A Robust Hybrid AR Spatial Mapping Application

A thesis submitted for the degree of Computer Games Technology (MSc)

by

Marco Longo

School of Design and Informatics,
Abertay University.

August, 2020

Abstract

Mobile applications that make use of Augmented Reality (AR) usually rely on two main techniques: markerless AR and marker-based AR. Each technique has its strengths and weaknesses and is therefore suitable for a specific kind of application. Generally, markerless AR and marker-based AR are implemented independently.

Although markerless AR is widely used nowadays, it presents some weak spots that are related to the assumptions that the technique makes regarding the real-world environment. The markerless approach usually fails when employed in featureless environments that do not provide enough information about the relative motion of the AR-enabled device in the physical space.

This thesis presents an alternative approach to AR, which allows the creation of a complete virtual map of an environment by using a single marker and a set of predetermined distances. Given that the hybrid AR system does not rely on feature detection by the device's camera, motion tracking is achieved by solely relying on the device embedded sensors, such as accelerometer and gyroscope. This introduces a series of limitations that are mostly related to the well-known inaccuracy of these sensors.

The results show that the hybrid AR approach cannot provide accurate enough results to offer a pleasing augmented reality experience. Although it is in theory possible to evaluate the device displacement by integrating the acceleration values coming in from the sensors, the inaccuracy of the results makes that the output cannot be used in practice. The calculated values are compromised by the heavy approximation that is required to perform the integration and by the constant noise that affects the sensors. Since the device position cannot be tracked in a reliable way, the real-world environment cannot be mapped correctly, and therefore virtual objects cannot be correctly placed and held in fixed positions in the virtual scene.

Table of Contents

Abstract ..... **ii**

List of Figures **1**

List of Tables **3**

1 Introduction **4**

2 Literature Review **8**

2.1 Historical Overview 8

2.2 Types of Augmented Reality 11

2.3 Virtual 3D Reconstruction: Problems and Solutions 15

2.4 Virtual 3D Reconstruction: Case Studies 17

3 Methodology **20**

3.1 Tools and General Approach 20

3.2 The ARCore approach 21

3.3 First approach: acceleration array 22

3.4 Second approach: cumulative acceleration 25

4 Evaluation and Discussion **28**

4.1 Testing methodology 28

4.2 Sensors noise evaluation 30

4.3 ARCore approach testing 34

4.4 First approach testing 40

4.5 Second approach testing 44

4.6 Data comparison 48

5 Conclusions and Future Work **53**

References: **55**

Appendices **57**

Appendix A – Application Demo 57

Appendix B – Tests Raw Data 57

List of Figures

Figure 1: A teleprompter system 8
Figure 2: Videoplace by Myron Krueger 9
Figure 3: ARQuake by Bruce Thomas et al. 10
Figure 4: Ethnobotany Workbook by McGrath et al. 13
Figure 5: IKEA Place app 15
Figure 6: Accumulated registered point cloud from lidar SLAM 16
Figure 7: KinectFusion 3D reconstruction of a static body 17
Figure 8: KinectFusion 3D reconstruction of a moving body 18
Figure 9: Real-time reconstructions of a dynamic scene 19
Figure 10: ARCore's Augmented Images Flowchart 21
Figure 11: Testing methodology flowchart 29
Figure 12: Normality test results for the raw acceleration samples 30
Figure 13: Raw acceleration data boxplot – X, Y and Z Coordinates 31
Figure 14: Illustration of device coordinate frame 31
Figure 15: Normality test results for the refined acceleration samples 32
Figure 16: Wilcoxon Signed Ranks test result for the acceleration datasets 33
Figure 17: Normality test results for the ARCore approach displacement samples 35
Figure 18: ARCore approach displacement boxplot - X Coordinates 36
Figure 19: Results of the one-sample t-test for the ARCore approach dataset 36
Figure 20: Normality test results, ARCore approach featureless environment 38
Figure 21: Boxplot, ARCore approach featureless environment - X Coordinates 38
Figure 22: One-sample t-test results, ARCore approach featureless environment 39
Figure 23: Normality test results, acceleration array regular environment 40
Figure 24: Boxplot, acceleration array regular environment - X Coordinates 41
Figure 25: One-sample t-test results, acceleration array regular environment 41
Figure 26: Boxplot, acceleration array featureless environment - X Coordinates 43
Figure 27: One-sample t-test results, acceleration array featureless environment 43
Figure 28: Normality test results, cumulative acceleration regular environment 45
Figure 29: Boxplot, cumulative acceleration regular environment - X Coordinates 46
Figure 30: Wilcoxon test results, cumulative acceleration regular environment 47
Figure 31: Boxplot, cumulative acceleration featureless environment - X Coordinates 48

Figure 32: Friedman test results, regular datasets 49
Figure 33: Post Hoc Tests results, regular datasets 49
Figure 34: Boxplot comparison, regular tests data 50
Figure 35: Friedman test results, featureless datasets 51
Figure 36: Post Hoc Tests results, featureless datasets 51
Figure 37: Boxplot comparison, featureless tests data. 52

List of Tables

Table 1: Ideal set of values, uniform motion of a device. 25
Table 2: Regular tests raw data 57
Table 3: Featureless tests raw data 58

1 Introduction

Augmented Reality is a rising technology that is rapidly expanding and finding many applications in several fields, such as gaming, medicine, education, business, and commerce. The main purpose of AR is to supplement reality by overlaying images, audio, video, and haptic sensations over a real-time view of the concrete world. Augmented Reality represents a variation of Virtual Reality (VR) technologies, which, in contrast to AR, completely immerse a user inside a virtual environment, while restricting his view of the real world. This emerging technology represents a new medium for creative expression through games and interactive applications that fulfill various purposes:

- *Navigation*: apps that facilitate navigation in the real world can highlight the directions a user is taking in order to, for example, clearly indicate the distance to the next turn.
- *Captioning*: AR apps can be used for captioning objects from the real world that are recognised by the device's camera. This is achieved by overlaying interactive objects on the virtual scene that provide specific information about the real object.
- *Commerce and retail*: Augmented Reality makes it possible to use a mobile device to scan a real-world environment such as a living room, and virtually place a product there to see how it would look and fit. By using this kind of applications, shoppers can "furnish" an entire room virtually, trading out styles of the various pieces of furniture and appliances to see how they work before buying.

There are three main characteristics that need to be present in order to have true Augmented Reality: AR combines real and virtual information, AR is interactive in real-time, AR operates in a 3D environment (Kipper and Rampolla, 2012). These are the features that draw the line between what is AR and what is not. An image altered in Photoshop or any other type of 2D overlay cannot be considered AR. Although film and television often feature virtual objects seamlessly blended to a 3D real environment, they cannot be classified as types of Augmented Reality as they do not present any kind of interaction with their users. Instead, a mobile app that allows virtual objects to be overlaid on top of a real-world scene captured by the device's camera, and to interact with them in various ways is a good example of Augmented Reality.

AR applications can be run on any computer – PC or mobile device – that has a monitor or display screen, a camera and tracking and sensing systems (e.g. GPS, accelerometer, gyroscope, etc.). These requirements make smartphones the most common method to access Augmented Reality content. Smartphones and tablets can not only use their cameras to detect environment features and markers, but also contain several sensors that provide an accessible way to track the device motion in 3D space. Other devices like Kiosks, Window Displays, AR Glasses and Head-Mounted Displays can also be used to offer even more enhanced AR experiences. However, the high prices that these products have in the contemporary market drastically reduce their usage, as they make these devices less accessible to the average consumer.

Corporations like Google and Apple have created dedicated development APIs for Augmented Reality, which support programmers in the creation of AR mobile apps for various purposes. ARCore is Google's platform for building Augmented Reality experiences. Using different libraries, ARCore enables every supported device to sense its environment, understand the world and interact with information. As it is stated on the official overview for Google ARCore (2019), all these features are achieved through three key capabilities:

- *Motion tracking*: as the AR-enabled device moves through the world, ARCore uses a process called simultaneous localization and mapping (SLAM) to understand where the phone is relative to the world around it. ARCore detects visually distinct features in the captured camera image called feature points and uses these points to compute its change in location. The visual information is combined with inertial measurements from the device's IMU (Inertial Measurement Units) to estimate the pose (position and orientation) of the camera relative to the world over time. By aligning the pose of the virtual camera that renders the 3D content with the pose of the device's camera provided by ARCore, it is possible to render virtual content from the correct perspective. The rendered virtual image can be overlaid on top of the image obtained from the device's camera, making it appear as if the virtual content is part of the real world.
- *Environmental understanding*: ARCore constantly improves its understanding of the real-world environment by detecting feature points and planes. ARCore looks

for clusters of feature points that appear to lie on common horizontal or vertical surfaces, like tables or walls, and makes these surfaces available to the AR app as planes. This information can be used to place virtual objects resting on flat surfaces. Because ARCore uses feature points to detect planes, flat surfaces without texture, such as white walls, may not be detected properly.

- *Light estimation*: ARCore can detect information about the lighting of its environment and provide the average intensity and colour correction of a given camera image. This information allows to light the virtual objects in the scene under the same conditions as the environment around them, increasing the sense of realism.

ARCore offers several additional features that help create an immersive AR experience. One worth mentioning is the depth understanding feature, which allows for virtual objects to be correctly placed even in complex 3D environments. By creating depth maps – images containing data about the distance between surfaces from a given origin – ARCore enables immersive and realistic experiences, such as making virtual objects accurately collide with previously detected surfaces or making them appear in front or behind real-world objects.

There are two types of simple Augmented Reality: marker-based and markerless.
Marker-based AR uses different types of visual cues that can be detected by a camera and used by the software to position virtual assets on the scene. The simplest kind of marker is represented by a black and white image that can consist of one or more basic shapes. A wide range of more complex images and QR codes can also be used provided that the camera is able to read them properly. The result achieved by using marker-based AR is a live render of the real scene with digital assets placed at the location of the markers (Katiyar, Kalra and Garg, 2015). Although marker-based techniques can provide really stable and immersive AR experiences, they present several limitations that are related to the need for markers to be placed in the real-world environment at all times. Marker-based solutions are generally suitable for applications that do not require the user to navigate in wide areas, but rather to constantly point the device to a static object that contains the marker, such as a table, a floor or a wall. AR applications that enhance the experience of tabletop games are a good example of systems that rely on marker-based

techniques to overlay virtual content over the real-world marker, which is usually represented by the game board.

In contrast to marker-based AR, Markerless AR applications do not need any prior knowledge of the real-time environment to overlay virtual content onto a scene and hold it in a fixed point in space. Instead, a combination of camera systems, dedicated sensors, and complex math is used to accurately detect and map the real-world environment. By having a complete scheme of the locations of the walls and the different points of intersection, a markerless AR application makes it possible to place virtual objects into a real context and hold their positions fixed without the need for an QR code or image (Schechter, 2019). These characteristics make markerless applications suitable even for scenarios in which the users can move through large areas and open spaces. Markerless Augmented Reality has seen its biggest impact in gaming, with the AR-enabled Pokémon GO becoming a smash hit back in 2016. There has been a steady stream of AR titles since then, though none of them has made an impact just like Pokémon GO yet. These games are still just scratching the surface of what AR can do, and ARKit/ARCore make it possible to create multi-user real-world games with 3D content that remains fixed in place for users to find and interact with.

Although the markerless technique provides excellent results in most cases, there are some scenarios in which the related applications seem to fail to accurately fix the virtual objects positions and track the user's movements. Plain walls and featureless objects in general do not provide the software enough data to build an exact map of the environment and could therefore cause the application to fail. This is the issue that this research tries to solve by adopting an innovative approach to AR (see Chapter 3).

In an attempt to help address the inaccuracy problems of markerless applications under unfavourable conditions, this thesis presents a prototype of an Augmented Reality mobile application that implements aspects of both marker-based and markerless AR. The application uses a single marker as a synchroniser, along with the dimensions of the real-world environment. Motion tracking is implemented through the use of the device embedded sensors and does not rely at all on the camera's ability to detect features in the real-world environment. Through this study, the question of whether it is possible to virtually map any kind of space, knowing its general layout and having a synchronisation point, will be answered.

2 Literature Review

Augmented Reality is the enhanced or resultant image produced by overlaying another image over a real-time view of one's surroundings (Peddie, 2017). The modern concept of Augmented Reality is the result of several studies that throughout the years attempted to develop technologies that could enhance real-world experiences. The following section outlines some of the most significant historical events that contributed to the making of what today we call AR.

2.1 Historical Overview

The origins of Augmented Reality date back to the beginning of the second half of the twentieth century, when a group of researchers working for the Philco Corporation began developing a television surveillance system with a head-mounted display. This is one of the first appearances in history of the technology like we know it nowadays. The originality of this idea led other companies and researchers to develop similar products whose purpose was to enhance the feeling of the real-world.

In early 1950s, the teleprompter (Figure 1), also known as autocue, was invented, and brought Augmented Reality to TV. The teleprompter is a display device, usually placed below the lens of a video camera, that prompts the speaker with a visual text of their script. The text appears right in front of the camera lens and this allows to create the illusion that the speaker is looking directly at the spectators. This device has many similarities with the current AR technologies, as it essentially overlays virtual content (the text) upon real-world objects (the camera).

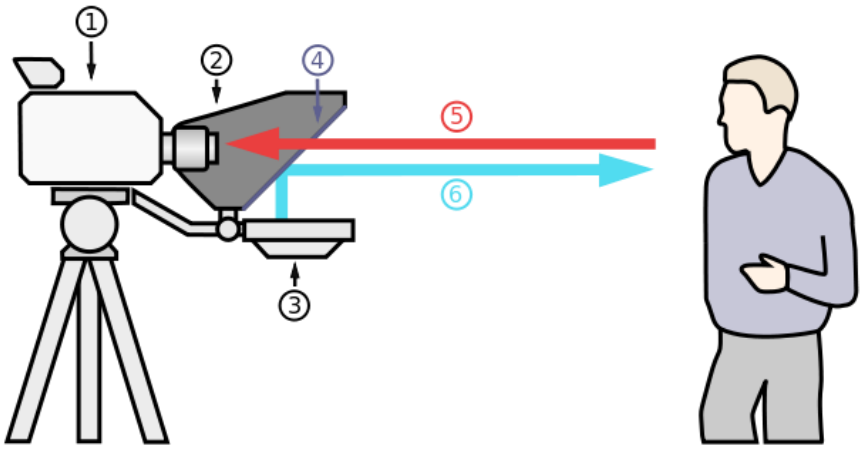

Figure 1: A teleprompter system. The image shows a teleprompter system with: (1) Video camera; (2) Shroud; (3) Video monitors; (4) Clear glass or beam-splitter; (5) Image from subject; (6) Image from video monitor (Wikipedia, 2018).

In 1963, the Bell Helicopter company, inspired by the work of the Philco Corporation, designed a servo-controlled camera paired with a remote viewing headset. The system provided the pilots with an augmented view of the ground, which was captured by the infrared camera placed under the helicopter. The camera was slaved to the headset and would therefore move along with the pilot's head. The idea was furthermore expanded by Ivan Sutherland, who in 1968 created the first Augmented Reality (and Virtual Reality) system. The so-called Sword of Damocles used an optical see-through head-mounted display which was equipped with six-degrees-of-freedom (6DOF) trackers. This was one of the earliest examples of devices that allowed tracking of both rotations and linear movements on all three axes.

