



# An approach to multi-platform augmented reality development for mobile devices

A dissertation by Anna-Marie Richter

Student Number 432081

Submitted to the department of Creative Media and Game Technologies

Saxion University of Applied Sciences

Submitted June 2020

Saxion Supervisor: Mark Schipper

Company Supervisor: Terence Geldner, Lars Grotehenne

Demonstration of the final product:

[https://drive.google.com/file/d/1a4qgMudS6\\_pvenyl304-hQR9UAbfbyZ4o/view?usp=sharing](https://drive.google.com/file/d/1a4qgMudS6_pvenyl304-hQR9UAbfbyZ4o/view?usp=sharing)

## Abstract

With recent innovations in handheld mobile device technology, the capabilities of mobile augmented reality have taken a leap as well. Many companies are investing in this technology to optimize their internal and external processes. This research aims to provide insights into the relation between mobile technology advancements and AR capabilities. With this understanding it is possible to determine relevant optimization processes throughout the development stages to ensure a wide range of supported devices. Additionally the research explores possible solutions on how development can be adapted to a multi-platform strategy to speed up prototyping processes and reduce maintenance time. It furthermore enables the prediction of future trends in order to be able to make the right decisions in developing for this medium long term. It does so by specifying the crucial hardware factors and limitations supporting the AR experience and sets this in relation to recent feature advancements in the field of AR to give an indication for which devices deliver the best experience and how the potential of older devices can be maximized through development optimization. Based on the review of literature it has been found that both camera quality and computational processors are the crucial internal factors influencing the quality of AR experiences.

The finding clearly shows a correlation between tracking stability and camera quality and amount of computational processors. Based on the test results a general recommendation can be given to opt for devices with no less than a 12MP camera and at least a hexa-core processing unit to support an optimal AR experience. In a subsequent prototyping approach, external factors influencing the quality have been investigated. The research has shown that reducing the polygon and object count of virtual models relieves the stress on the CPU and supports a more stable tracking especially on lower end devices. Generally, the research has shown an advantage of iOS devices over Android devices.

This is due to Apple's recent release of iOS 13 and the new A13 bionic chip, enabling a more sophisticated set of features. Since Android devices are more diverse and do not receive such timely updates it stands to reason if Android devices will catch up to Apple's innovations.

## Acknowledgements

I would first and foremost like to thank my auntie for walking the pug and keeping it overall very well fed, happy and mentally stable throughout not only confinement but also this project.

I would furthermore like to thank my Saxion coach Mark Schipper for providing constant guidance, moral boosts, feedback.

I also thank my company supervisors Lars Grotehenne and Terence Geldner for welcoming me to IAV and accompanying me to the best of their abilities despite the challenging circumstances.

## **Table of Content**

<b>Index of abbreviations i</b>	<b>9</b>
<b>Table index ii</b>	<b>10</b>
<b>Image index iii</b>	<b>11</b>
<b>1. Introduction</b>	<b>12</b>
<b>2. Background</b>	<b>13</b>
<b>2.1 Company</b>	<b>13</b>
<b>2.2 Goal</b>	<b>13</b>
<b>2.3 Problem Definition</b>	<b>14</b>
<b>2.4 Scope</b>	<b>14</b>
<b>3. Methodology</b>	<b>15</b>
3.1 Literature Research	16
3.2 Design Thinking	16
3.3 Action Research	16
3.4 Business Readiness Rating Model	16
3.5 Prototyping	17
3.6 Source Reliability	17
3.7 Objectivity	17
<b>4. Theory</b>	<b>18</b>
4.1. Definition of Augmented Reality	18
4.2. Marker-vision based tracking	19
4.3. Markerless Vision Based AR	19
4.4 Hardware enabling markerless tracking	20
4.5 SLAM	21
4.6 Spatial Understanding	22
4.7 External factors affecting markerless tracking	22
<b>4.8. Cross Platform Development</b>	<b>22</b>
4.8.1 ARKit	23



4.8.2 ARCore	24
4.8.3 Multi-platform: AR Foundation	24
4.9 Summary of Theory	25
<b>5. Test results for iteration stages of the application</b>	<b>26</b>
5.1 Requirements Analysis	27
5.2 Test Results	27
5.2.1 Cross-Platform AR Frameworks	27
5.2.2 Plane Recognition and Tracking stability	28
5.2.3 Model Iteration	29
5.2.4 Screen Resolution Independent UI	30
5.2.5 Light Estimation	31
5.2.6 Occlusion	33
5.2.7 Transparent Light and Shadow Receiver Shader	34
<b>6. Discussion</b>	<b>34</b>
<b>7. Conclusion</b>	<b>35</b>
<b>Appendix I</b>	<b>37</b>
<b>Appendix II</b>	<b>39</b>
<b>Appendix III</b>	<b>41</b>
<b>Appendix IV</b>	<b>47</b>
<b>Appendix V</b>	<b>50</b>

## **Index of abbreviations i**

AR - Augmented Reality  
BRR - Business Readiness Rating Model  
COM - Concurrent Odometry and Mapping  
CPU - Central Processing Unit  
IMU - Inertial Measurement Unit  
ML - Machine Learning  
SDK - Software Development Kit  
SLAM - Simultaneous Localization and Mapping  
ToF - Time of Flight Camera  
UI - User Interface  
VIO - Visual Inertial Odometry

## Table index ii

[Table 1: Test devices](#)

[Table 2: Feature Availability and Supported Devices](#)

[Table 3: Use case specific device compatibility](#)

[Table 4: Student contribution chart](#)

## Image index iii

[Figure 1: Design Thinking Model](#)

[Figure 2: Action Research Model](#)

[Figure 3: Overview of Augmented Reality Cases](#)

[Figure 4: Example of marker based AR](#)

[Figure 5: Example of QR Code](#)

[Figure 6: Example of object recognition](#)

[Figure 7: Hardware components of mobile phones](#)

[Figure 8: Blob detection in SLAM](#)

[Figure 9: Corner detection in SLAM](#)

[Figure 10: Tracked landmarks in an image and their location in a mapped view](#)

[Figure 11: AR Feature Point Clustering](#)

[Figure 12: SLAM performed by ARKit, as demonstrated at WWDC 2018](#)

[Figure 13: Spatial mapping mesh covering a room](#)

[Figure 14: Worldwide interest in AR platforms](#)

[Figure 15: Unitys AR Ecosystem ARFoundation ARCore and ARKit](#)

[Figure 16: Image of Rect Transform Anchor Preset Settings.](#)

[Figure 17: Image of Canvas Scaler Settings](#)

[Figure 18: Final result Light Estimation on iPhone X in natural lighting](#)

[Figure 19: People Occlusion in ARKit3](#)

[Figure 21: Plane Occlusion Shader](#)

