is intuitive.

Completely disagree □—□—□—□—□ Completely agree

45. It is easy to complete the game.

Completely disagree □—□—□—□—□ Completely agree

46. The game has irritating characteristics.

Completely disagree □—□—□—□—□ Completely agree

47. If you have found irritating characteristics, please indicate which ones:

_____

48. The game performance can improve with time.

Completely disagree □—□—□—□—□ Completely agree

49. How satisfied are you relatively to this game mode?

Completely unsatisfied □—□—□—□—□ Completely satisfied

50. The game helped in the interaction with the robotic wheelchair.

Completely disagree □—□—□—□—□ Completely agree

51. Visualizing your game performance helped in understanding how to improve the wheelchair control.

Completely disagree □—□—□—□—□ Completely agree

52. Consulting the user manual is enough to play this game mode.

Completely disagree □—□—□—□—□ Completely agree

53. What were the main difficulties?

_____

54. Add any type of observations relatively to this configuration method.

_____

Post-experiment questionnaire - Method comparison.

55. What configuration method did you find most intuitive to use?

□ Desktop PC

□ Smartphone

□ Equally intuitive

56. What configuration method did you find most efficient to use?

□ Desktop PC

□ Smartphone

□ Equally efficient

57. What configuration method did you find the most precise?

- □ Desktop PC
- □ Smartphone
- □ Equally precise

58. Did you used manual insertion mode?

- □ Yes
- □ No

59. If you used the manual insertion mode. Did you preferred the visual interaction mode or the manual insertion mode?

- □ Visual interaction mode
- □ Manual insertion mode
- □ Each interaction mode has its advantages

60. Which of the games has the objects that best helped you to realize the purpose of the game?

- □ Game with static objects
- □ Game with the moving object
- □ Helped equally

61. Which of the games you appreciated the most?

- □ Game with static objects
- □ Game with the moving object
- □ Equally appreciated

62. Which of the games helped you the most in learning to control the robotic wheelchair?

- □ Game with static objects
- □ Game with the moving object
- □ Helped equally