The first interactive Augmented Reality system dates back to 1975, when Myron Krueger developed the Videoplace technology (Figure 2). The device surrounded the users and was able to respond to their movements and actions without the use of gloves or headsets. With this system, the users were able to interact with virtual objects for the first time.

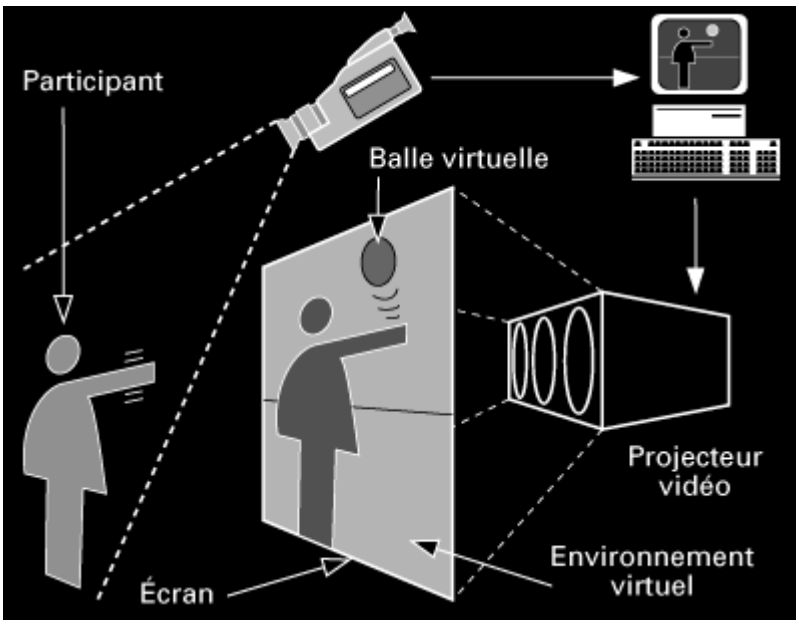

Figure 2: Videoplace by Myron Krueger. A visual scheme of the various components of the Videoplace system created by Myron Krueger. This is the first device in history allowing interaction with virtual content without the use of any specific equipment for the user (Krueger, Gionfriddo and Hinrichsen, 1985).

Throughout the years, these systems were improved and mainly employed in the field of aeronautics. The main goal was to provide highly reliable head-mounted displays for pilots that would help them while flying under unfavourable conditions. It was only in 1990 that the term "Augmented Reality" was finally coined by two Boeing researchers,

Thomas P. Caudell, and David Mizell. This came about from their work centred on simplifying the manufacturing and engineering process for aircrafts assembly. The two researchers designed a software that could overlay virtual content on top of the real world. The information displayed by the software helped the construction workers by highlighting the process of how certain cables in the building procedure were supposed to be connected.

In 1996, Jun Rekimoto promotes the idea of using 2D markers as the physical objects or places where the real and virtual environments fuse together. His AR prototype called NaviCam used markers to automatically identify where the digital information was to be presented. This prototype is one of the first marker-based systems that allowed camera tracking with 6DOF. The markers presented by Rekimoto are still in use today and share the same purposes with the ones that are employed in today's Augmented Reality.

In late 1990s, Augmented Reality finally became a distinct field of research, and several AR conferences and workshops were started. The official definition of the three key elements that make Augmented Reality is attributed to Ronald Azuma, who states that AR:

- Connects real and virtual worlds
- Is interactive in real time
- Allows movement in three dimensions

Figure 3: ARQuake by Bruce Thomas et al. The above images respectively show the players perspective through the AR headset and the equipment that is required to play the game (Wearable Computer Lab, no date).

Augmented Reality was brought to the field of games in 2000, when Bruce Thomas et al. create an AR version of the famous game Quake (Figure 3). A head-mounted display with six-degrees-of-freedom trackers, GPS, a digital compass, and a marker tracking system was used to give a first-person augmented view of the game and to provide inputs to control the game. Four years later, the first Augmented Reality system on a mobile phone was accomplished by Oliver Bimber, Christian Lessig, and Mathias Möhring. The researchers presented an AR system that was able to detect and differentiate various markers through the mobile's camera and to correctly integrate pre-rendered 3D graphics into the video stream in real time.

In 2009, Qualcomm recognises the high potential and synergy that AR and mobile devices have and launches project Vuforia, a Software Development Kit (SDK) for mobile devices that allows the creation of Augmented Reality apps. In the last ten years, AR has found its way through different fields and is nowadays employed in several industries. The next section discusses the various types of Augmented Reality that are commonly used today, by also pinpointing the sectors in which it is frequently employed.

2.2 Types of Augmented Reality

Kipper and Rampolla (2012, p. 36) provide a taxonomy of Augmented Reality that is based on recognition methods. Recognition can be defined as the process by which the hardware and the software determine where and how to augment reality. Different devices use different strategies to achieve recognition, but they all implement one of the following four methods:

- Pattern
- Outline
- Location
- Surface

The Pattern Augmented Reality system performs simple pattern recognition on a marker, which is most of the times represented by a basic shape. When this is successfully recognised, the system creates an area in which digital elements can be placed. These elements can be anything from 3D models, audio clips, videos, or any other piece of digital information. This method is generally employed when using devices that, through the use of a webcam, make the user part of the interactive video feed.

The Outline method is the process by which a part of the user's body is recognised and then blended seamlessly with any piece of digital information. This AR strategy enables the user to interact with 3D objects by using natural movements, such as picking up a virtual item with a real hand. The system tracks the outline of the recognised part of the body and adjusts the pose of the virtual content accordingly. Since this is done in real time, the software is also able to respond to movements of the user's body by correctly redrawing the virtual objects in their new positions. This type of AR is very common in most of the popular social-media mobile apps.

The Location method is based on detailed GPS location information. By using triangulation, the GPS provides the global coordinates at which the device is located and enables the AR system to use the camera to precisely overlay virtual objects, such as icons and text, over buildings and notorious places all around the real world. Location AR is often used through a mobile device. Modern smartphones have everything that is needed to enable a location-based AR application: a camera, a display, GPS capabilities and several sensors. One common implementation of this type of Augmented Reality is represented by AR browsers. Much like internet browsers, AR browsers allow their users to find information about the world around them. These systems are designed to provide the users with details about anything they point the device's camera at. AR browsers can also merge with the Pattern and Outline methods by allowing to identify objects such as QR codes or advertisement signs. For instance, an AR browser can recognise an image from an advertisement and provide information about the related product or point the user in the direction of the nearest store where it can be found.

Surface Augmented Reality is accomplished by using screens, floors or walls that respond to the touch of people and provide them with virtual information in real time. These systems combine surface computing with AR in order to create environments where any surface can be fully interactive. The Kinect is a good example of a device that implements Surface Augmented Reality. This motion sensing input device designed for the Xbox game console allows for interaction through movements and voice commands rather than the traditional input buttons. The Kinect can be classified as an AR-type platform that allows any surface to be used as a computing interface. Its depth camera technology

enables the embedded cameras to detect the ranges of physical objects and therefore to understand the ongoing interaction between the user and the surface.

AR applications can also be classified by the content area that they address. Augmented Reality is nowadays employed in various industries; therefore, it is possible to create a list of categories of AR apps.

- Education
- Science
- Business and Manufacturing
- Medicine
- Public safety and military
- Art
- Advertising
- Games and Entertainment

This does not represent a comprehensive list of all the applications of AR, but rather this list presents broad categories that cover many of the currently available apps (Craig, 2013).

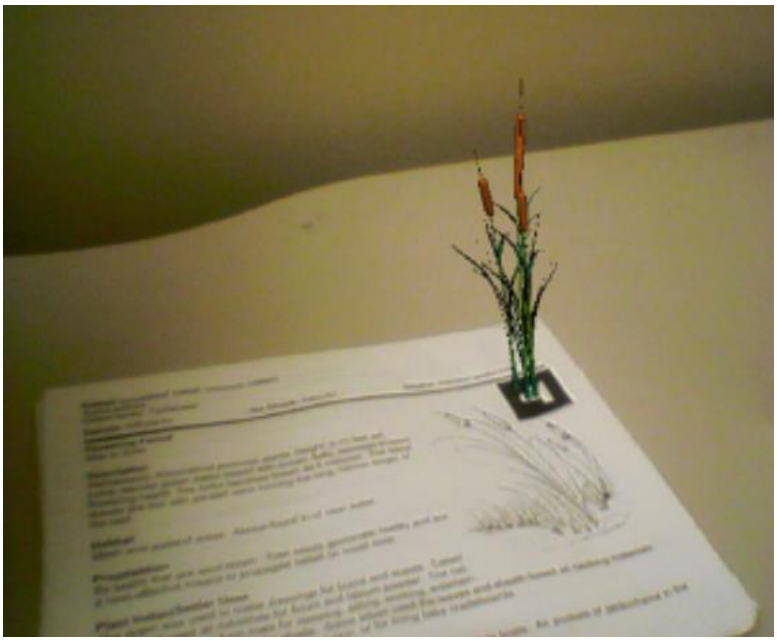

Figure 4: A page of the Ethnobotany Workbook by McGrath et al. When the page is viewed through an AR device, the user sees the page enhanced with a 3D graphical representation of the plant (Craig, 2013, p. 250, fig. 8.23)

In the field of education, magic books represent one of the most commonly used products. These books are just regular children's books that are animated when seen through an

AR-enabled device, such as a smartphone or a head-mounted display. One interesting aspect about these books is that they are totally standalone in physical space, which means that the book is useful even on its own. However, it also offers the opportunity to enhance the reading experience through Augmented Reality. This product is not only employed in the field of children's education, but it also finds usage in both science and medicine sectors. The Ethnobotany Workbook by McGrath et al. (Figure 4) is a notorious example of the magic book class of AR applications. The book was designed as a sort of field guide to help the middle school reader identify different species of native plants. Several plant models were created to enhance the reading experience through AR. By using a set of markers associated with the various species of plants, the users could visualise the related 3D models. Other applications that use similar ideas to the ones of the presented workbook are nowadays used to teach anatomy and other medicine related subjects. Anatomy magic books are especially useful as they allow their users to directly interact with the different physical systems of the human body and provide a clearer understanding of how the different parts are interconnected. This is just an example of how Augmented Reality can offer experiences that not anybody could try otherwise.

Augmented Reality applications are often used in the field of manufacturing to provide interactable suggestions and instructions aimed to aiding the building processes. A similar purpose is shared by the applications employed in public safety, where AR devices are generally paired with surveillance systems to enable experiences that are more interactive (e.g. camera systems that can rotate along with AR-enabled head-mounted displays). In the fields of art and advertising AR apps are generally used to overlay additional digital information about the real-world objects that can be scanned by the AR-enabled device's camera. Augmented Reality is rapidly finding its spot even in the field of games and entertainment, with applications such as Pokémon GO which caught the interest of millions of people all around the world right after its release.

Retail product visualisation is one of the areas in which Augmented Reality is able to reliably provide very useful experiences. Applications such as IKEA Place (Figure 5) allow shoppers to furnish entire rooms virtually by trading out styles of cabinets, flooring, and appliances in order to see how they look and fit in their homes' environment. This gives freedom to the users as it enables them to shop in their own space. The trend of usage of this category of Augmented Reality is continuously accelerating with each new

generation of both hardware and software. Commerce and retail apps share similar implementations as they all use markerless techniques to virtually map real-world environments. This is a relevant topic to this research as it introduces the problem of virtually reconstructing a physical environment by just using a camera and a set of sensors to understand the space in which the device is immersed. The following sections present some of the main studies that have been carried out on the field and provide more information on the techniques that were designed by the various researchers throughout the years.

Figure 5: IKEA Place app. A showcase example of the IKEA retail app which allows to virtually place furniture inside the user's house through markerless Augmented Reality (Schechter, 2019).

2.3 Virtual 3D Reconstruction: Problems and Solutions

The long-standing problem of virtually reconstructing a three-dimensional real-world scene has been analysed by several experts in the fields of computer vision and computer graphics. Most of the research that has been carried out in this subject area make use of two key algorithms: Simultaneous Localization and Mapping (SLAM) and Iterative Closest Point (ICP). Both these algorithms find their origins in the field of mobile robot navigation and are nowadays widely used in both Virtual Reality and Augmented Reality systems.

The SLAM problem can be defined as follows: a mobile robot roams an unknown environment, starting at an initial location. Its motion is uncertain, making it gradually

more difficult to determine its current pose in global coordinates. As it roams, the robot can sense its environment with a noisy sensor. The SLAM problem is the problem of simultaneously building a spatial map of an environment while determining the robot's position relative to this map (Stachniss, Leonard and Thrun, 2016). This is one of the biggest challenges of robot automation. Indeed, it is fairly easy to build a spatial map of an environment provided that the relative position of the robot is known and, vice versa, to extrapolate the location of the robot from a spatial map of the environment. However, SLAM challenges the impossible by gathering both information at the same time. Three main paradigms were designed through the years to solve this problem. The first, known as Extended Kalman Filter (EXF) SLAM, is in robotics the earliest but has become less popular due to its performance limitations. The second approach uses non-parametric statistical filtering techniques known as particle filters. The third paradigm is based on graphical representations and applies nonlinear sparse optimisations to solve the SLAM problem. All these techniques rely on Monte Carlo methods to predict the robot's pose at any given time in the most accurate way possible. These paradigms are effectively re-used by the new incremental implementations of SLAM that are nowadays used in commercialised systems such as Google's ARCore. The same ideas are replicated in modern self-driving cars, where the SLAM problem is simplified by making extensive use of highly detailed map data collected in advance. Lidar SLAM (Figure 6) is a notorious example of these techniques, which employs laser lights to evaluate distances by measuring the reflections through specific sensors.

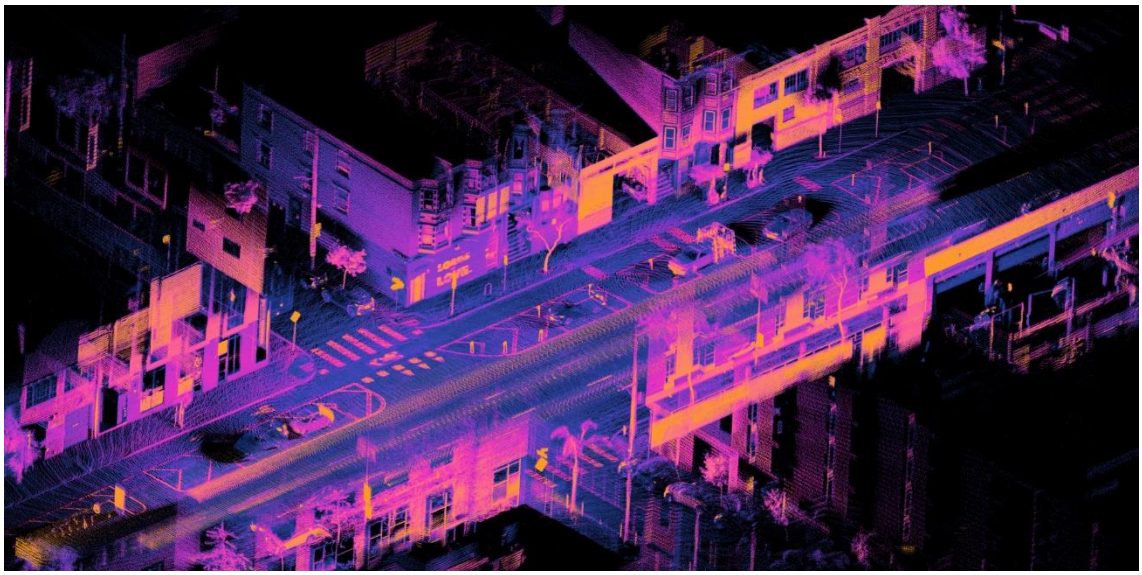

Figure 6: Accumulated registered point cloud from lidar SLAM. The image shows an orthographic projection of a registered point cloud captured over 18 seconds using a lidar system mounted on a moving car. The points are registered in real time as the vehicle's trajectory is estimated using a simultaneous localization and mapping algorithm (Lu, 2019).