[Figure 22: Custom shader to render light and shadow on transparent geometry](#)

## 1. Introduction

Augmented Reality is a rapidly expanding field of technology that is currently being applied to a wide range of industries. It is already possible to shop and try out goods at home as Ikea proved in their IKEA Place App ([IKEA, 2019](#)) or combine health and fitness with popular augmented reality games such as Pokemon Go ([Niantec, 2016](#)). However augmented reality also becomes more prevalent in business and industry related application areas such as holding virtual conferences, heads up displays in cars and supporting the product work cycle in all areas from the initial design phase to the sales and aftersales process. AR allows to efficiently test different options of configurations such as colors, forms and models in the virtual space without requiring resources. As an example, with the help of augmented reality engineers at Mercedes Benz are working with a tool that allows to fit a conceptual engine into an existing chassis ([Schart, 2014](#)). A resource saving planning of production and processes are already achieved by the company Trumpf by fitting the virtual machinery into the real environment and simulating the flow of resources in AR ([Trumpf, n.d.](#)). For industrial partners in the development and after sales processes mobile augmented reality is a useful tool to demonstrate current development stages to internal and external stakeholders, communicate any impediments with graphical representation and collaborate on finding creative solutions. Mobile augmented reality has a great advantage in this case as opposed to other interactive virtual platforms such as AR headsets or virtual reality. The setup is minimal and the required device is easily portable to any office space or congress. Especially as an automotive IT service company focused on the after sales segment it is vital to stay up to date on the latest technologies to ensure clients receive cutting edge solutions and to not fall behind competitors. For this reason the current state and future trends of the augmented reality technology will be explored to ensure the company is equipped with the right expertise to refine, optimize and digitalize work processes with minimal resource investment.

In the past, image and object recognition approaches were an innovative choice to recognize and augment printed media or real life car models to create a customer experience in the sales segment.

Object recognition allows potential buyers to view the car in different colors or configurations or serve as an interactive manual to perform light maintenance tasks. The major drawback is however that an existing object is required to serve as a marker, rendering this approach useless for any conceptual or process focused operations.

In recent years the state of hardware and software has advanced greatly. Nowadays, new mobile phones are equipped with a range of high tech sensors and HD cameras. Through a combination of camera systems, dedicated sensors, and complex math it is possible to detect and map the real-world environment without relying on image markers or object markers. This allows to place virtual content freely in the world, enabling a more sophisticated set of features and content.

In this research paper the current state of this technology and its application potential is evaluated in regards to technological hard- and software innovations.

Due to the prevailing Covid-19 crisis user testing to verify and improve the app as a business case is not possible. To confirm the actual real life usability and effectiveness of an instructive manual application extensive user testing needs to be performed to verify best practices and iterate on human behaviour to test the app in regards to its contextual meaning.

## 2. Background

### 2.1 Company

As an independent company for the automotive and supply industry, IAV offers engineering expertise in automotive and IT, hardware and software, products and services since 1983. With a global workforce of more than 7500 employees they have been helping their business customers implement projects with cutting edge solutions in facilities all over the world ([IAV, 2020](#)). To ensure their clients receive the best fitting solution for complex projects, IAV utilizes the potential of state of the art technologies such as AI and big data as well as virtualization and automation technologies. Following their principles of innovation they are now in pursuit of realizing projects in the field of augmented and virtual reality. To be up to industry standards and on par with the latest trends and practices, they have assigned a bachelor thesis project with the goals of researching and applying this technology. As an IT Service Company settled in the after sales segment, IAV requires an augmented reality application that showcases recent technological trends and innovations in augmented reality and how they can be used in the context of the automotive industry. The goal is to create a showcase demonstrating the newest features to convince both external and internal stakeholders of the technology, as well as provide a foundation of knowledge about recent innovations and how they are related to hardware prerequisites. Previous approaches include experimenting with the marker and object based tracking of the Vuforia SDK. However, since then marker-less approaches based on surface detection have diversified the possibilities greatly.

### 2.2 Goal

With recent advancements in technology a new multi platform oriented development process has been introduced. The application that is being developed should be deployable to a wide range of devices both iOS and Android systems to guarantee flexibility in their application. Therefore the focus of the practical analysis and development will include an approach to a cross platform development strategy that ensures a unified behaviour throughout all devices capable of AR. The application that is being developed benefits the company in the area of showcasing prototypes and their features on the example of a virtual car manual. The app is based on the markerless visual tracking method and uses the latest innovations in the field of augmented reality in the context of the automotive industry. A feature point recognition approach allows to place the example vehicle freely in the world. The optimization and development processes are confirmed through individual in depth feature test sessions. A user can potentially use the app to better understand certain features and functionalities of the car with the help of augmented reality. The educational efficiency has to be verified by separate user testing which is not part of the scope of this thesis.

## 2.3 Problem Definition

How can an automotive IT solutions company utilize recent technological advancements in the context of mobile augmented reality to create a showcase application for both internal and external stakeholders to demonstrate prototypes and their features that requires minimal iteration time for multi platform deployment and supports a wide range of devices? In order to answer the main question the following aspects will be discussed:

- How are hard- and software components in mobile devices influencing the AR experience?
- What are common issues in cross-platform development and how can they be solved?
- Which relevant features have been enabled through recent technological advancements and how can they be integrated into a multi-platform project?

The first question is required to estimate which devices will support the newest features and generate a basic understanding of the principles of how augmented reality operates in order to be able to make educated decisions on the optimization processes and development choices later on. It furthermore provides the basis for an outlook on potential future developments of the technology and the devices required to support this. The second question relates to the development approach when creating applications for a range of devices and platforms. It entails an analysis of current frameworks supporting cross-platform AR development on the market. It furthermore highlights the most common challenges in cross platform development and how they can be solved within the framework. The third question gives an overview of the features available and potential future features based on the findings of question 1. Current features are tested and analyzed in regards to their compatibility with a cross platform solution.

## 2.4 Scope

The paper will focus on the technical prerequisites and recent advancements in the field of mobile augmented reality using the markerless feature point detection technology. Marker based solutions are not subject to this research. Due to the current Covid-19 circumstances this thesis will focus only on the technical aspects and recent advancements in the field of mobile augmented reality. This approach ensures that testing does not require any other people and can be facilitated by the student instead. Subject of discussion will be AR frameworks, optimization processes when handling cross platform development and AR features. The prototype has to be created in Unity3D. The supporting AR framework will be chosen based on the research results. The prototype has the purpose of supporting and demonstrating the technical findings of this research. It does not attempt to provide a finished user experience solution. The actual content of the app is not subject to this research. To confirm the usability of an augmented user manual additional user research in regards to the content and UI structure would have to be performed. In the current situation this is not possible. For this project the multi-platform approach only relates to iOS and Android devices.