The Iterative Closest Point (ICP) algorithm is a widely used solution for geometric alignment of three-dimensional models when an initial estimate of the relative pose is known. The algorithm starts with two basic meshes of the models and an initial guess of their relative position and rotation. The models' transform is iteratively refined by repeatedly generating pairs of corresponding points on the two meshes while minimising a specific error metric (Rusinkiewicz and Levoy, 2001). In the field of Augmented Reality, the ICP algorithm is generally used to refine the data gathered through the statistical SLAM solutions. Indeed, AR typically implements SLAM to create an approximate map of the real-world environment and an estimate of the device position, and then refines this information by applying the ICP algorithm whenever new data arrives from the camera and from the tracking and sensing systems.

2.4 Virtual 3D Reconstruction: Case Studies

Almost every study in the above-mentioned area of AR has a reference point in the research of Newcombe et al. (2011) which goes under the name of KinectFusion. The technique developed by the researchers of KinectFusion enables a user holding and moving a standard Kinect camera to create detailed 3D reconstructions of an indoor scene. The depth data collected by the Kinect camera is used to track the 3D pose of the sensor and to reconstruct geometrically precise 3D models of the real-world environment in real-time (Figure 7). KinectFusion uses SLAM and ICP to create a surface-based representation of the physical scene. The system continuously tracks the six-degrees-of-freedom pose of the Kinect camera and gathers new viewpoints of the scene that are used to refine a global virtual representation of the environment.

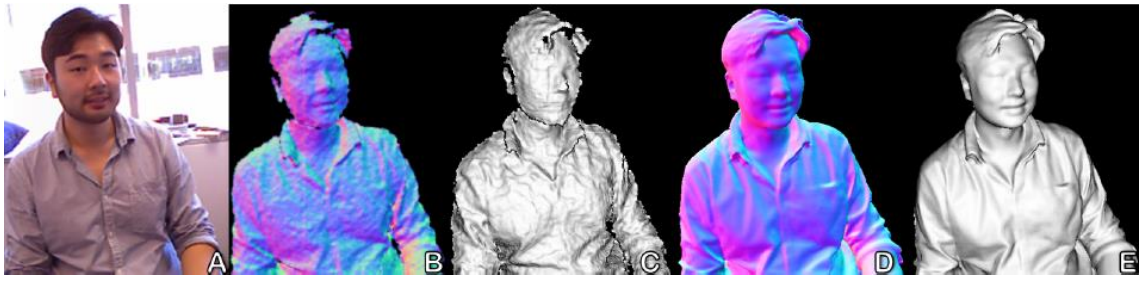

Figure 7: A) RGB image of scene. B) Normals extracted from raw Kinect depth map. C) 3D Mesh created from a single depth map. D and E) 3D model generated from KinectFusion showing surface normals (D) and rendered with Phong shading (E) (Newcombe et al., 2011, p. 560, fig. 2).

Beyond scanning, KinectFusion enables more realistic forms of AR, where 3D virtual objects are overlaid onto and interact with the real-world representation. These objects can be rendered from the same perspective as the tracked physical camera, thus giving

the illusion of them being part of the real-world environment. The live 3D models also allow composited virtual graphics to be accurately occluded by the real world. Although this produces visually pleasing reflections on glossy virtual objects, the quality of the occlusions is quite low around the edges of objects due to significant noise along depth discontinuities.

Several successive studies based on the KinectFusion system have been carried out in order to apply similar techniques on different mobile devices. As an example, Chen et al. (2017) developed an Augmented Reality framework based on the SLAM technique that improves the method used in KinectFusion and can achieve real-time 3D environment reconstruction by using any mobile device that is equipped with a depth camera and Inertial Measurement Units (IMUs). This technique laid the foundations for today's mobile AR, which sees smartphones as the most common devices to experience Augmented Reality. Although both KinectFusion and the above technique can generate high-fidelity three-dimensional models of the real-world objects, their core systems make a very strict fundamental assumption – that camera tracking is always performed on a static scene. This is very uncommon in practice, as generally physical objects such as the users' hands inevitably appear in the scene, move dynamically and therefore impact tracking and reconstruction (Figure 8).

Figure 8: KinectFusion 3D reconstruction of a moving body. A user moves freely in front of a fixed Kinect. Live raw data (top) and shaded reconstruction (bottom). Left: Scene without user. Middle: User enters scene but is only partially reconstructed due to motion. Right: Continued scene motions cause tracking failure (Newcombe et al., 2011, p. 562, fig. 8).

To overcome this limitation, Lu et al. (2018) designed a robust optimisation scheme that segments out dynamic objects and static environment and implements a 6DOF pose prediction algorithm to reduce the number of iterations in the ICP algorithm (Figure 9). Their approach handles two essential technical issues. First, the system needs to be able to detect dynamic objects and to distinguish them from the static environment under the moving camera. Second, the software should support fast and rapid motions during reconstruction, which is something hard to tackle by using conventional solutions.

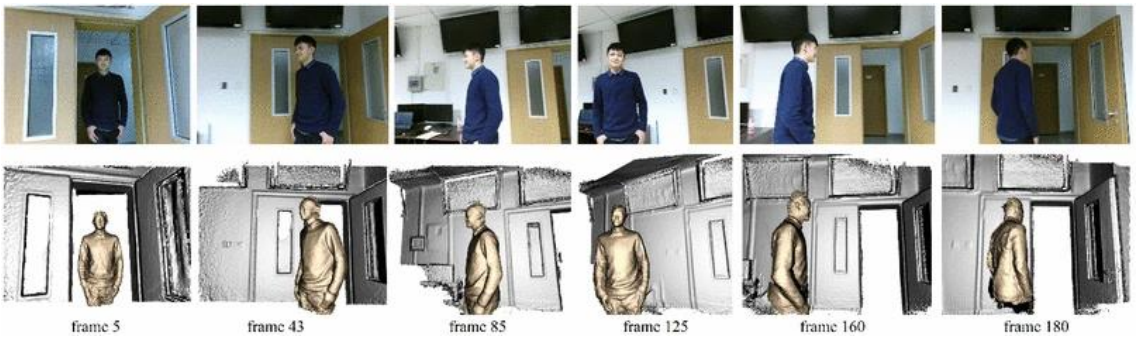

Figure 9: Real-time reconstructions of a dynamic scene. Both the person and camera are moving while the background is still. The method only requires single depth camera to capture the live depth map and generate plausible results in real time. Gold colour denotes moving object and silver colour indicates static background (Lu et al., 2018, p. 2, fig. 1).

To solve these issues, a sequential 6DOF pose prediction algorithm is implemented to avoid ICP failures under large motion. Furthermore, the application implements all relative techniques (segmentation, tracking, volumetric fusion, rendering) and presents a dense simultaneous localization and mapping (SLAM) system capable of reconstructing dynamic objects and static environment in real time.

All the cited techniques rely on feature detection by the device camera and, therefore, cannot operate well in environments that lack of feature points. Plain walls, floors, or ceilings, along with featureless objects might reduce the system's accuracy. An inexact reconstruction of the real-world scene causes the virtual objects not to be held in a fixed position on the scene, and this consequently breaks the user's immersion and lessens the AR experience. The following chapters describe and analyse the hybrid solution attempted throughout this research, which tries to overcome the limitations of traditional markerless systems. The hybrid approach is presented as a reliable alternative to markerless AR, which is expected to provide satisfactory results when employed in both featureless and regular environments.

3 Methodology

3.1 Tools and General Approach

The goal of this study is to assess whether the proposed hybrid technique that merges aspects of both markerless and marker-based AR can produce a reliable system for three-dimensional reconstruction of real-world environments. The hypothesis under test is that an Augmented Reality system relying on a single marker as a synchroniser and the dimensions of the physical space can allow to build an accurate virtual map of the chosen environment and therefore enable stable positioning of various virtual objects. An Augmented Reality mobile app has been developed to evaluate whether the technique provides the expected results. The application does not rely on the device camera's ability to capture features in the objects, as it tracks the user's movements by just using the device's sensors (accelerometer, gyroscope, etc.). The synchronisation marker represents the starting point for every calculation, and all the tracking is done relatively to this marker.

The application was developed in Unity by using the ARCore SDK – Google's Software Development Kit for mobile Augmented Reality – to implement the key functionalities of the AR system. Android studio was also used to carry out several tests on the Android sensors and on the integration of the values given by them. The Unity engine was chosen as the main development platform as it allows to easily manage all the graphics elements in a virtual scene and therefore to keep the main focus of the research on the hybrid AR technique. The ARCore SDK was used as a starting point to build the Augmented Reality app as this is the most commonly used development kit for Android devices.

Two versions of the app have been built in order to test two different approaches to the problem. Both these systems use ARCore libraries in order to initialise the Augmented Reality scene and Unity API functions to manage graphics objects and to read data coming in from the sensors. The general flow of these applications involves scanning the synchronisation marker, initialising the device position in the virtual scene, and initiating the tracking of the device through the sensors. The project's applications were built using C#, as this is Unity's native programming language.

3.2 The ARCore approach

In order to compare the attempted solutions with the tracking done by ARCore, a simple test app has been implemented using the camera tracking systems offered by Google's SDK. This application makes use of all the functionalities offered by ARCore, including the camera feature detection system. As stated above, the application starts by scanning a marker and then proceeds with the initialisation of the virtual camera and of the motion tracking system.

Marker recognition was achieved by using Augmented Images in ARCore. This feature set allows to build AR apps that can respond to 2D images in the user's environment, such as posters or product packaging. When provided with a database that contains a set of reference images, ARCore is able to detect where those images are physically located in the real world. When an image is being tracked, estimates for its position, orientation and physical size are automatically evaluated. These estimates are continuously refined as ARCore gathers more data (Figure 10). The image's position and orientation keep being tracked even when the image temporarily moves out of the camera view. When the project's application is started, a message suggesting looking for a marker is prompted, and when the recognition is successful, the tracking of the device begins. This approach is common to all the attempted solutions in this research.

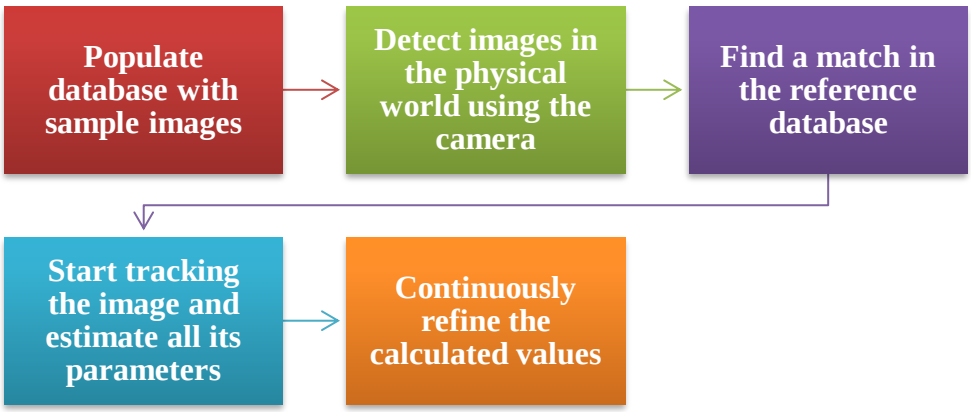

Figure 10: ARCore's Augmented Images Flowchart. A schematic description of the general flow that any ARCore app employs to detect images and markers in the real world.

In this approach the device position in the virtual scene is calculated by simply using ARCore functionalities for camera tracking, which are included in the Frame class. This class holds information about ARCore's state including tracking status, the pose of the camera relative to the world, estimated lighting parameters, and information on updates

to objects (like Planes or Point Clouds) that ARCore is tracking. When the marker is successfully recognised, the real distance between the device and the marker is evaluated through the use of Unity API functions and the Augmented Images feature set. Assuming that the horizontal and vertical displacements of the device relatively to the marker are approximately zero, the initial position value for the device is set to zero on the x and y axes, and to the previously calculated distance on the z-axis. During every frame update, the virtual camera's pose is updated with the values given by the Frame class and the device's position relative to the marker is evaluated by considering the initial offset. As the user moves in the real-world environment, the pose of the virtual camera changes, the Frame class is automatically updated, and the relative position is adjusted accordingly on all three axes.

With this approach, the application behaviour is overall correct, and the camera tracking is very accurate. ARCore achieves these results by merging the data collected by the Android device's sensors with the camera feature detection system, which allows to compensate for the high inaccuracy of the sensors. When used in environments that present many features, the application provides very reliable results, with displacement values that are very close to the real-world distances. However, when the application runs in featureless environments, after the marker moves out of the camera view, the tracking stops as it cannot detect any more feature points. This is the limitation that this research aims to overcome. The next two approaches present the two attempts that have been carried out in order to solve the above-mentioned problem. Both the solutions solely rely on the Android device's sensors data to track the user's movement and do not use any of the camera feature detection system.

3.3 First approach: acceleration array

The first attempt to overcome the weaknesses of markerless AR uses an array of acceleration values coming from the sensors to evaluate the device's displacement at every frame update through a two steps integration. The flow of the app is the same as the one presented in the previous section. Marker recognition is achieved by using the functionalities that have been previously described. The decision of using sensors to track the user's movements derives from the analysis of previous studies that have been carried out in this field, especially those using sensor fusion to achieve even better motion tracking systems.

The application uses the accelerometer as the main sensor to get the raw acceleration data, which is then refined through a high-pass filter. Once recognition is complete, the system evaluates the distance from the marker in order to set the virtual camera's starting position. This is done in a similar way as the one described previously. After this standard setup, the device tracking starts. The acceleration values from the accelerometer are fetched once every frame update and used to fill the above-mentioned array. This data structure has a limited number of entries and works as a FIFO (first in, first out) queue, which means that the first item to be removed when the structure is full, will be the item corresponding to the head of the array. The queue is used to evaluate the displacement of the device through a two steps integration that also takes acceleration values from previous frames into consideration. This whole process allows to translate the sensors data into movements of the device in the virtual scene.

The data coming in from the sensors needs to be refined before it can be used to calculate velocity and displacement. In order to do this, a high-pass filter is applied to the acceleration values that are fetched during each frame. This filter allows to obtain measurements of the real acceleration of the device by removing the contribution of the force of gravity. The application samples the values of acceleration and gravity acting on the device on the three axes (x, y, and z) by using the Input class in Unity. Since these values are already divided into separate coordinates, a simple subtraction between the various components allows to achieve the desired result. The refined acceleration values are then stored in the acceleration array.