The testing is limited to the devices provided by the company. The following devices are used for all test procedures:

Device	Released	Processor	Cores	RAM	Screen Resolution	Camera
iPhone 7	2016	Apple A10X Fusion	4	2GB	750 x 1334 pixels, 16:9 ratio (~326 ppi density)	12 MP
iPhone X	2017	Apple A11 Bionic	6	3GB	1125 x 2436 pixels, 19.5:9 ratio (~458 ppi density)	12 MP
iPad Pro	2017	Apple A10X Fusion	6	4GB	1668 x 2224 pixels, 4:3 ratio (~265 ppi density)	12 MP
Samsung Galaxy Tab 3	2013	Intel Atom	2	1GB	600 x 1024 pixels, 16:9 ratio (~170 ppi density)	3.15 MP

*Table 1:* Table of test devices

Because of these limitations, the most recent AR features will not be included in the testing or the final prototype and instead only be mentioned in theory as they are not supported on any of these devices.

There is no budget for development or testing. All development and testing was done in home office.

### 3. Methodology

In this chapter the methodologies used in the project will be presented and motivated. The main methodologies used are Design Thinking, Action Research, literature research and the Business Readiness Rating Model.

#### 3.1 Literature Research

Literature research was done in a desk research approach by analyzing articles and publications found on the internet. The following keywords were used:



**Keywords AR Theory:** Mobile Augmented Reality, types of mobile augmented reality, SLAM, COM, plane tracking, image tracking, AR supported devices, hardware of mobile devices

**Keywords ARFrameworks:** multi platform development approaches, cross platform AR frameworks

**Keywords Cross platform development:** Unity UI optimization, optimizing AR for mobile devices, AR best practices

In order to identify reliable sources only information from trustable websites such as Unity's official documentation or published research papers have been considered.

### 3.2 Design Thinking

The project roughly follows the design thinking approach (*Figure 1: Design Thinking Model* ([Kreativtechniken.info, n.d.](http://Kreativtechniken.info))). Especially during the initial idea finding phase the empathize and define process were used to identify possible projects with the client.

Through empathy maps the needs of the client were analyzed to identify the issues that can be solved by research. Together with the client the process of defining and ideating went through multiple iterations until the final concept was established. The prototyping and testing cycle has been used separately for each individual development subject.

### 3.3 Action Research

Action Research in combination with the Design Thinking framework were the main research methods used throughout the project. Action research is a philosophy and methodology of research generally applied in the social sciences. It attempts transformative change through the simultaneous process of taking action and doing research, which are linked together by critical reflection ([Research-Methodology, n.d.](#)). The general model followed is illustrated by *Figure 2 Action Research Model*. ([Research-Methodology, n.d.](#)). The issue was first analyzed. Then possible solutions were identified through desk and literature research and the findings were applied in development, followed by testing and reflection on their usability. This process was repeated until the desired outcome was achieved.

### 3.4 Business Readiness Rating Model

The evaluation of the augmented reality frameworks follows the Business Readiness Rating Model (BRR) which is considered an open standard for the evaluation of open source frameworks but can also be used for the comparison of proprietary software. The model is divided into 4 phases in which the separate software framework components are gradually evaluated.

#### 1) Phase 1: Quick Assessment Filter:

Definition and application of the criteria on the established list of all possible software products previously collected for a first pre-selection of frameworks ([SpikeSource, Intel Corporation, 2005](#))

#### 2) Phase 2: Target Usage Assessment:

Weighting and prioritization of the 12 categories and their metrics on which the evaluation of the frameworks happen ([SpikeSource, Intel Corporation, 2005](#))

### **3) Phase 3: Data Collection & Processing:**

The normalized metrics are now applied to the previously collected data and calculated against the weighting factors of the individual metrics. By default a scale from 1-5 is used for this in which 1 is defined as not acceptable and 5 is defined as outstanding. [\(SpikeSource, Intel Corporation, 2005\)](#)

### **4) Phase 4: Data Translation:**

From the sum of evaluations for the single metrics an overall score is created per category. Finally, these category ratings are accounted for with the weighting of the single categories and in a decision matrix the BRR point score is determined for each software product. [\(SpikeSource, Intel Corporation, 2005\)](#)

For more information please refer to the [Test Report Chapter 2: Cross Platform Development Frameworks](#).

## **3.5 Prototyping**

Multiple prototypes were produced and tested in Unity3D in order to validate the functionality of the method identified in the desk research. UI prototyping has been done through paper sketches first, followed by prototyping in Unity. All features were tested and adapted through individual separate prototyping before combining them to a final product. Separately tested features are AR functionalities, UI optimization, Model optimization and AR tracking.

## **3.6 Source Reliability**

The used sources have been evaluated critically based on their reliability. The authors' credibility (degree, relevant career etc.), the year of publication as well as where the information was published has been considered. In uncertain cases the information has been compared to other sources. If a consensus was reached between multiple authors the source has been deemed credible. The sources used for technical solutions were mostly drawn from the official documentation of the framework.

## **3.7 Objectivity**

Objectivity of the results is guaranteed by verifying each feature through individual tests and adapting the prototype based on the results. Each feature is graded on established measurable metrics. The description of the chosen method together with the prototype makes it possible to recreate the test results for each feature individually. It should be noted however that the results of augmented reality feature testing vary heavily with the test environment conditions and might not yield the same results if tested in a different environment.

## **4. Theory**

To ensure the concept of augmented reality is clear, firstly a brief introduction on the subject will be given. Then the relation between hard- and software components of mobile devices are analyzed to give an indication on what influences the quality of tracking in AR space, what enables a persistent experience and how it can be improved. To understand choices in the development process the underlying technology of markerless AR tracking is explained. In a more practical approach the different AR frameworks currently on the market are tested and

analyzed based on the BRR model ([SpikeSource, Intel Corporation, 2005](#)). The different approaches and challenges to AR cross platform development will be explained, with special focus on how Unity's ARFoundation can help to solve some of the issues as the chosen development framework identified through BRR testing (for test methods and results please refer to the [Test Report Chapter 2: Cross Platform Development Frameworks](#)). The influence of hardware and environmental circumstances on the quality of AR experiences are tested, analyzed and evaluated within the chosen AR framework to give a recommendation on the devices that allow for a persistent experience and the supporting conditions. The identified solutions to AR cross platform challenges are then tested on a range of devices to ensure their accurateness. Through independent, separate test sessions, basic features of ARFoundation are evaluated for both iOS and Android devices in regards to performance and usability in the context and the findings applied to the final product. In a general conclusion all steps taken will be reviewed and their suitability justified by the test results.

#### 4.1. Definition of Augmented Reality

Augmented reality is the extension of the perception of reality with computer generated images like text information, images or 3D models, visualized through digital devices like mobile phones, tablets or AR glasses. Other than in virtual reality, AR does not replace the complete reality, rather it enhances it with additional information. Ideally, the virtual and real world blend together seamlessly and provide a reality with more information. *Figure 3: Overview of Augmented Reality Cases* by P. Milgram ([Milgram, P. 1995, December 21](#)) illustrates the relation between AR to the real and the virtual world. AR is closest to the real world for it is only enhanced with information. Virtual reality is completely computer-generated. In [1997 Ronald T. Azuma](#) formulated a common definition for AR. According to him augmented reality is a realtime interactive, three dimensional copy of reality that has been enriched with enhancing, artificial content. Azuma suggests the following 3 foundational characteristics to define an AR system:

- Combination of real and virtual
- Interaction in real time
- 3 dimensional relation between real and virtual objects

The requirements of a realtime interaction is essential for identifying AR applications. Often the content is realized in a three dimensional way. To enable a persistence in realtime, the object's position in relation to the position in the real world needs to be tracked. This concept will be defined in more detail in the following chapters. In vision based mobile augmented reality it is distinguished between two common types:

- Marker-vision based
- Markerless vision based

In both types the mobile device is trying to map the environment using computer vision, while at the same time trying to match what it sees with what it has seen in the past, so it can tell where it is.

By analyzing the video feed, software is able to find several kinds of visual features in the scene that become the foundation for it to build up spatial awareness.

## 4.2. Marker-vision based tracking

An AR system can be trained to detect specific images or 3D objects. This method is called marker based AR (*Figure 4: Example of marker based AR* ([Salah-ddine, 2019](#))). In marker based augmented reality, the position and orientation of virtual objects is defined by a physical marker, also called fiducial marker in the real world. This can be a picture or template system in which the object's position is recognized by matching the pattern of the acquired image with a pre-stored template or an ID encoded system such as QR codes (*Figure 5: Example of QR Code*. ([QR Code Generator](#))) that are identified through a decoding algorithm. During this process the visual “tracker” of the application is looking for these predefined markers and places the virtual objects on top of it. The tracking also works with 3D markers known as object tracking (*Figure 6: Example of object recognition* ([Wikitude, 2019](#))). The process of recognizing a predefined image in the world does not require elaborate hardware since the device does not need to know its position in space but instead is only matching the camera feed with the predefined images.

Nor does it require a high quality camera if the marker provides enough contrast. In contrast to the much more complex markerless approach this allows to draw more complex models while still maintaining a stable experience. However, there are major drawbacks to the marker based approach: Even the slightest occlusion of the markers causes the tracking to interrupt. Markers are not very convenient for some use cases as they restrict the range of motion heavily. Additionally, the augmented content is not unlimitedly scalable.

## 4.3. Markerless Vision Based AR

Markerless tracking requires a much more sophisticated approach. The device needs to be aware of its location in space, its orientation and movement and its relation to the environment. As opposed to the aforementioned marker based approach, markerless tracking avoids the need of having to prepare the environment with fiducial markers beforehand and allows the user to move freely in a room. This expands the applicability range greatly. Through image processing algorithms and calculations feature points that occur in the environment are detected. They provide the data required to determine position and orientation of the device ([Ziegler, 2010](#)). This is achieved through spatial computing. Spatial computing is enabled through a device called Inertial Measurement Unit. The IMU consists of the accelerometer, gyroscope and magnetometer and enables a prediction of the orientation and location of the phone ([Mourcou, 2015](#)). In most cases however it is necessary to combine the IMU data with other sensors to mitigate noise that leads to inaccuracy. For example by combining GPS data with compass data. The following phone sensors and components play a role in AR tracking:

#### 4.4 Hardware enabling markerless tracking

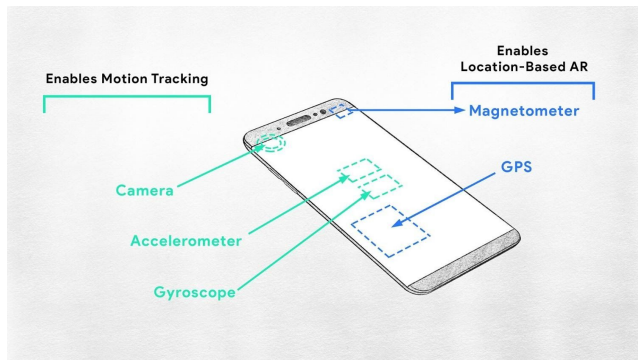


Figure 7: Hardware components of mobile phones ([Chopra, 2018](#))

**Accelerometer:** Measures acceleration, which is speed divided by time. It is the measure of change in velocity and required to enable the tracking of the device's motion.

**Gyroscope:** Measures and/or maintains orientation and angular velocity. When changing the rotation of the phone while using an AR experience, the gyroscope measures that rotation and thus ensures that the digital assets respond correctly.

**Magnetometer:** Gives phones a simple orientation related to the Earth's magnetic field. This device is key to location-based AR apps.

The data from the following sensors can be used:

**Phone Camera:** The camera supplies a live feed of the surrounding real world upon which AR content is overlaid.

**True Depth Sensors/ToF:** A ToF camera uses infrared light to determine depth information. The sensor emits a light signal, which hits the subject and returns to the sensor. The time it takes to bounce back is then measured and provides depth-mapping capabilities. ([Samsung, n.d.](#))

ToF sensors also enable to estimate the direction of where the light is coming from.

**GPS:** The GPS receiver in phones receives geolocation and time information from the global navigation satellite system.

**CPU/GPU:** The power of camera processing is closely related to the CPU power of a device. In addition to the camera, phones rely on complex visual processing technologies like machine learning and computer vision to produce high-quality images and spatial maps for mobile AR.

([Grossi, 2019](#); [Chopra, 2018](#); [Prof. Daponte, n.d](#))

A process called Sensor Fusion uses the data from these sensors mentioned above to predict where the IMU should be based on the current measurement and the previous measurement. But even with this efficient filtering process the result is still not accurate enough to support sophisticated markerless augmented reality. It is necessary to periodically correct the predictions and IMU based tracking with another measurement, a second opinion to ensure a reliable estimation. This process is called Simultaneous Localization and Mapping (SLAM) ([Patterson, 2017](#)).

## 4.5 SLAM

According to Andreas Jakl ([\[Jakl, 2018\]](#)), the SLAM algorithm has two aims:

1. **Build a map** of the environment based on 2D camera data and motion sensors
2. **Locate the device** within that environment

The set of SLAM algorithms calculate the device's exact position through the spatial relationship between itself and multiple feature points in order to map and track the environment. Feature points are visually distinct features and are used to compute the device's change in location. The visual information is combined with measurements from the IMU to estimate the pose (position and orientation) of the camera relative to the world over time. It does this based on the Kalman filter principles which are used to achieve accuracy in cases where an exact value or outcome cannot be measured ([\[Google, n.d.\]](#)).