Since real integration is a computationally heavy task, a simpler method is required. The adopted approach assumes that the acceleration values are constant within the same frame update and uses the uniformly accelerated motion formulae to evaluate velocity and displacement. Obviously, this introduces errors in the calculations as the method represents an approximation of real integration (Equation 1). Therefore, the final results are not expected to be extremely accurate. If acceleration is constant throughout the frame update, velocity and displacement can be easily evaluated by using the time interval that goes between two consecutive frame updates. In this case, this is represented by the delta time value provided by Unity's Time class.

$$\Delta x = \int_0^t v_x(t')dt' = \int_0^t \left[\int_0^{t'} a_x(t'')dt'' \right] dt'$$

Equation 1: Three-way relationship between acceleration, velocity, and displacement. Acceleration can be translated into velocity and displacement through integral calculus. Displacement's first derivative is indeed velocity, and velocity's first derivative (which coincides with displacement's second derivative) is acceleration.

All the calculations are done using the acceleration array that is filled through the filtering function. Velocity and displacement are obtained by summing the infinitesimal contributes from the previous frames' acceleration values. The size of the array determines how far in the frame history the application looks during each update. For every iteration of this calculation, the average of two consecutive acceleration values is multiplied by the delta time, and the result is added to the total velocity. Within the same iteration, displacement is also increased by a value that is equal to the result of the multiplication between the previously evaluated velocity and the delta time.
Due to the high inaccuracy of the sensors and to the heavy approximation that is introduced by integrating the acceleration values, this version of the application cannot perform as intended. The displacement values generated when the device moves in space, for the most part, do not resemble the real movement of the user. Moreover, the change in position is sometimes slow due to the importance that is given to the old frame updates when performing the integration.

In an attempt to solve this last issue, an alternative version of the app has been built. The same approach is implemented through a system that collects the acceleration samples multiple times per frame. By doing this, the acceleration array is filled in with the same amount of values that are however fetched during the same frame update, with a fixed time interval. Both the refinement of the acceleration raw values and the integration are performed through the same sets of instructions. Obviously, in this case, the delta time variable is swapped with the above-mentioned fetching time interval. Although this approach allows to reduce the delay in the displacement updates, the results are still inaccurate and tracking still does not behave correctly.

The main reason why tracking is not reliable in both of these approaches is that both positive and negative values of acceleration are treated the same way during motion. When moving a device in a straight line the acceleration values registered by the

accelerometer oscillate between positive and negative values. This is due to a peculiar property of acceleration that can be explained through a simple example. A device resting on its back on a table, with the bottom facing the user is moved to the right for exactly one meter. If the acceleration is perfectly distributed throughout the whole motion, the values registered by the accelerometer will be as follows (Table 1).

Table 1: Ideal set of values given by the uniform motion of a device.

Distance	Acceleration	Velocity (%)	State
0.0 m	0.0 m/s^2	0 % (Min)	Resting
0.0834 m	0.5 m/s^2	16.67 %	Accelerating
0.1668 m	1.0 m/s^2	33.33 %	Accelerating
0.25 m	1.5 m/s^2	50 %	Accelerating
0.3334 m	1.0 m/s^2	66.67 %	Accelerating
0.4168 m	0.5 m/s^2	83.34 %	Accelerating
0.5 m	0.0 m/s^2	100% (Max)	Accelerating
0.5834 m	-0.5 m/s^2	83.34 %	Decelerating
0.6668 m	-1.0 m/s^2	66.67 %	Decelerating
0.75 m	-1.5 m/s^2	50 %	Decelerating
0.8334 m	-1.0 m/s^2	33.33 %	Decelerating
0.9168 m	-0.5 m/s^2	16.67 %	Decelerating
1.0 m	0.0 m/s^2	0 % (Min)	Resting

If this scenario is replicated in the current implementation of the application, the negative values of acceleration would be considered to be movements to the left and therefore the final displacement of the device would be zero instead of one meter. The second approach that was explored during the research focuses on a different solution that attempts to avoid this issue by using a cumulative value of acceleration instead of various independent samples stored in an array.

3.4 Second approach: cumulative acceleration

The second attempt to overcome the weaknesses of markerless AR uses a single cumulative acceleration value instead of an array of samples. The app evaluates the device's displacement at every frame update through a two steps integration. The flow of

the app is the same as the one presented in the previous sections. Marker recognition is achieved by using the functionalities that have been previously described.

This version of the application uses the gyroscope as its main sensor to get the user's acceleration data. The relative Unity API wrapper class (Gyroscope) is used to initialise the device's sensor and to fix its update interval. Since the app is limited to 30 frames per second, this value corresponds to a 30Hz frequency. The Gyroscope class in Unity allows to obtain the filtered values for the user's acceleration by automatically removing any contribution coming from the force of gravity. This simplifies the code and eases the whole process of fetching the data and performing the two steps integration. The Unity function InvokeRepeating is used to gather multiple samples of linear acceleration from the gyroscope within the same frame. This function allows to invoke any custom method repeatedly every certain number of seconds, independently from the Unity frame update loop. The gyroscope refresh rate and the time interval that goes between each successive execution of the sampling method are exactly the same. The linear acceleration sample is always added to the cumulative acceleration and its value is used immediately after being fetched. Before integrating the acceleration data, a simplified Kalman filter (Equation 2) is applied to the raw linear acceleration coming from the gyroscope. This allows to reduce the noise in the values given by the sensor.

$$a = raw \times \varphi + a \times (1 - \varphi)$$

Equation 2: Simplified version of the Kalman filter. Refines the acceleration values read through the sensors by using a specific filtering factor to conveniently weight the two addends in the equation, the raw acceleration, and the previously refined acceleration values. This allows to reduce the impact of noisy samples.

In the above formula, *a* represents the refined acceleration, whose value depends on both the raw acceleration (*raw*) and the previous value of the refined acceleration itself. The filtering factor (φ) is a constant value between 0 and 1 that weights the two addends in the equation. A low value of the filtering factor is more appropriate in this case as it gives more importance to the old values, and therefore allows to cut off noisy samples. The output of the filter is subtracted from the acceleration raw value and the result is added to the cumulative acceleration. Velocity and displacement are then calculated by using the same formulae that were explained in the previous section. In order to have more precise results, timestamps are used to calculate the time interval that needs to be used while performing the integration. A timestamp is saved every time the sampling function is

called and the difference between the current time and the last saved timestamp is used as the delta time in the equation.

By accumulating the acceleration values, the issue described in the previous section does not occur anymore. The example scenario presented in the first approach provides better results when replicated in this version of the app. After the device has moved one meter to the right, the acceleration value correctly goes back to 0 and the final value for the virtual displacement of the device is closer to the real-world displacement.

Even with this approach, the process of integrating the acceleration values coming from the gyroscope is not accurate enough to provide satisfactory results. The calculations that are required to evaluate velocity and displacement are approximations of the real values and therefore a realistic movement of the virtual device cannot be achieved. Although the Kalman filter helps to reduce the noisiness of the gathered acceleration samples, the device keeps perceiving random acceleration inputs even when it is standing still, and as a result the integration output is altered.

Several tests have been carried out on all the presented solutions in order to assess whether the hybrid approach can represent a reliable alternative to traditional Augmented Reality. A comprehensive list of all the obtained results along with a discussion on the pros and cons that each approach provides is presented in the following section. The collected data sets have been compared using statistical tests in order to highlight how the various strategies differ from each other.

4 Evaluation and Discussion

4.1 Testing methodology

Three main sets of tests have been carried out to evaluate the noisiness of the sensors and the performance of the various versions of the app in both regular and featureless environments. For each category, all the tests have been done under the same conditions and by following a precise set of rules that determined the motion of the device during the data collection. Due to the lack of specific tools and proper equipment for testing, every dataset was built by manually testing the app with a standard device in common environment settings. By having access to tracks for regulating the device's movements and wider featureless spaces, the experiments would have suffered less from the additional noise introduced by hand motion. In order to mitigate this problem, several repetitions of the same experiment were done throughout the various tests, so that the collected datasets would statistically have less inaccurate samples.

The noisiness of the sensors was evaluated by using a simple app built in Android Studio that allowed to register the pure values of acceleration without any sort of prior refinement. The data samples were collected and stored in CSV (Comma Separated Values) files for further analysis in SPSS version 26 (IBM, 2019). For this set of tests, the device was fixed on top of a flat surface without any force affecting it but the force of gravity. By registering the acceleration values coming from the sensors of a stationary device, this test would allow measuring the level of noise affecting the calculations in the attempted solutions.

The various versions of the app presented in the previous chapter share a common testing methodology. Different data samples were recorded while the app was running with the device moving along one axis in both directions. The purpose of the tests was to assess how the estimated displacement differed from the starting position after moving the device back and forth along the same axis and returning it to the initial position (Figure 11). During the described motion, the displacement values were collected only when the device was facing the marker, as this position represented the starting point for tracking. All these values were stored in CSV files for further analysis through statistical tests. By

evaluating how far these displacement values are from the initial position, the relative accuracy of the various versions of the application was estimated.

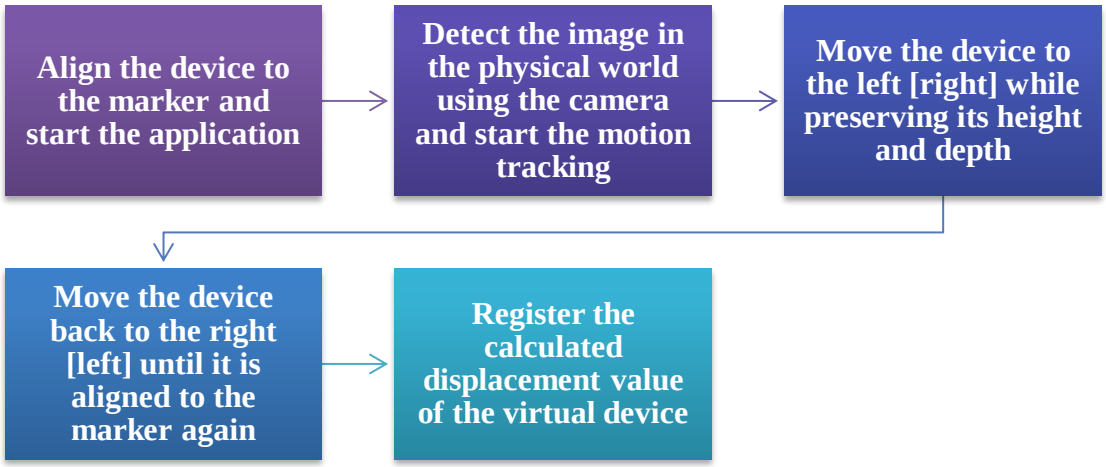

Figure 11: Testing methodology flowchart. The sequence of actions that were made while carrying out tests for the various versions of the hybrid app.

4.2 Sensors noise evaluation

In order to demonstrate how the noisiness of the sensors can affect the integration processes described in the previous chapter, two main tests have been carried out. The data coming from an Android device accelerometer has been collected for a set interval of time while the device was stationary on a flat surface. The first test allowed to collect the raw data coming from the sensors, without any sort of refinement, while the second one allowed to gather the output values obtained from a simplified Kalman filter (see Equation 2, page 12) applied to the raw acceleration. Both tests were carried out under the same exact conditions (described in chapter 4.1, page 14). The data has been stored in CSV files and analysed using the tools offered by the SPSS software.

The first test produced a set of raw acceleration samples represented by vectors containing three components, one for each axis (*X*, *Y*, and *Z*). About ten samples were removed from the bottom of the dataset in order to avoid having false outliers. Since the data was collected manually, the last few recordings of the test were influenced by the finger pressing the screen, which causes a random force to affect the sensors, especially on the local z-axis of the device. The analysis on the collected data showed that the population of samples is normally distributed on all three axes (Figure 12).

Tests of Normality

	Shapiro-Wilk		
	Statistic	df	Sig.
X	.999	734	.803
Y	.996	734	.083
Z	.997	734	.252

Figure 12: Normality test results for the raw acceleration samples. The three components of acceleration are normally distributed as the significance values given by the Shapiro-Wilk tests are greater than 0.05.

From the Descriptives and the boxplots (Figure 13) it is evident that most of the collected values vary within the range of ± 0.02 m/s^2. This is also confirmed by the interquartile range values for all the three coordinates (X: 2.02×10^{-2} m, Y: 2.01×10^{-2} m, Z: 2.41×10^{-2} m). Although most of the samples lie within this range, the minimum and maximum values go up to an absolute value of 0.07 m/s^2. This shows how the noisiness of the values is something that cannot be easily predicted and appropriately controlled. By comparing the values of the three components, it is possible to see that the *Z* component is the one

affected by the highest amount of noise. The standard deviation for the values on the z-axis is indeed the highest among the three components, meaning that this is the most variable set. The reason behind this result is that, among the three axes, the z-axis is the one that was mostly affected by force of gravity during the test. This is because the phone was fixed on a table with its back facing the surface (the z-axis on all android devices corresponds to the axis that perpendicularly cuts the screen, as shown in Figure 14).

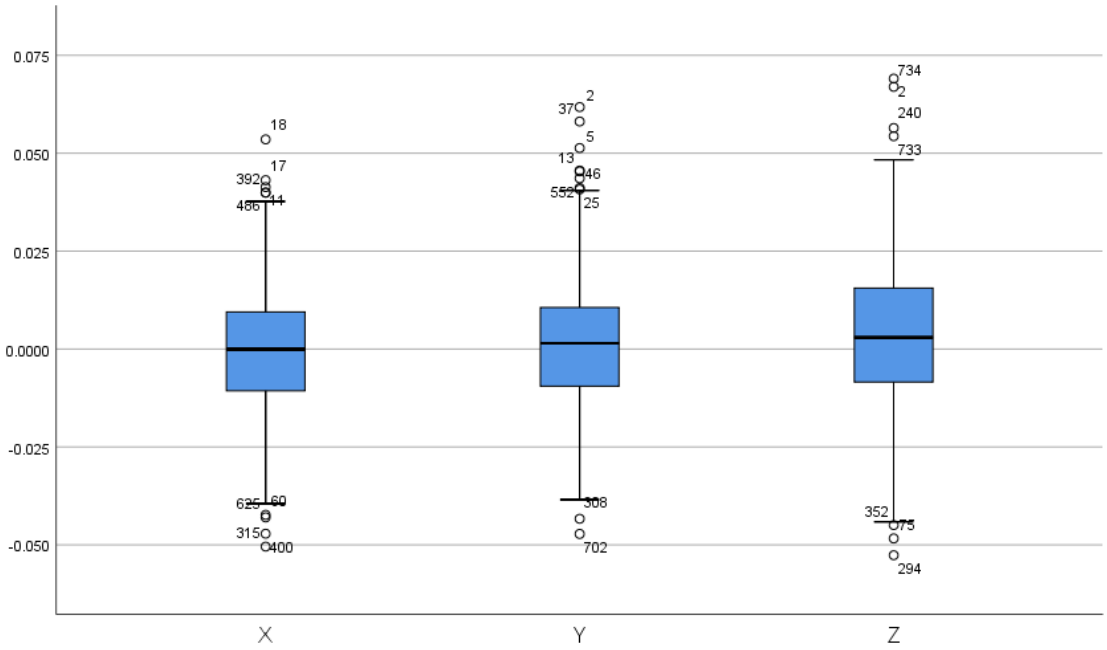

Figure 13: Raw acceleration data boxplot – X, Y and Z Coordinates.