According to Professor Daponte (Prof. Daponte, n.d.) from the university of Sannio the following feature points can be detected through the SLAM approach.

### Corner detection:

Algorithms for searching points that have maximum curvature  
Algorithms for identifying the intersection points of edge segments

### Blob detection

Region of an image in which some properties are constant or vary within a prescribed range of values



Figure 8 + 9: Blob detection in SLAM and Corner detection in SLAM ([\[Prof. Daponte, n.d.\]](#))

When working with the markerless approach, it is therefore vital that enough of these feature points are present. *Figure 10: Tracked landmarks in an image and their location in a mapped view from the MonoSLAM algorithm by [Davison\(2007\)](#)* showcases what should be achieved: Tracked feature points, their relation in space as well as the inferred camera position. The feature points created through SLAM are scanned for clusters that appear to lie on common horizontal or vertical surfaces like tables or walls and thus make the surface available to the application as planes (*Figure 11: AR Feature Point Clustering.* ([\[Mukherjee, 2018\]](#))). As the computer vision system is analyzing each video frame and trying to identify feature points in it, it is simultaneously matching what it finds to what it has found in previous frames. Using this data it is possible to anchor AR objects onto identified feature points or planes the SLAM system is tracking. By anchoring the virtual object the object will stay in its position relative



to the real world even as the user moves their device.

#### 4.6 Spatial Understanding

At the moment mobile AR frameworks rely on tracking simple planes due to the processing power limitations of mobile devices. AR wearables such as the HoloLens, using true depth sensors already try to infer more knowledge through spatial understanding. Time of flight cameras, also known as depth cameras map out the surroundings, creating a basic three-dimensional representation of what is in front of them (*Figure 13: Spatial mapping mesh covering a room* ([Microsoft, 2018](#))). This allows for digital objects being occluded by real world objects and creates a more immersive experience.

#### 4.7 External factors affecting markerless tracking

World tracking requires a high degree of accuracy to create realistic AR experiences. It is dependent on details of the device's physical environment that are not always consistent or are difficult to measure in real time without some degree of error. The tracking of natural features presents several challenges that impact the outcome. [Yudiantika \(2015\)](#) observed several of these factors that affected the success of object tracking in an AR application:

- **Shape and texture of the real world surface:** Tracking is easier when an object presents a unique shape and texture.
- **Color of the object:** The background color of the object determines the contrast between the object and the rest of the environment. Tracking is facilitated when there is a greater contrast between the two.
- **Room lighting:** The intensity of the light illuminating will affect the markerless tracking since the camera needs to properly capture the specific features of the objects and environment.
- **Light reflection:** light reflections can interfere with the tracking.
- **Type and position of the lights:** natural light, incandescent light (bulb) etc.

All of these external factors affect the amount of feature points that can be detected by the system as mentioned previously and thus directly influence the quality of tracking.

#### 4.8. Cross Platform Development

A cross platform development generally means that the software should run on multiple different target platforms such as iOS, Android or Windows devices. The benefit of cross platform development is that a piece of software only has to be developed once and can be deployed to different target platforms without additional development time. According to [Alcala Toca \(2011\)](#) there are different types of frameworks that support this kind of cross platform development. There are native frameworks which provide connection points, meaning native functions for the respective device. This can on one hand be enabled through native containers. On the other hand the source code can be executed with a natively compiled code by an interpreter. Alternatively the source code can be converted into native code or is simply developed in a platform independent native programming language. In this thesis the definition of a cross platform will be identified through a single common source code for AR applications, also called a shared code basis.

Instead of multiple source codes for the corresponding platform, with a shared code it is possible to only write a single source code once and deploy it on multiple devices. As a result developing for different platforms requires less time, resources and a faster publication. Likewise the maintenance costs are reduced considerably. Through the distribution on multiple platforms more customers can be reached. In the Test Report [Chapter 2 Cross Platform Development Frameworks](#) different frameworks that integrate into the Unity engine and support a common code cross platform development are analyzed and evaluated on the requirements of the framework on this specific use case based on the Business Readiness Rating Model. On the basis of the evaluation results ARFoundation has been decided to be used to develop the application. ARFoundation is integrated into the Unity license and can be used without additional charge. Throughout all test criteria it provides constant good to very good results. Especially the amount of supported features and the excellent documentation of such have led to the decision to use this framework in the project. ARFoundation receives frequent feature updates and bug fixes, ensuring its viability presumably even for future projects. The big community of Unity provides a lot of tutorials and instructions on how to achieve the best results, making this platform very beginner friendly. In the following ARFoundation and its development process will be explained further.

Unity's AR Foundation provides a layer of abstraction to the open source SDKs ARCore and ARKit. In 2017, Apple and Google introduced two competitive application programming interfaces, supporting the creation of augmented reality applications for mobile devices: ARKit (September 19, 2017) and ARCore (March 1st 2018). Besides the fact that ARCore and ARKit are free of charge, they offer an abundance of features which were previously only available in commercial versions of competitive SDKs. Even in early stages ARCore and ARKit already caught attention on the market as the *Figure 14: Worldwide interest in AR platforms* ([Google Inc., 2018](#)) suggests and are since then popular development choices.

#### 4.8.1 ARKit

As mentioned previously, real advancements happened when Apple improved their processing power first with the A9 Bionic processors. In 2017 in the subsequent announcement of iOS 11 Apple first introduced ARKit, a new framework that allows developers to easily create augmented reality experiences for iPhone and iPad. Included in the release was Core ML which enables developers to create smarter apps with powerful machine learning that predict, learn and become more intelligent. Designed for iOS, this new framework for machine learning lets all processing happen locally on-device ([Apple, 2017](#)). With the release of iOS 13 and iPadOS 13 Apple is undoubtedly the leading force in mobile AR development, introducing innovative features such as Motion Capture, Realtime People Occlusion and Face Tracking. With ARKit it is only possible to develop for iOS devices. More specifically, iPhones starting with iPhone 6s and iPads starting with iPad Pro. In order to use the newest features however a device with at least an A12 Bionic chip is required, starting with the iPhone XS. Currently, only the front facing camera features a true depth sensor in order to support the facial recognition. Apple is expected to release a rear facing true depth sensor camera in upcoming phones however, enabling spatial awareness similar to the Hololens.



## 4.8.2 ARCore

ARCore is Google's answer to Apple's ARKit. It was released in early 2018 ([Google Inc., n.d.](#)). At first, ARCore was primarily focused on Android as the main platform for creating AR experiences.

However, over the last two years ARCore has expanded to also provide several APIs that allow to create AR experiences for iOS as well. Google's ARCore provides motion tracking, environment understanding, and light estimation. The platform supports devices running Android 7.0 or later and iOS 11.0 or later ([Google Inc., n.d.](#)). All of these features are equally present in ARKit. ARCore is however missing many more advanced technological features presented in ARKit such as people occlusion. This is due to the fact that Android devices differ greatly in their computational power and supported features have to be selected carefully in order to be deployable on a wide range of devices.

## 4.8.3 Multi-platform: AR Foundation

ARFoundation tries to solve this problem by allowing developers to create AR apps that work on the widest possible range of devices by creating a common API and a set of AR components that work in conjunction with either AR Core or ARKit. Because ARFoundation is built on the core AR capabilities common to both platforms, it is based on the most stable or solid AR capabilities. It however also allows access to native ARCore and ARKit features directly via their respective Unity packages. This makes it possible to use features from both ARCore and ARKit in the same project. Features that are native to ARKit will however still only function on a supported iPhone. As *figure 15: Unity's Ecosystem Foundation, ARCore and ARKit* illustrates, as opposed to the native SDKs AR Foundation wraps ARKit and ARCore's low-level APIs into a cohesive framework. This way ARFoundation can communicate with multi-platform APIs in Unity without the need to know whether it is communicating with ARCore or ARKit. This allows for additional Unity specific utilities such as AR session lifecycle management and the shader graph. The framework supports devices running Android 7.0 or later and iOS 11.0 or later ([Unity. n.d.](#)).

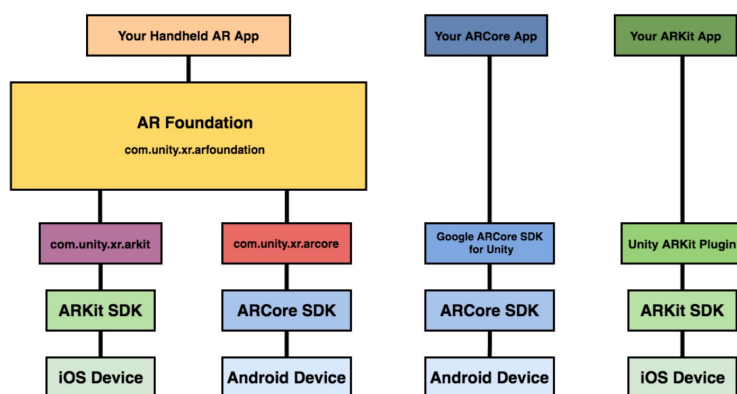


Figure 15: Unity's AR Ecosystem ARFoundation ARCore and ARKit. ([Unity. n.d.](#))

In case a feature is only available for one platform, ARFoundation will add special hooks on the objects. As soon as it becomes available, only the packages need to be updated instead of rebuilding the app entirely. This simplifies the development process tremendously.

## 4.9 Summary of Theory

While marker based tracking requires only the evaluation of the camera feed, the findings clearly suggest that in a markerless tracking approach the quality of tracking AR imagery is not only based on the camera quality but involves complex algorithms that require high amounts of computational resources. This means that the chosen AR method should be considered carefully for each use case.

A marker based approach might be restricted in its action circle, it does support a high range of devices regardless of their computational power however.

The current state of technology requires a fine balance between precision and efficiency.

According to [Ziegler \(2010\)](#): “On the one hand, the more information the application gathers and uses, the more precise is the tracking. On the other hand, the fewer information the calculations have to consider, the more efficient is the tracking. Efficiency is a huge issue for tracking on mobile devices. The available resources are very limited and the tracking cannot even use all of them, as the rest of the application needs processing power too.” Ziegler’s thesis also gives an explanation as to why markerless AR cannot be recommended to be used on all mobile phones even though they are technically able to perform SLAM. While inertial measurement units and sensors work similarly in both old and new devices, the hardware that has experienced the greatest improvements over the last years are the cameras as well as the computational processing units. These are the critical internal factors affecting the AR experience. On one hand a high quality camera will yield more trackable feature points and a more accurate image of the environment. On the other hand more tracking points also mean higher processing power is required to perform the complex calculations. Because Apple and Google are using VIO/COM approaches, the expectations on the camera quality are not as high. A standard single RGB camera is enough to match the criteria. Any gaps resulting from poor image quality can be balanced by IMUs. Most of the work is hence done by the CPU and algorithms. To determine exactly how much CPU and GPU power and camera quality is required to provide a degree of tracking that is industry ready, multiple tests on a range of devices are facilitated in the following iteration stages. This is done by comparing the provided test devices and determining their tracking quality in relation to their hardware based on a set of defined criteria and generalizing the results to give an indication in what CPU and camera range a device can be considered appropriate (see [Appendix II, table 3: Use case specific device recommendations for results](#)). Next to the high usage of computational resources for the tracking of the environment, the 3D content has to be rendered on screen as well. This suggests that in order to save computing space for the tracking, the model should be highly optimized to take up as little resources as possible. For application in the industry the model however still has to provide enough detail to convey important design decisions for example. Another part of the testing will therefore be to what degree a model has to be optimized in terms of polygon and object count in order to still enable smooth tracking on a range of devices. As stated in chapter 4.7 *Factors affecting markerless tracking* critical external factors that enable reliable and accurate tracking experiences are the environment texture and lighting conditions. The specific usage conditions for this application are not specified. In order to give a clear recommendation, the quality of tracking should be tested in different

lighting conditions and surfaces to determine how the ARFoundation framework reacts to them. The goal of testing is to give an indication of how much the lighting and presence of feature points in the environment influence the tracking quality. ARFoundation provides a base for creating a common shared code application that can be deployed to both iOS and Android. When it comes to more complex features it is clear that iOS provides more advanced solutions because of their advanced A12 hexa-core processing units, as chapter 4.8 suggests. It is up to testing to determine in what way a multi-platform AR framework can unify the opposing system capabilities of iOS and Android.

## 5. Test results for iteration stages of the application

The app being developed showcases the recent set of AR features and technologies in the context of the automotive industry. Relevant focus is on realistic depiction of the 3D object and smooth reliable tracking experience on the widest range of devices possible. As a potential use case the app should be constructed as a digital user manual, visually supporting small repair and service actions the user might want to facilitate on their car. When the user opens the app he is firstly introduced to the AR space through a tutorial UI to guide him through the scanning and object placement process. Then the main menu is presented, representing the main indexes of the car manual. As an example for this app the user can decide between the options “Repair”, “Service” and “Features”. When selecting a point a sub menu displays the different actions available while at the same time highlighting the action areas on the car model in AR space. This way the user already gets an indication of where the action should be performed.