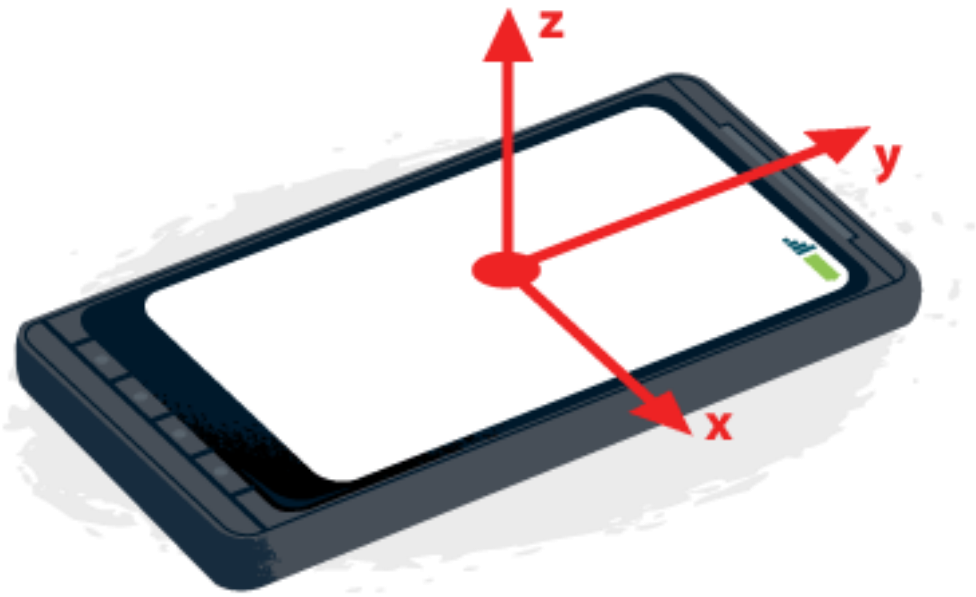

Figure 14: Illustration of device coordinate frame. The image shows the coordinate system that Android devices use to track movements and rotations with 6DOF (LePage, 2019).

The test was carried out to better understand the behaviour of the noise affecting the sensors, in an attempt to reduce its effects on the AR app's performance. By separating the positive and negative values of acceleration for each component, it is possible to determine thresholds for which the acceleration value could theoretically be considered noise and could therefore be ignored. By examining the positive and negative means for each individual coordinate, the thresholds can be calculated and then used to cut off any acceleration samples that lie within the evaluated range. Although this idea was implemented for both the solutions presented in the previous sections, it did not provide satisfactory results, as it caused small movements of the device to be completely ignored. Only movements that caused high changes in acceleration would be recognised by the application and translated into the relative displacement.

The second test that was carried out is very similar to the previous one but uses a simplified Kalman filter to reduce the noisiness of the collected data. The application was tested under the same conditions described before, with the Android device lying on a flat surface without any force affecting it but the force of gravity. The dataset containing the refined acceleration values was analysed using the same tools used for the first test, in order to highlight the main differences between the two.

In this case, according to the Shapiro-Wilk tests, the values for the three components of acceleration are not normally distributed (Figure 15).

Tests of Normality

	Shapiro-Wilk		
	Statistic	df	Sig.
X	.990	740	.000
Y	.982	740	.000
Z	.949	740	.000

Figure 15: Normality test results for the refined acceleration samples. The three components of acceleration are not normally distributed as the significance values given by the Shapiro-Wilk tests are lower than 0.05.

This means that, although the filter can help reduce the noisiness of the values, it also alters the distribution of the acceleration samples. When comparing the interquartile ranges' values in the two tests, it is clear that the refined acceleration samples lie within

a range that is much narrower than the previous one. The Descriptives for this analysis show that these values are about one order of magnitude smaller than the ones observed during the analysis of the first test (X: 5.71×10^{-3} m, Y: 6.12×10^{-3} m, Z: 7.45×10^{-3} m). Moreover, the minimum and maximum values for this dataset do not exceed the absolute value of 0.03 m/s^2. These results suggest that the filter effectively reduces the noisiness of the acceleration values coming from the sensors when the device is stationary, even though it does not nullify the noise completely. The *Z* component is the most variable set even in this case as it presents the highest value of variance among the three coordinates. This suggests that the noise would increase if the device were moving, as in this case many different forces other than the force of gravity would affect the device and therefore all the three axes would be subject to higher variance.

In order to assess whether the difference between the observed sets is significant a Wilcoxon Signed Ranks test was carried out individually on each component (Figure 16). This test was the most appropriate since the datasets were built using the same application, without any refinement in the first case and with the Kalman filter applied in the second case. The two datasets are related to each other, and since the filtered data is not normally distributed a non-parametric test for related samples was required. The test pairs for the Wilcoxon Signed Ranks test were built by merging the previously analysed datasets and by pairing the variables depending on the coordinate to which they belong. When carrying out this test, the samples that represented outliers in the first dataset were taken into consideration. This ensures that the pairs for the Wilcoxon test are correctly related based on time.

Test Statistics[a]

	FilteredX - RegularX	FilteredY - RegularY	FilteredZ - RegularZ
Z	-3.226[b]	-1.132[c]	-1.880[b]
Asymp. Sig. (2-tailed)	.001	.258	.060

a. Wilcoxon Signed Ranks Test

b. Based on negative ranks.

c. Based on positive ranks.

Figure 16: Wilcoxon Signed Ranks test result for the acceleration datasets. The test allows to highlight whether there is a significant difference between the raw acceleration values coming from the sensors and those refined through the Kalman filter.

Since the chosen data collection methodology did not provide a way to stop recording samples at exactly the same time on both tests, some of the samples at the end of the second dataset had to be cut off in order to match the number of samples in the first one. For the x-axis, the results of the Wilcoxon test show a significant value at the 5% level (Wilcoxon Test $Z = -3.226$, $p = 0.001$), meaning that the previously observed difference is significant on that specific axis. For the y and z axes, the results of the Wilcoxon test show that there is no statistically significant difference between the two sets. Taken together, these results suggest that although there is a visible difference between the two collected datasets, this cannot be considered significant. Although the simplified Kalman filter reduces the noisiness of the sensors slightly, it does not have a significant impact on the performance of the application, as there is still enough noise leftover to affect the output of the integration processes.

4.3 ARCore approach testing

The first version of the application, that was built using ARCore's approach to track the device's movements over time, was tested by using a strategy that is common to the two attempted hybrid solutions. All three versions of the app were tested in both featureless and regular environments, in order to compare the performances of each solution in the two scenarios. During every test, the AR-enabled device was held vertically on the edge of a flat surface, with its camera facing an easily recognisable marker. Thus, the Augmented Images system was able to immediately detect the image, and then to initiate motion tracking. When collecting data, the device was moved horizontally while trying to keep it close to the edge of the surface. Since this process was carried out without any of the previously mentioned specific tools (see chapter 4.1, page 14) for governing the device's motion along the surface, there might be some errors in the gathered values, as all the movements were subject to the inaccuracy of hand motion. The main goal of these tests is to assess how precise the tracking of the application can be, by looking at the displacement values that are calculated while the device is moving back and forth along the same axis (the x-axis in this case). In an ideal system that tracks the user's movements perfectly, the X coordinate of the virtual device is equal to 0 when the device and the marker are perfectly aligned in the real-world environment. This is because the initial position of the device in the virtual scene is set to zero when the marker is recognised by the Augmented Images feature set.

The main assumption that was made for all these tests is that the trend in the values for the *X* coordinate of the device displacement is similar to the ones of the other two components (*Y*, and *Z*). This is reasonable because the calculations that are performed to evaluate all three coordinates are exactly the same in all the versions of the application. Moreover, as it was already shown by the previous analysis on accelerometer's data, the average values of noise for all three coordinates are really similar to each other. Therefore, if the applications were tested under the same conditions but while moving the device along a different axis, the final results would be approximately, if not exactly, the same. The main reason behind the choice of the x-axis is that the horizontal direction represents the most comfortable direction for manually moving objects while following a precise guideline to reduce errors.

The first test with this version of the app was carried out on a regular environment, filled with objects that allow the camera to detect several features in the real-world environment. The marker was placed on top of a table at a certain distance from the edge of the surface that was used as a guideline for the device's motion. Starting from the initial position, the device was moved to the left of the marker until the border of the table was reached and then back to the right up to the other end of the table. Throughout this process, the displacement values were recorded only when the camera was physically aligned with the marker. Since the experiments were carried out by solely relying on hand motion, the alignment between the device and the marker could not be measured down to the millimetre, therefore the collected data samples might be affected by approximation errors. The data was stored in CSV files and analysed through the use of SPSS tools.

Firstly, a test of normality (Figure 17) was carried out on the dataset in order to decide how to further analyse the data. According to the Shapiro-Wilk test, the values of the *X* components of the displacement samples that were collected during this first test are normally distributed.

Tests of Normality

	Shapiro-Wilk		
	Statistic	df	Sig.
ARCoreApproachX	.942	15	.414

Figure 17: Normality test results for the ARCore approach displacement samples. The X components of displacement are normally distributed as the significance value given by the Shapiro-Wilk test is greater than 0.05.

By looking at the Descriptives and the boxplot (Figure 18), it is possible to see that the values are overall very close to zero. The interquartile range for the dataset is equal to 1.99×10^{-2} m, while the minimum and maximum have values of -5.52×10^{-2} m and 2.17×10^{-2} m, respectively. The mean value of displacement is very close to 0.01 m, meaning that this version of the app can be considered accurate down to the centimetre.

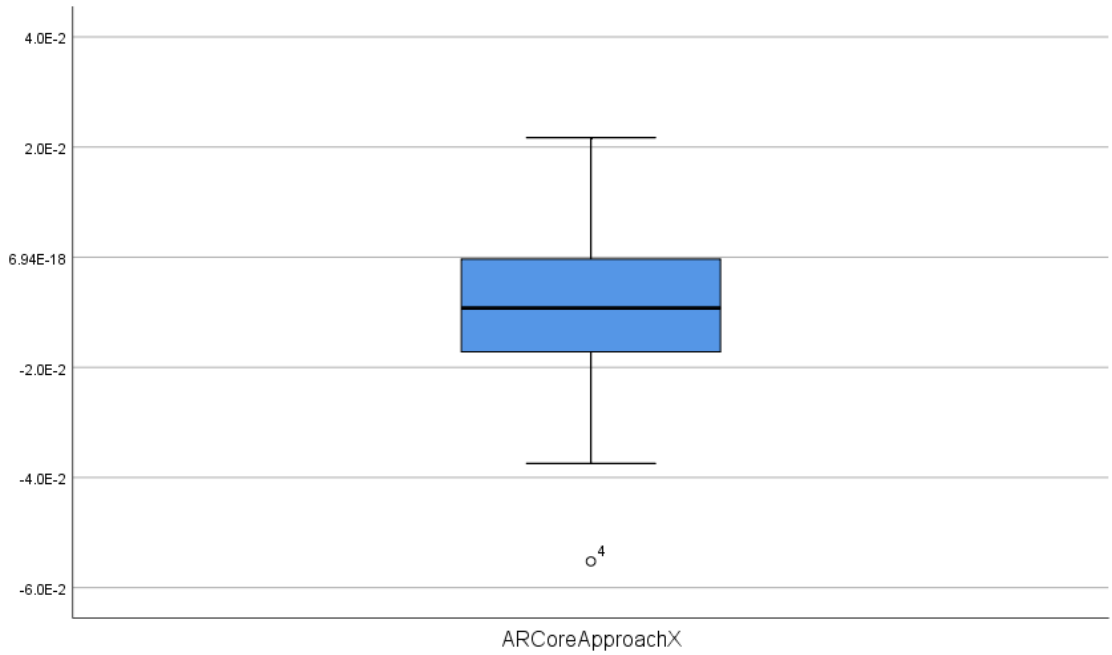

Figure 18: ARCore approach displacement boxplot - X Coordinates.

The analysis suggests that the application is accurate enough, as most of the values are close to the estimated displacement. In order to assess whether there is a statistically significant difference between the samples in the dataset and the expected value, a one-sample t-test was carried out (Figure 19). This test is the most appropriate for this data since the displacement values are normally distributed on the x-axis, as shown by the Shapiro-Wilk test.

One-Sample Test

	Test Value = 0					
	t	df	Sig. (2-tailed)	Mean Difference	95% Confidence Interval of the Difference	
					Lower	Upper
ARCoreApproachX	-2.253	14	.041	-1.076674E-2	-2.101707E-2	-5.164091E-4

Figure 19: Results of the one-sample t-test for the ARCore approach dataset. The test allows to determine whether the mean value of the samples in the dataset is equal or different to a predetermined expected value.

The results of the test show that at a 5% significance level, the mean of the analysed data cannot be considered to be statistically equal to the expected value, zero. The p-value for the test is lower than 0.05 therefore there is a significant difference between the dataset's mean and the predicted value. However, the evaluated mean difference is equal to -1.07×10^{-2} m, and this is a negligible value in most of the scenarios in which the app is employed.

The results of this first test indicate that although the application built using ARCore's approach is visibly accurate, there is still a significant difference between the calculated displacement and its expected values. This might be due to two main reasons. The former is related to the accuracy limitations that affect even sophisticated systems as Google's AR software. Both Android sensors and the camera's feature detection system cannot be flawlessly precise, as they still introduce some sort of approximation to the calculations in order to make the final products responsive and interactive. The latter is instead related to the testing methodology. As it has already been stated before, since every data sample was recorded by manually moving the device over the surface, the displacement values might be subject to accuracy errors. By just relying on hand motion the alignment between the marker and the device could not be precisely measured. Therefore, the collected displacement samples could have been recorded in positions that differed, even slightly, from the initial pose.

The second test was carried out under similar conditions, but in a featureless environment. The device was held vertically on the edge of a flat surface while its camera was facing a completely plain wall. Thus, the marker would be recognised, tracking would still start correctly, but after moving the device for a certain distance the ARCore system would not detect enough features to correctly estimate the device's displacement. For this test, the marker was fixed on the left side of the wall, leaving the right side of the wall completely featureless. All the limitations and the assumptions made for the first test can be applied to this test as well.

Throughout the test, in order to gather samples of displacement, the device was moved in a similar pattern as the first test. Since the marker was fixed on the left side of the wall, the device could only be moved to the right and then back to the left to face the marker once again. Even in this case, the displacement samples were recorded during motion

only when the marker and the device were physically aligned. The dataset was stored in a CSV for further exploration through SPSS tools.

A Shapiro-Wilk test for normality (Figure 20) was carried out even in this case to determine the distribution of the population of samples.

Tests of Normality

	Shapiro-Wilk		
	Statistic	df	Sig.
ARCoreApproachX	.926	15	.236

Figure 20: Normality test results for the ARCore approach displacement samples in a featureless environment. The X components of displacement are normally distributed as the significance value given by the Shapiro-Wilk test is greater than 0.05.

The results of the test suggest that the collected dataset contains a population of values that is normally distributed. The Descriptives and the boxplot (Figure 21) for this analysis show that the range of values for this test is much wider when compared to the one of the first test. Moreover, most of the values in the set are positive values and so is the mean displacement value as well. This is due to the testing methodology used to carry out this specific test.

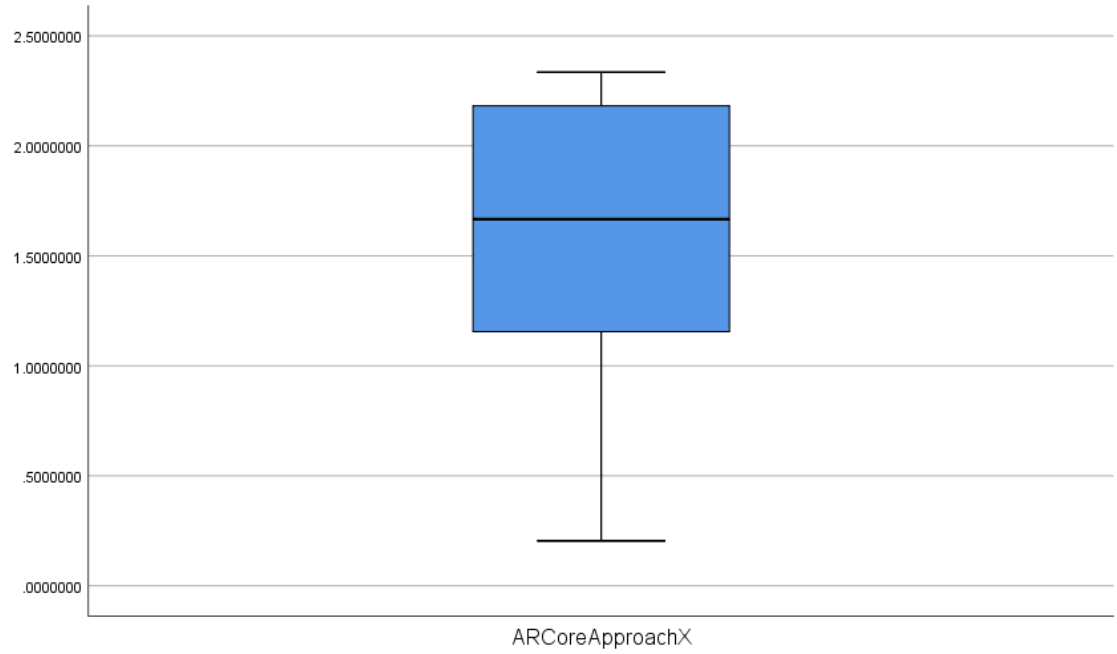

Figure 21: Boxplot for the ARCore approach displacement data in a featureless environment - X Coordinates.

Since the marker was fixed on the left side of the wall, the first motion the device could make was a linear motion to the right. During the test, the ARCore system would stop tracking any sort of movement whenever the marker moved out of the frame. Whenever the app was not able to detect enough features, the displacement value would stop being updated, until some other features were finally detected. This specifically happened when the device moved back to the left and matched the marker position again. At that point, the displacement value for the virtual device would resume its updating process, but the starting value would be altered. This is because, due to the lack of features in the environment, the movements on the way back were not registered by the ARCore tracking system. For instance, if the device had moved 1 meter to the right before pausing the displacement updates, the starting position for the virtual device would still be 1 meter when the device is back to its starting position. This is the reason why most of the samples in the dataset are positive values.

These results confirm the lack of precision of markerless AR systems under unfavourable conditions and therefore support the purpose of this research. As it is explained in more detail in the following sections, the proposed hybrid approach does not suffer from this issue, as it does not rely on the camera detection features. In order to confirm that the collected values are statistically different from the expected displacement, a one-sample t-test was carried out on this dataset (Figure 22). Even in this case, this is the most appropriate test as the Shapiro-Wilk test for normality showed that the population of samples is normally distributed. The results of the test indicate that the mean displacement value for the samples in the dataset is significantly different from the expected value, zero. The mean difference for this test is way higher than the one highlighted in the analysis of the first test dataset, and in this case it is not negligible. This demonstrates how the app becomes more inaccurate as the environmental conditions become unfavourable for the correct execution of the markerless system.

One-Sample Test

	Test Value = 0					
					95% Confidence Interval of the Difference	
	t	df	Sig. (2-tailed)	Mean Difference	Lower	Upper
ARCoreApproachX	9.326	14	.000	1.593253967	1.226831367	1.959676566

Figure 22: Results of the one-sample t-test for the ARCore approach dataset in a featureless environment. The test allows to determine whether the mean value of the samples in the dataset is equal or different to a predetermined expected value.

4.4 First approach testing

The acceleration array approach that was described in the previous chapter was tested using a methodology that is similar to the one adopted for the ARCore approach tests. Two tests have been carried out to look at the behaviour of the app in both regular and featureless environments. Throughout testing, the device was held vertically and moved along the edge of a flat surface while facing the marker. Therefore, only the *X* components of the gathered displacement vectors were considered when carrying out the analysis. The motion patterns used to test the previously analysed version of the app were replicated in order to acquire results that are comparable.

The purpose of these tests was to assess the accuracy and reliability of the application by looking at the calculated displacement values after moving the device back and forth from a fixed starting point. As it was previously mentioned, in an ideal situation the collected samples should all have values that are exactly equal to zero. By performing a one-sample t-test, the difference between the mean value of the population and the expected result was evaluated. Data collection and analysis were carried out by using the same strategies and tools presented in the previous sections.

The first test was carried out in a regular environment filled with objects that enrich the scene with several features. Although in both of the attempted hybrid solutions this is not a requirement for the correct flow of the application, having the same set of tests for all versions of the app helps making comparisons between them. The expected result in this case is to have statistical figures that are similar for both tests, as the environment should not affect the performance of the system at all. A Shapiro-Wilk test for normality (Figure 23) was carried out in order to determine the distribution of the collected data samples. The results of the test were used to determine which approach to follow when further analysing the dataset.

Tests of Normality

	Shapiro-Wilk		
	Statistic	df	Sig.
AccelArrayX	.925	15	.233

Figure 23: Normality test results for the acceleration array approach displacement samples in a regular environment. The X components of displacement are normally distributed as the significance value given by the Shapiro-Wilk test is greater than 0.05.

The results of the test show that the population of samples is normally distributed, as the observed p-value is greater than 0.05. This result suggests that the dataset can be further analysed using parametric tests. By looking at the Descriptives and the boxplot (Figure 24) it is already clear that the collected values are not equally distributed around to the zero. The mean value for the dataset is equal to −0.6 m, and both the minimum and maximum values are negative.

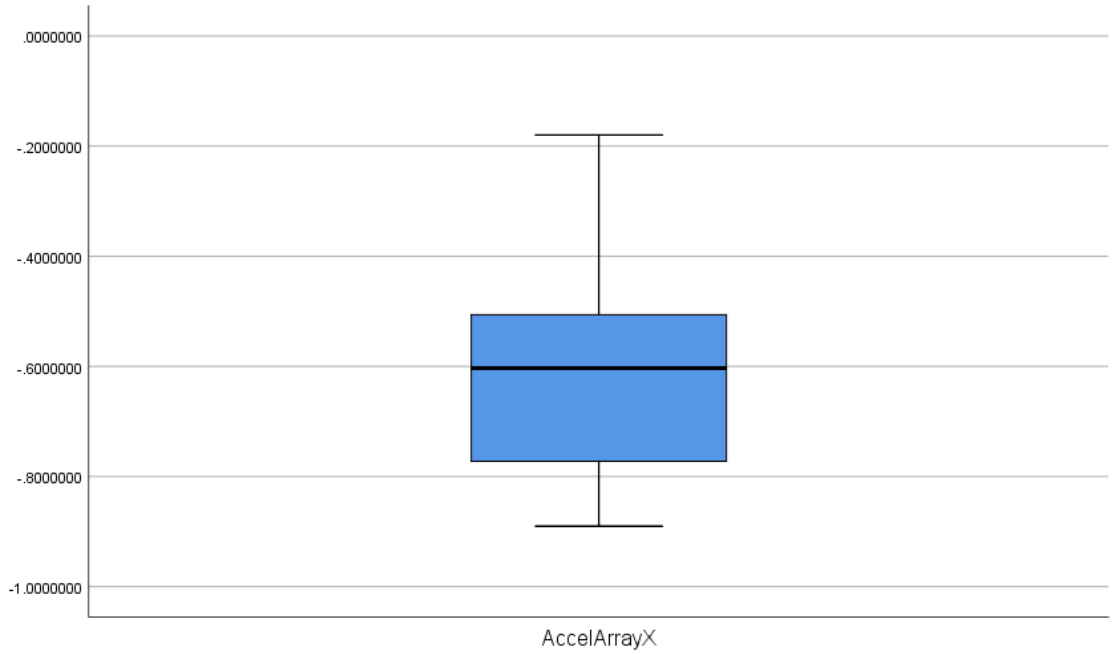

Figure 24: Boxplot for the acceleration array approach displacement data in a regular environment - X Coordinates.

In order to assess whether there is a significant difference between the collected displacement samples and their expected values, a one-sample t-test (Figure 25) was carried out. Even in this case, this is the most suitable test as the data has been previously confirmed to be normally distributed.

One-Sample Test

	Test Value = 0					
					95% Confidence Interval of the Difference	
	t	df	Sig. (2-tailed)	Mean Difference	Lower	Upper
AccelArrayX	-10.453	14	.000	-.5997995800	-.722872891	-.476726269

Figure 25: Results of the one-sample t-test for the acceleration array approach dataset in a regular environment. The test allows to determine whether the mean value of the samples in the dataset is equal or different to a predetermined expected value.

The test shows a significance value that is lower than 0.001, meaning that the analysed difference can be confirmed to be significant. The evaluated mean difference is very close to the mean value of the entire dataset, and it is not as a negligible value as the one seen in the one-sample t-test for the ARCore approach.

Taken together, these results suggest that this first attempted solution is not as accurate as the previously analysed one. This is an expected result, following the limitations that were already mentioned in the previous chapters. The prevalence of negative values in the collected dataset is due to the chosen testing methodology. Throughout the various tests in regular environments, after the tracking was started, the first movement done by the user was a movement to the left. This motion induces a negative acceleration on the device, which for this version of the app is recorded in terms of acceleration values inside the acceleration array. One of the issues of the currently analysed approach is that the values kept inside the data structure can sometimes badly affect the calculations. Since the values inside the array come from previous frames, there is always a considerable amount of importance given to the past when calculating the current displacement of the virtual device. Therefore, when moving the device back towards the starting position, the actual acceleration value – which should be positive in this case – would most of the times end up being negative or close to zero. Thus, since the displacement vectors are almost always evaluated using negative accelerations, their values end up being negative as well.

The well-known limitations of Android sensors also contribute to the inaccuracy of the app. The data fetched from the accelerometer and the gyroscope is naturally noisy and causes small errors in the calculations that become very big when cumulated over time. Moreover, the integration process that is carried out on the raw acceleration data introduces conspicuous approximation. By converting acceleration into velocity, and then velocity into displacement the app is integrating twice, and this magnifies the inaccuracies. This issue can be tackled by using an external source of information that is able to appropriately correct the imprecision of the data coming from the sensors. ARCore does that by pairing Android sensors with the camera's feature detection system. By looking at the visual differences between two consecutive frames, ARCore systems are able to deduce the direction of motion, which is then confirmed by the sensors data. This allows to compensate for any noise and allows to filter the input data so that only real motions of the device are taken into consideration. By only having information on the

sensors' states, it is very complicated to emulate the real motion of the device, as there are many factors that can affect the acceleration values in similar ways.

A second test was carried out using a similar methodology, but in a featureless environment. The purpose of the test was to confirm that the performances of the app are not related to the conditions of the environment. The collected data was analysed through the same set of statistical tests and similar results were found. The population of samples is normally distributed even in this case, and the difference between the mean displacement and its expected value was found to be significant. The boxplot (Figure 26) highlights how the trend of values is very close to the one observed in the analysis of the first test.

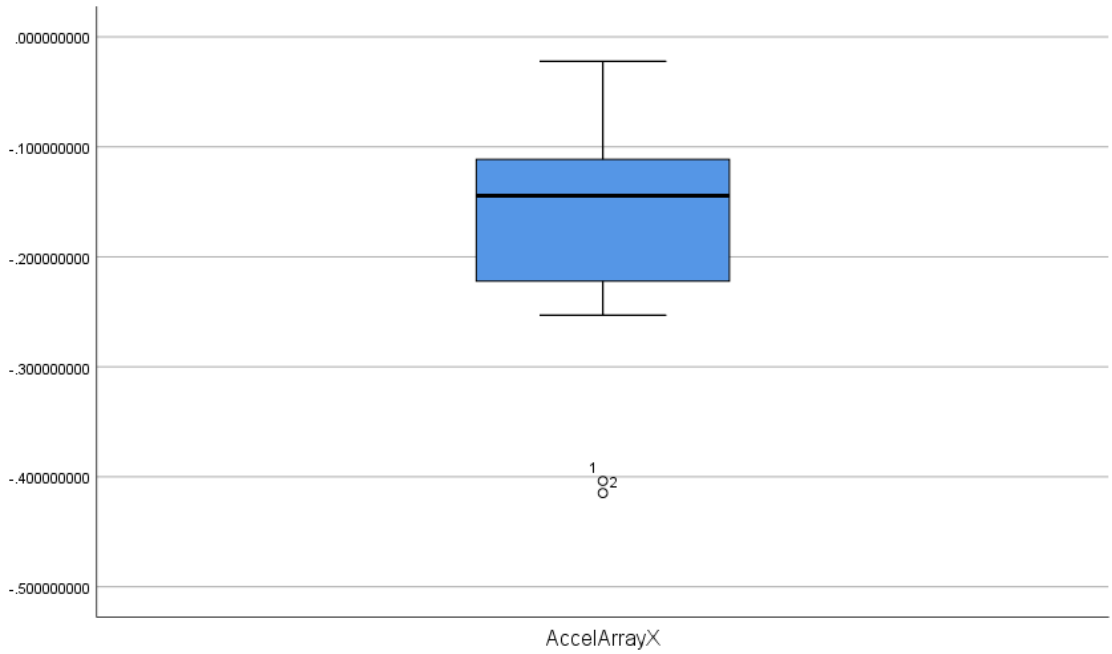

Figure 26: Boxplot for the acceleration array approach displacement data in a featureless environment - X Coordinates.

One-Sample Test

	Test Value = 0					
	t	df	Sig. (2-tailed)	Mean Difference	95% Confidence Interval of the Difference	
					Lower	Upper
AccelArrayX	-5.832	14	.000	-.175958865	-.240666527	-.111251203

Figure 27: Results of the one-sample t-test for the acceleration array approach dataset in a featureless environment. The test allows to determine whether the mean value of the samples in the dataset is equal or different to a predetermined expected value.

The mean difference found with the one-sample t-test in this version of the application (Figure 27) is smaller than the one seen during the analysis of ARCore's approach in a featureless environment. However, this value is still non negligible when thinking of scenarios in which the application would be used, and it is still very high when compared to the mean difference seen in the regular test for ARCore's version. This shows how the hybrid approach can give relatively better results under unfavourable conditions for the markerless techniques, but still not accurate enough to make it a reliable system.

4.5 Second approach testing

The final set of tests was carried out on the last attempted solution which features the cumulative acceleration approach. As it was already described in the previous chapter, the idea behind this approach was to solve the issues of the acceleration array approach, especially the heavy importance given to the past samples of acceleration. Two tests were carried out even in this case, one in a regular environment and the other in a featureless environment. Obviously, the results should not differ between the two tests since the application does not rely on feature detection algorithms, but only on sensors data. The testing methodology for both tests is the same as the one presented in the previous sections.