When selecting a hotspot or sub-menu point, more in depth information on how to perform the repair or maintenance task is displayed in a 2D UI. The features and technical requirements are developed, tested and documented separately as the Test Report suggests before integrating them into the final app. This is to ensure they comply with the requirements of the main research question. In order to answer the question of how a multi platform approach may benefit the development workflow the testing and iteration was divided into different steps relating to each research question:

- Different AR cross platform frameworks were analyzed and tested in regards to their suitability for this use case with the help of the BRR model
- Compatibility and tracking performance of multiple devices was tested to give an indication of the device suitability for markerless AR use cases
- Different degrees of model optimization were tested in regards to how it affects the tracking quality on different devices
- A multi platform UI support framework is established and tested on different devices
- The framework is tested on its ability to provide a unified solution to the highly diverging prerequisites and feature availabilities of the iOS and Android platform
- Relevant features including Light Estimation, Realtime Reflections, Occlusion and Shader support for different rendering pipelines are tested in regards to their multi-platform support

In the following, the findings of the separate testing steps will be presented.

[Appendix IV](#) provides a detailed overview of the student’s contributions to the project.

## 5.1 Requirements Analysis

Generally, it is advisable to establish the requirements the piece of software should fulfill. Even though basic functional requirements have been established, the testing and iteration only focuses on the technical system requirements. The requirements analysis can be found in the [Test Report Appendix I Requirements Traceability Matrix](#). Based on the established system requirements, the following test iterations have been performed and evaluated:

## 5.2 Test Results

### 5.2.1 Cross-Platform AR Frameworks

In the [Test Report Chapter 2 Cross Platform Development Frameworks](#) different frameworks that integrate into the Unity engine and support a common code cross platform development are analyzed and evaluated on the requirements of the framework on this specific use case based on the Business Readiness Rating Model. In the scope of this evaluation 55 augmented reality SDKs have been found of which 6 could potentially be suitable for a markerless application in a business environment.

On the basis of the evaluation results ARFoundation has been decided to be used to develop the application. Throughout all test criteria it provides constant good to very good results. Especially the amount of supported features and the excellent documentation of such have led to the decision to use this framework in the project. Since ARFoundation provides a layer of abstraction to the individual APIs ARKit and ARCore the functionalities from both APIs can be integrated into the project. While ARFoundation allows to even integrate the most recent innovative features introduced with ARKit 3, as discovered in the development process at least an iPhone XS is required to test these features. Furthermore it has been discovered that even with ARFoundation as an abstraction layer, Android devices still did not yield convincing results and seemingly could not integrate as well as iOS devices. While the application was deployable and running with the basic set of features such as placing AR content, the more advanced features light estimation and reflection probes did not function.

### 5.2.2 Plane Recognition and Tracking stability

The main component in markerless vision based tracking is surface detection. Well-mapped surfaces allow for realistic placement of virtual objects in real space. The quality of the surface detection translates to the quality of the entire application. Content placed in AR space should stay persistent at all times. The following set of test iterations focus on these aspects.

In a first approach ARFoundations ability of recognizing planes was tested on all devices to determine the device's suitability for markerless AR use cases. The test device's technical specifications can be found in the [Test Report Chapter 1.2.1 Test Devices Figure 1: Test device comparison table](#). The following points were subject of observation:

- Accuracy of planes detected in relation to the total surface area
- Working in various lighting conditions
  - Impact of brightness and color of light on virtual objects daylight, incandescent light (60 W bulb)

- Working in unfavourable conditions
  - Impact of a shape on plane mapping: flat surface, single-colored, devoid of pattern and texture such as plane tables,  
flat surface with a pattern such as wooden tables

The detailed test results can be found in the [Test Report Chapter 3: ARFoundation Tracking and Persistence](#).

## Results

The accuracy of tracking stayed above 90% for the more recent iOS devices regardless of the lighting condition as long as a structured surface was present. The older Samsung Galaxy Tab 3 device showed major difficulties in any condition except bright natural light and a highly structured surface, resulting in an average success rate of only 52%. All phones failed to return any results on unstructured surfaces because no feature points could be detected. The general recommendations stated in Chapter 4.7 Factors affecting markerless tracking can be confirmed.

In a second test iteration the model persistency in AR space was tested. As discussed previously, in order to track the environment the device needs points in space to orient itself. The points are called feature points. In ARFoundation these feature points are either mapped to create a plane or they can be used as anchor points ([Unity, n.d](#)). To determine the accuracy of these trackables, testing has been facilitated for placing an object on a plane versus using anchor points to anchor an object to a point to estimate which approach yields a more accurate tracking (see [Test Report Chapter 3: ARFoundation Tracking and Persistence](#)). The following points were subject of observation:

- Positioning of 3D content on planes
- Working in various lighting conditions
  - Impact of brightness and color of light on virtual objects [daylight, incandescent light (60 W bulb)]
- Working in unfavourable conditions
  - Impact of a shape on plane mapping: flat surface, single-colored, devoid of pattern and texture such as plane tables,  
flat surface with a pattern such as wooden tables
  - Work in motion (rapid movement, different angles, losing track of the object)

During the testing it was measured how stable the model stays in place when moving the device.

## Results

As can be seen from the [Test Report Chapter 3.2: ARFoundation Object Persistence](#), no real difference could be detected between the accuracy of anchors and planes. Creating anchor points that are not anchored to planes is not usually recommended because they are resource intensive according to the Unity documentation ([Unity, n.d](#)). Therefore in order to save computational resources the object is simply anchored to the plane without the use of anchor reference points. The approach should be evaluated based on the use case. If only a single object should be displayed the use of anchors can be an option. In case of tracking multiple

objects using anchor reference points would take up too much computational resources: The lowest tested device was the Samsung Galaxy Tab 3 with only a dual-core processor, 1GB of RAM and a 3.5MP camera ([Test Report Chapter 1.2.1 Test Devices Figure 1: Test device comparison chart](#)). The tablet did not yield convincing results in any of the tests. The lowest tested iPhone was the iPhone 7 which already featured a quad-core processing unit and 12MP camera ([GSMArena, n.d.](#)). The device still showed minor difficulties in tracking objects with a higher polygon count. Especially in graphically demanding showcases more processing power is required. Both iPad Pro and iPhone X yield the best results. In all test cases they provided an optimal stable tracking of the model, even in unfavourable lighting conditions. Both devices show similar hardware specifications: Hexa-core processor, 3-4GB of RAM and a 12MP camera. Out of all test devices they were the most technically advanced models. It can be confirmed that even though the camera quality was the same for all Apple devices the models with more processor power yielded the best results. As reflected by the plane and model tracking testing in the test report chapter 3 and 4 devices with a camera quality below 12MP failed to recognize a sufficient amount of feature points in the environment to support the tracking. Similarly, devices that had only dual core processors failed the tests as well.