The purpose of these tests is again to assess the accuracy and reliability of the application by looking at the evaluated displacement values. The previously presented ideal scenario applies even in this case. Data collection and analysis were carried out by using the same strategies and tools presented in the previous sections. The same set of statistical tests was used in order to allow easier comparisons with the previous approaches.

The first test was carried out in an environment that contained several objects to enrich the scene with different features. Data collection was carried out in a similar way as before, by using a flat surface as a guideline for the device movements. The displacement samples were collected in CSV files and then analysed using SPSS tools. A Shapiro-Wilk normality test (Figure 28) was first carried out on the data to determine the distribution of the population of samples.

Tests of Normality

	Shapiro-Wilk		
	Statistic	df	Sig.
CumulativeAccelX	.872	15	.037

Figure 28: Normality test results for the cumulative acceleration approach displacement samples in a regular environment. The X components of displacement are not normally distributed as the significance value given by the Shapiro-Wilk test is lower than 0.05.

The results of the test show that the dataset contains values that are not normally distributed, as the significance value reported by the test is lower than 0.05. This already suggests that the analysed data is different in some ways from the previously analysed datasets. The Descriptives and the boxplot (Figure 29) present a wide range of values for the displacement samples, with a median value of −5.38 m and an interquartile range of slightly more than 12 m. Moreover, the minimum and maximum values for the set are very distant from each other (−28.04 m and 3.02 m respectively).

It is already possible to predict that the outcome of this analysis will be very different from the results of the previous tests. The main reason behind this is that this version of the application is the one the suffers the most from sensors drift. This phenomenon is a well-known limitation that affects most inertial measurement units (IMUs), such as the accelerometer and the gyroscope. Drift can be defined as the ever-increasing difference between where the system thinks it is located and its actual location. It is caused by internal calculations of the IMUs that contribute to the process of dead reckoning. This is a concept that finds its origins in the field of navigation. The process of dead reckoning consists in calculating one's current position by using a previously determined position and estimations of speed over elapsed time. Dead reckoning is subject to cumulative errors and requires external systems, such as the Global Positioning System (GPS) in order to function properly. This is an analogous scenario to the one described when presenting ARCore's solution to the problem, where the camera's feature detection system and the Android sensors were the protagonists. Since the cumulative acceleration approach never resets the acceleration value, the errors caused by the sensors drift are continuously stockpiled and this causes the integration processes to have heavily distorted inputs and to consequently produce inaccurate outputs. Indeed, a constant error in the acceleration values coming from the accelerometer results in a linear error in velocity and a quadratic error growth in position.

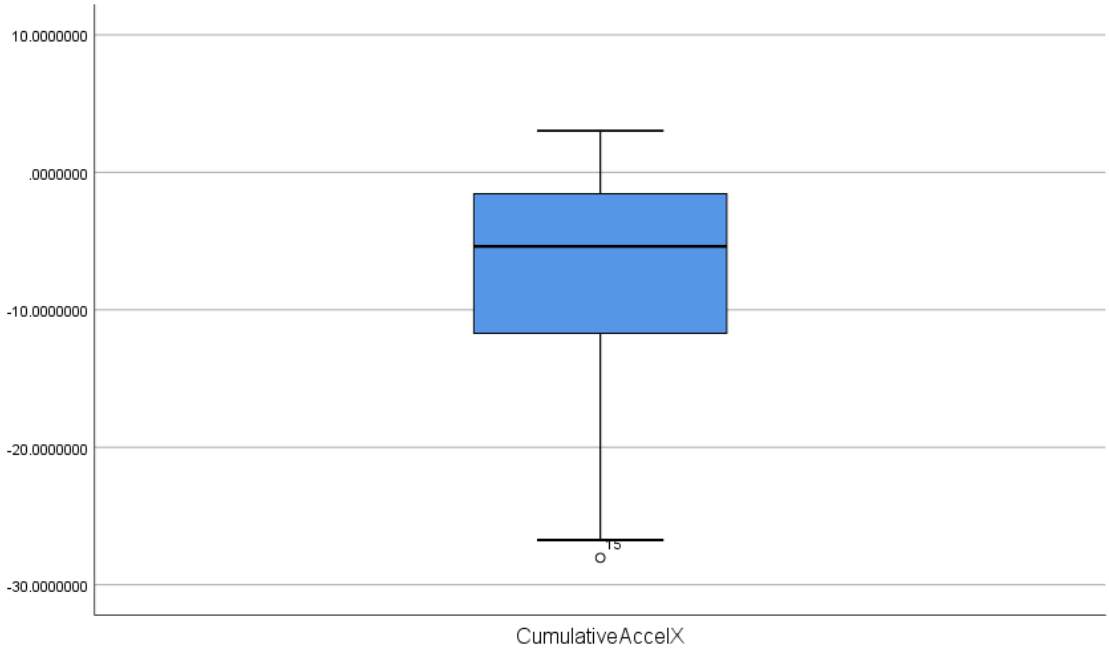

Figure 29: Boxplot for the cumulative acceleration approach displacement data in a regular environment - X Coordinates.

The most crucial issue with the cumulative acceleration approach, however, is that there is no specific rule to determine the state of motion of the device when the registered acceleration is zero. In this scenario, the evaluated velocity value for the virtual device would need to be set to zero only when the physical device is not moving. Without using any sort of external information, and by just relying on internal forces that affect the device – those registered by the sensors – there is no exact way of overcoming this limitation. The application would either be too restrictive by forcing the velocity value to zero every time the registered acceleration is null or keep the old velocity value until a new change in acceleration is detected. In both cases, the app would perform poorly. In the former, all the device movements that happen at a constant velocity would be completely ignored, therefore only sudden shifts would be categorized as real motion. In the latter, the virtual device would indefinitely keep on moving at a constant velocity – the last one outputted by the integration process – even when the user is not applying any force to the device. This last approach gave better results in more scenarios than the first one did; therefore, it was implemented in the cumulative acceleration approach. This decision explains why the values in the dataset are subject to a sharp increase towards the end. Although the test of the application started by reporting quite accurate approximations of the displacement values, as the app progressed in its execution, the small errors from both the integration processes and the sensors themselves piled up and

thus altered the position of the virtual device. During the test, the calculated velocity quickly drifted to a value that was so high that it could not be adjusted anymore by the successive changes in acceleration. The high velocity inevitably caused the displacement values to sharply rise even when the device was at rest.

It is possible to overcome this limitation by introducing any external factor that provides information on the device that is not related just to its internal state. For instance, the way ARCore solves the issue is by using the camera detection system to analyse and compare consecutive frames. The application in this case has an impeccable criterion to properly reset the velocity value. Whenever no different features are detected between two or more successive frames, it is safe to assume that the device is not physically moving. This condition would confirm that the acceleration is null because the device is in a resting state and would therefore guarantee that resetting the velocity value is not a mistake.

To complete the analysis of the dataset a statistical test has been carried out to determine the significance of the difference between the median displacement and its expected value. Since the data was confirmed to not be normally distributed, a non-parametric sign test (Figure 30) was carried out on the data as an alternative to the one-sample t-test. An auxiliary variable containing the expected value of displacement was created in order to construct the samples pairs that are required by the test.

Hypothesis Test Summary

	Null Hypothesis	Test	Sig.	Decision
1	The median of differences between CumulativeAccelX and MedianX equals 0.	Related-Samples Sign Test	.118[a]	Retain the null hypothesis.

Asymptotic significances are displayed. The significance level is .050.

Figure 30: Results of the non-parametric sign test for the cumulative acceleration approach dataset in a regular environment. The test assesses whether there is a significant difference between the median value of the samples and a predetermined expected value.

Although the results of the test show that the difference is not significant, it is clear from the Descriptives that this is not the case. This is also evident when noticing that the number of ties within the two groups is equal to zero.

A second test was carried out on this version of the app in order to confirm the consistency of its behaviour in a featureless environment. As for the previous approach the expected result is to have similar figures as the ones seen in the first test. This is because this

approach does not rely on the camera's feature detection systems to perform motion tracking. Data collection was carried out by fixing the marker on the left-hand side corner of a featureless wall and by moving the device back and forth along its local x-axis.

The distribution of the data samples is very similar to the one shown by the analysis of the previous dataset. The trend in values highlighted by the boxplot (Figure 31) reflects the one seen in the previous analysis. This time, the velocity and the consequent displacement drift were mostly positive, and this is due to the described testing methodology. Indeed, the first movement performed by the user after the marker was recognised was always a motion to the right-hand side, which corresponds to the positive side of the local x-axis of the device.

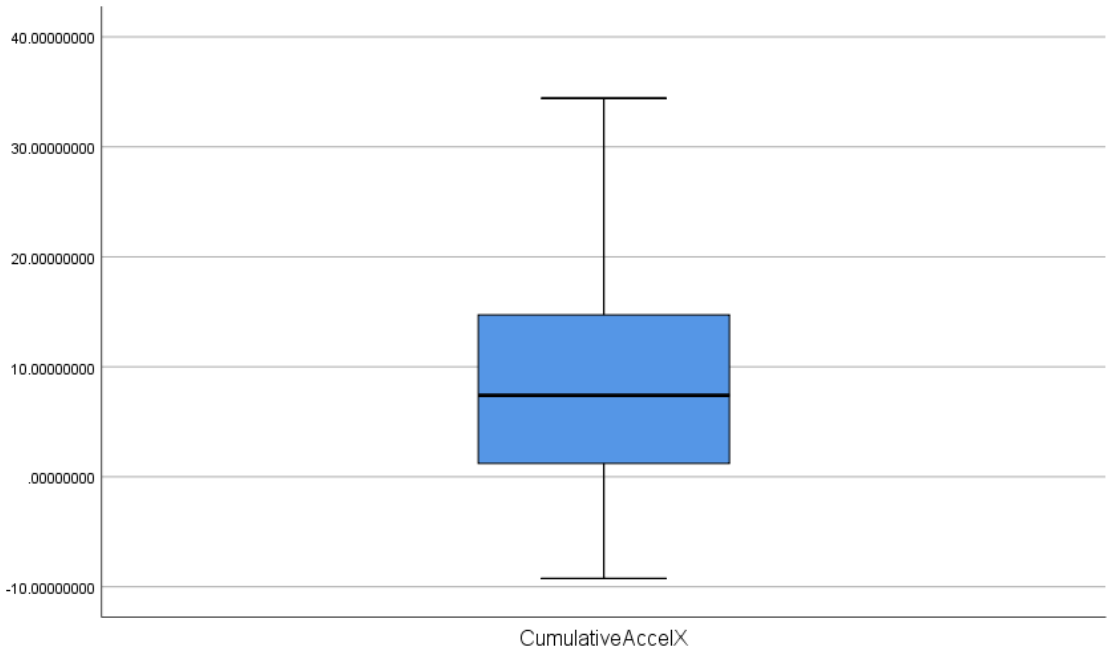

Figure 31: Boxplot for the cumulative acceleration approach displacement data in a featureless environment - X Coordinates.

4.6 Data comparison

The analysed datasets were compared through the Friedman test in order to highlight whether the revealed differences between the various approaches presented are significant or not. This statistical test is a non-parametric alternative to the one-way ANOVA test, which allows to evaluate differences between groups that are made of ordinal or continuous variables. The Friedman test was the most suitable test for the current scenario as it allows having one or more groups of data that are not normally distributed.

The *X* components of the displacement samples collected through the regular tests were gathered in a single SPSS file. The three groups that were analysed contain respectively the values obtained from the tests that were carried out on each version of the application. Since the group containing the samples from the cumulative acceleration approach is not normally distributed, the Friedman test (Figure 32) was chosen as the statistical solution to evaluate the differences between the datasets.

Ranks

	Mean Rank
AccelArrayX	1.80
ARCoreApproachX	2.67
CumulativeAccelX	1.53

Test Statistics[a]

N	15
Chi-Square	10.533
df	2
Asymp. Sig.	.005

Figure 32: Results of the Friedman test for the regular datasets. The test determines whether there are differences somewhere between the related groups, without indicating where those differences exactly lie.

The test results show that there is a statistically significant difference in the displacement values obtained from the three different versions of the application ($\chi^2(2) = 10.533$, $p = 0.005$). Since the Friedman test reported a statistically significant result, a Post Hoc analysis was conducted in order to highlight where the detected differences lie. For this purpose, separate Wilcoxon signed-rank tests (Figure 33) were carried out on the three different combinations of the related groups.

Test Statistics[a]

	ARCoreApproachX - AccelArrayX	CumulativeAccelX - AccelArrayX	CumulativeAccelX - ARCoreApproachX
Z	-2.783[b]	-2.726[c]	-2.669[c]
Asymp. Sig. (2-tailed)	.005	.006	.008

a. Wilcoxon Signed Ranks Test

b. Based on negative ranks.

c. Based on positive ranks.

Figure 33: Results of the Post Hoc Tests for the regular datasets. Three separate Wilcoxon signed-rank tests were carried out on the different combinations of the related groups in order to highlight the differences detected by the Friedman test.

The Post Hoc tests were conducted with a Bonferroni adjustment applied, resulting in a significance level set at $p < 0.017$. This was required to avoid incurring into a Type I error, which is very likely to occur in analyses that involve multiple comparisons. There

was a statistically significant difference in displacement between the ARCore and Acceleration Array approaches (Z = −2.783, p = 0.005). Similar results were seen in the other two group pairs: Cumulative Acceleration and Acceleration Array (Z = −2.726, p = 0.006), Cumulative Acceleration and ARCore (Z = −2.669, p = 0.008). These results confirm that the already predicted differences between all three approaches are statistically significant. By looking at a comparison between the boxplots (Figure 34) it is possible to state that the ARCore approach is the one that provides the best approximations for displacement.

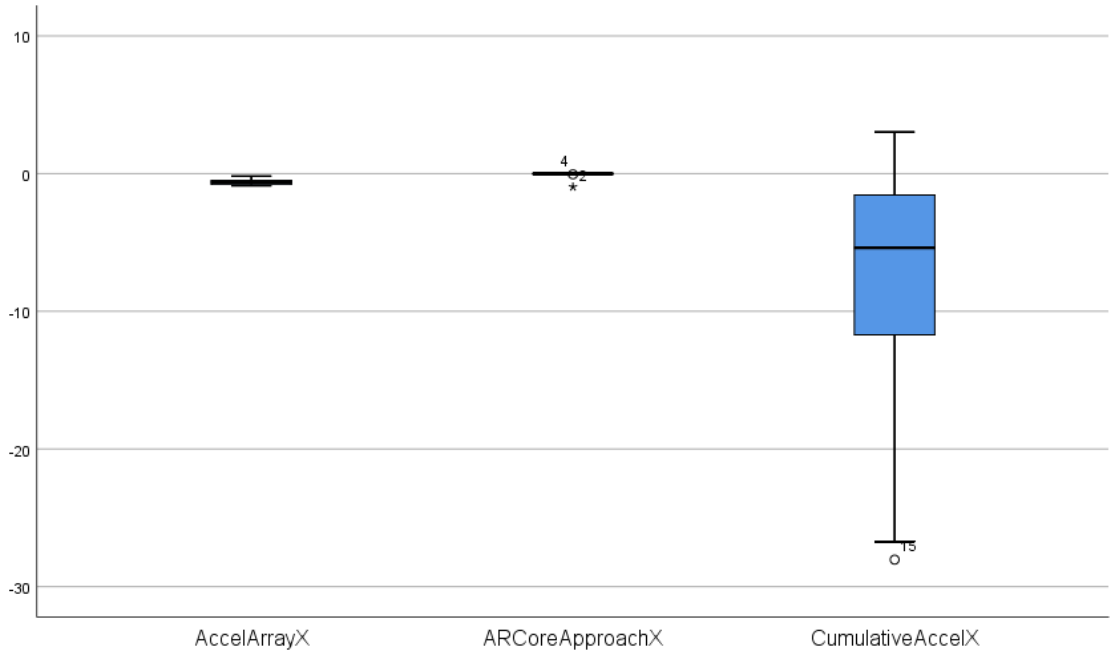

Figure 34: Comparison between boxplots for the three different versions of the application - Regular tests data.

The Acceleration Array approach, although presenting various issues that were described in chapter 3, provides better approximations to the ones given by the Cumulative Acceleration approach, which really suffers from sensors drift. None of the approaches provides exact matches with the estimated value of displacement, but the ARCore approach is the only solution among the three that offers reliable outputs from the calculations.

The same set of statistical tests was carried out on the datasets collected through testing in featureless environments. The three groups for the Friedman test were built by gathering the *X* components of the displacement samples obtained through the previously

presented tests. The Friedman test (Figure 35) was the most appropriate statistical solution as the cumulative acceleration approach dataset was not normally distributed.

Ranks

	Mean Rank
AccelArrayX	1.27
ARCoreApproachX	2.27
CumulativeAccelX	2.47

Test Statistics[a]

N	15
Chi-Square	12.400
df	2
Asymp. Sig.	.002

Figure 35: Results of the Friedman test for the featureless datasets. The test determines whether there are differences somewhere between the related groups, without indicating where those differences exactly lie.

The results of the test indicate that there is a statistically significant difference in the displacement values obtained from the three different versions of the application ($\chi^2(2) = 12.400$, $p = 0.002$). Since the Friedman test reported a statistically significant result, a Post Hoc analysis was conducted in order to highlight where the detected differences lie. For this purpose, three separate Wilcoxon signed-rank tests (Figure 36) were carried out on the different combinations of the related groups.

Test Statistics[a]

	ARCoreApproachX - AccelArrayX	CumulativeAccelX - AccelArrayX	CumulativeAccelX - ARCoreApproachX
Z	-3.408[b]	-2.385[b]	-2.045[b]
Asymp. Sig. (2-tailed)	.001	.017	.041

a. Wilcoxon Signed Ranks Test

b. Based on negative ranks.

Figure 36: Results of the Post Hoc Tests for the featureless datasets. Three separate Wilcoxon signed-rank tests were carried out on the different combinations of the related groups in order to highlight the differences detected by the Friedman test.

In order to avoid errors, the Post Hoc tests were conducted with a Bonferroni adjustment applied, resulting in a significance level set at $p < 0.017$. There were no significant differences between the Cumulative Acceleration and Acceleration Array approaches ($Z = -2.385$, $p = 0.017$) or between the Cumulative Acceleration and ARCore approaches ($Z = -2.045$, $p = 0.041$), despite an evident discrepancy between these last two groups. However, there was a statistically significant difference in displacement between the ARCore and Acceleration Array approaches ($Z = -3.408$, $p = 0.001$). Although the results of the Post Hoc tests show that the differences between the Cumulative Acceleration and Array Acceleration approaches and between the Cumulative Acceleration and ARCore

approaches are not significant, it is clear from the comparison of the boxplots (Figure 37) that this is not the case.

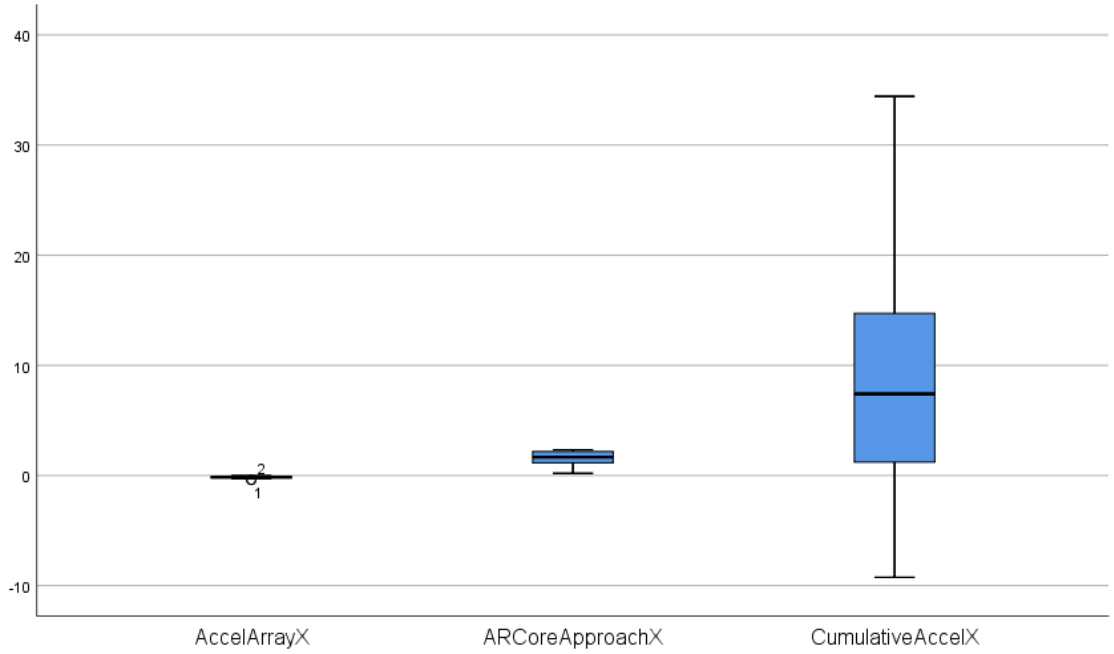

Figure 37: Comparison between boxplots for the three different versions of the application - Featureless tests data.

It is clear from the boxplots that the Cumulative Acceleration approach presents values that are way different compared to the ones of the other two solutions, as it is still the solution that suffers the most from sensors drift. For the featureless tests, since the environmental conditions were not optimal for the correct functioning of the ARCore solution, the approach that is the closest to the estimated result is the Acceleration Array approach. This demonstrates the potential of the hybrid AR technique that however comes with several weaknesses and reliability issues.

Taken together, these results show that the ARCore approach represents so far the best solution among the three, especially when employing the app in environments that provide enough features for the system. The app might be affected by failures under unfavourable scenarios, but it is able to reliably cope with sensors inaccuracy and drift. The proposed hybrid AR solutions did not offer the same level of performance as the ARCore approach, but there might still be ways to take advantage of their hidden potential. The following chapter describes a handful of solutions as future work for the app, which may provide more satisfactory results.

5 Conclusions and Future Work

Augmented Reality is a highly interactive experience of a real-world environment that merges reality with computer-generated perceptual information in a seamless way. Markerless AR – the most commonly implemented technique in today's applications – can perform very well in most cases but cannot cope with featureless environments. Under these conditions, the virtual maps of the physical spaces are often inaccurate, and this results in virtual objects not having fixed positions on the scene. This is a very relevant issue since AR experience heavily depends on having seamless ensembles of real and virtual information. The purpose of this research was to create a mobile application that generates mixed realities by using aspects of both marker-based and markerless AR, without relying on the device camera. The end goal of the project was to build a system that is able to virtually map any physical space, provided that a synchronisation marker and the environment dimensions are given. By reading the data collected by the Android sensors, the application tracks the AR-enabled device's position and backtracks the users' movements to correctly hold various virtual objects in place on the virtual scene.

Two different approaches were implemented as the hybrid AR solutions that attempted to solve the issues of markerless AR. While the former uses an array to store the various acceleration values that are continuously gathered by the sensors, the latter accumulates those values in a single variable that represents the cumulative acceleration of the system. In both the approaches, the acceleration samples were used to calculate velocity and displacement of the device through a two-step integration process. The outputs of these calculations were seen not to be accurate enough to provide a satisfactory AR experience. With the first approach, the application reacted to the changes in real-world displacement really slowly, and that caused the device's estimated position not to be updated properly. Although with the second solution the previous issue was solved, the new system heavily suffered from sensors drift which constantly drove the estimated parameters far away from their actual values.

The performance of both the approaches were analysed through several statistical tests in order to assess the accuracy of the solutions and to compare them with ARCore's markerless approach. While presenting the data collected through tests of the apps, the various issues with every approach were highlighted and explained accordingly. The

results showed that none of the two solutions performed satisfactorily when compared with the algorithms implemented in Google's Augmented Reality SDK. The estimated values in both cases could not be considered to be reliable enough to build immersive AR experiences. ARCore's tracking was also tested under unfavourable conditions and was confirmed to present issues when the environment does not contain enough features. The comparison between the featureless tests of the various approaches showed how the hybrid approach possesses some potential that further development of the technique might take advantage of.

One of the major improvements that could help the tracking system to be more reliable would be using multiple synchronisation markers instead of just one. By having several synchronisation points, the app would be able to correct the device's virtual position and therefore partially compensate for the accumulated drift and the noisiness of the sensors. Since the positions of these markers would be known to the system, it would be fairly simple to adjust the estimated displacement of the device on the fly.

Although the above-mentioned approach could correct the values drift, it would not completely solve the issue since the application would still need to deal with approximations and noise. An alternative method that could further improve the system's performance would consist in using three synchronisation markers to triangulate the device's position at any given time. By knowing the markers position relatively to the real-world space and by knowing the dimensions of the environment, it would be possible to estimate a relative displacement by just looking at the device's distance from each of the three markers. It is indeed mathematically possible to evaluate the 3D position of an object by knowing the distances between this object and three different fixed points in space.

Provided that the proposed improvements would satisfactorily increase the accuracy of the system, the app would ultimately represent a reliable alternative to the commonly used markerless AR techniques. The final version of the software could potentially be useful for several applications that require mapping of small and large spaces – from rooms interior design, to museums tourist guides – and that cannot cope with the constraint of having to avoid featureless environments.

References:

Chen, C., Chen, W., Peng, J., Cheng, B., Pan, T., Kuo, H. and Hu, M. (2017) 'A Real-Time Markerless Augmented Reality Framework Based on SLAM Technique', *2017 14th International Symposium on Pervasive Systems, Algorithms and Networks & 2017 11th International Conference on Frontier of Computer Science and Technology & 2017 Third International Symposium of Creative Computing*, pp.127-132.

Craig, A.B. (2013) *Understanding augmented reality concepts and applications*.

Google (2019) *ARCore Overview*. Available at: https://developers.google.com/ar/discover (Accessed 10 July 2020).

Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A. and Fitzgibbon, A. (2011) *KinectFusion: Realtime 3D reconstruction and Interaction Using a Moving Depth Camera*.

Katiyar, A., Kalra, K. and Garg, C. (2015) *Marker Based Augmented Reality*.

Kipper, G. and Rampolla, J. (2012) *Augmented reality an emerging technologies guide to AR*. Waltham, Mass.: Syngress.

Krueger, M., Gionfriddo T. and Hinrichsen, K. (1985) *Videoplace – an Artificial Reality*.

LePage, P. (2019) *Device Orientation & Motion*. Available at: https://developers.google.com/web/fundamentals/native-hardware/device-orientation (Accessed 12 August 2020).

Lu, D.L. (2019) *Ouster OS1-64 lidar point cloud*. Available at: https://commons.wikimedia.org/wiki/File:Ouster_OS1-64_lidar_point_cloud_of_intersection_of_Folsom_and_Dore_St,_San_Francisco.png (Accessed 24 August 2020).

Lu, F., Zhou, B., Zhang, Y. and Zhao, Q. (2018) 'Real-time 3D scene reconstruction with dynamically moving object using a single depth camera', *The Visual Computer; International Journal of Computer Graphics*, Vol.34(6), pp.753-763.

Peddie, J. (2017) *Augmented reality: where we will all live*.

Rusinkiewicz, S. and Levoy, M. (2001) 'Efficient variants of the ICP algorithm', *Proceedings Third International Conference on 3-D Digital Imaging and Modelling*, pp.145-152.

Schechter, S. (2019) *What is markerless Augmented Reality?*. Available at: https://www.marxentlabs.com/what-is-markerless-augmented-reality-dead-reckoning/ (Accessed 21 July 2020).

Stachniss, C., Leonard, J.J. and Thrun, S. (2016) 'Simultaneous Localization And Mapping', *Springer Handbook of Robotics*, pp.1153-1175.

Wearable Computer Lab (no date) *ARQuake: Interactive Outdoor Augmented Reality Collaboration System*. Available at: http://www.tinmith.net/arquake/ (Accessed 20 August 2020).

Wikipedia (2018) *Teleprompter Schematic*. Available at: https://commons.wikimedia.org/w/index.php?curid=334814 (Accessed 17 August 2020).

Appendices

Appendix A – Application Demo

The following video is a demo of the application that was implemented throughout the presented research. All three versions of the app are showcased in the video in both regular and featureless environments. This allowed to highlight the behaviour of the system under different conditions. The scenario in which the video was recorded is different to the one in which the tests were carried out, therefore the displacement values in the demo might slightly differ from the ones in the analysed datasets.

Link to YouTube:

https://www.youtube.com/watch?v=0m0kALdvdYM&feature=youtu.be

Appendix B – Tests Raw Data

The following tables contain the raw values of displacement (*X* coordinates) obtained while carrying out the various tests on the three versions of the hybrid AR application. Both the datasets gathered through experiments in regular and featureless environments are listed below.

Table 2: Regular tests raw data

AR Core Approach	Acceleration Array	Cumulative Acceleration
-1.446979E-002	-0.1832288	2.2839590
-9.631269E-003	-0.1798563	3.0286780
-8.294735E-003	-0.3913195	2.0647450
-5.523531E-002	-0.5840251	0.1868839
8.164726E-003	-0.5132332	-3.3122320
-1.985362E-002	-0.7467238	-3.5856310
4.588254E-005	-0.4990216	-3.9414070
-3.744815E-002	-0.6124700	-5.3856990
-7.509366E-004	-0.5909225	-7.095450
5.862154E-004	-0.7193773	-7.7091550
-2.354374E-002	-0.6032601	-11.567610
-1.924649E-003	-0.8780393	-11.847150

-9.215169E-003	-0.8070745	-12.860710
2.170442E-002	-0.8902201	-26.752150
-1.163497E-002	-0.7982216	-28.042060

Table 3: Featureless tests raw data

ARCore Approach	Acceleration Array	Cumulative Acceleration
0.2043266	-0.40381770	-1.3269820
0.6921126	-0.41477140	-9.246060
0.9245813	-0.025795570	-6.9277780
1.0840580	-0.022230019	-0.05711488
1.2263370	-0.1158930	2.4785230
1.4232920	-0.07089850	2.5388720
1.5210320	-0.14439020	8.2154180
1.6664840	-0.213859790	11.600340
1.8123600	-0.11369640	7.4096550
2.0299180	-0.20186080	3.8899470
2.1366590	-0.10891930	14.695550
2.2272950	-0.14434150	14.74880
2.2887180	-0.23028240	28.331970
2.3262140	-0.25303570	30.969140
2.3354220	-0.17559070	34.424750