### 5.2.3 Model Iteration

Model optimization plays an important role in mobile development. Especially in markerless tracking the model needs to be optimized in regards to its object and polygon count to a degree that ensures smooth, easy rendering without requiring too much computational resources which would impact the tracking quality. In a series of tests different models and degrees of optimization have been tested in the AR application to investigate the tracking quality in relation to the polygon count and the hardware specifications of the device. The specifications of the models tested can be found in the [Test Report Chapter 4.3.2: Models Figure 27: Test Models](#).

## Results

The original Truck model ([Test Report Chapter 4.3.2: Models Figure 27: Test Models](#)) provided by the company did not yield a steady tracking result on either of the devices. While the company experienced reliable tracking with the marker based approach, this model proved to be unoptimized for a markerless tracking solution. As previously mentioned this is due to the fact that markerless tracking requires much more computational resources. A high poly model does not leave enough resources to perform accurate SLAM. As mentioned in the comparison table, the model consists of at least 500 separate objects and >5 million polygons resulting in too many separate draw calls to the GPU, requiring too much computational power. After consulting with the company's graphics expert it was concluded that an optimization of this model would be out of scope of this project. As an alternative the Seat Ateca Model was provided. As seen in the [Test Report Chapter 4.3.2: Models Figure 27: Test Models](#), while the Ateca still consists of 1 Million polygons, the mesh was already much cleaner and provided a better base for optimization processes.

The Samsung Galaxy Tab 3 as the lowest end device did not yield good results with any of the high poly models. The lower the poly and object count however the better the tracking experience was even on the not so powerful device.

Another aspect to consider when optimizing models for AR use is the problem separate objects might cause with light estimation (see [Test Report Chapter 4: Figure 42: Light](#)



[estimation on separate objects \(door and body\)](#)). The light will be calculated for each object separately which might result in uneven lighting conditions. It should therefore be considered carefully which objects are really required as separate objects for interaction to ensure even lighting.

#### 5.2.4 Screen Resolution Independent UI

Because a cross platform approach enables applications to be deployed on a wide range of devices also the UI system has to be considered to fit a wide variety of screen sizes both vertical and horizontal. There are currently hundreds of different devices with different screen sizes. There is no common way of fixing these resolution issues in Unity. There are however a number of best practices recommended by Unity and its community of developers. The general approach recommended in the official Unity manual ([Unity Manual, n.d.](#)) is to use anchors to adapt the UI elements to different screen ratios. The manual mentions that UI elements are by default anchored to the center of the parent rectangle, meaning they are kept in a constant offset from the center. If the resolution is changed to a landscape aspect ratio however, with this setting the buttons may not even be inside the rectangle of the screen anymore. One way to overcome this issue is to use anchors to tie the UI element to a specific position (*Figure 16: Image of Rect Transform Anchor Preset Settings.* [\(Unity, n.d.\)](#)). When changing the aspect ratio now, the UI elements will stay in their respective position. Since the UI elements keep their original size they may change size when the screen size is changed to a smaller or larger resolution. To overcome this side effect the official manual recommends to use the Canvas Scaler component to even out the size percentages (*Figure 17: Image of Canvas Scaler Settings* [\(Unity Manual, n.d.\)](#)). By setting the UI Scale Mode in the Canvas Scaler component to Scale With Screen Size it is possible to specify a resolution to use as a reference. If the actual screen resolution varies from the reference, the scale factor of the Canvas is set accordingly.

### Results

The difference between a UI that has not been optimized according to the steps mentioned as opposed to an optimized UI and the detailed iteration steps taken can be found in the Test Report [Chapter 5: Multi Platform UI](#). Generally, the manual recommendations yield convincing results. The UI elements scale as expected in different orientations. The UI layout and focus of the application need to be considered carefully for each use case however and the UI design must be adapted accordingly. In scenes where textual information is not the main focus it is easier to adapt to both portrait and landscape orientations as the buttons and objects only take up minimal space and can be anchored easily to the canvas and still provide equally good performance. It is evident that for this use case a support for both portrait and landscape mode is not optimal as the textual information can not be read properly in landscape mode as *Figure 34: UI Iteration III* in the [Test Report Chapter 5.4.3 UI Iteration III](#) suggests.

The initially agreed upon system requirement of the content having to scale in both portrait and landscape mode has been revised in correspondence with the company. For this specific use case of displaying a lot of text a landscape orientation is suboptimal as the user can only see a small part of the text without scrolling. It has been decided to instead lock the orientation in portrait mode which makes it easier for the user to read the information instead. With this approach the UI remains its aspect ratio throughout all test devices.

### 5.2.5 Light Estimation

Lighting plays a vital role in creating a realistic and convincing scene. ARFoundation features Light Estimation as a way to adapt the virtual lighting to the real world lighting conditions.

Testing is performed in 3 separate steps:

1. In the first step the general recognition of the real world environment lighting is tested on different devices by deploying the sample scene provided by Unity. The sample scene features a UI interface on which the detected values are outputted. The sample project can be found under:  
<https://github.com/Unity-Technologies/arfoundation-samples>
2. In a second step these received values are applied to the virtual lighting in the scene to synchronize the virtual and the real world light and make the virtual object appear to adapt to the lighting.
3. In a third step the impact of realtime reflection probes on realism is tested. Reflection probes serve as a way to project the real world camera feed onto reflective surfaces of the virtual object and make it appear as if it were reflecting the real environment

The following points were subject of observation:

- Response of the device in recognizing lighting conditions
- Response of the model to a change in lighting conditions
- Application of realtime reflections on reflective materials on the virtual model
- Adaption of color correction on the virtual model in different lighting conditions
- Overall adaptation of light estimation on different devices

### Results

1. As seen in the [Test Report Chapter 6.3.1 Light Estimation Values: Figure 35: Light estimation values](#) both iPhone X and iPhone 7 recognize the prevailing lighting situation and output the detected values to the screen. The brightness is estimated between a value of 0 to 1, where 0 represents dark and 1 represents light. The Samsung Tab 3 apparently does not support any form of light estimation and did not output any values. As a second value the iPhones detected the overall color temperature of the environment, with values < 5000 representing warm tones and values > 5000 representing cool tones. The Samsung Tab 3 did not yield any color temperature results, suggesting that this feature is unavailable on this device.
2. In the second step the detected values are applied to the virtual scene lighting to match it to the real world lighting and have the model be lit in correspondence with the real world lighting. On both iPhones in darker environments the object looks



darker and in light environments the object looks well lit. It is also clear to see that in an environment in which color temp is detected as <5000, meaning warm lighting, the object displays more orange hues. The model on the Samsung device appears unlit since no values have been detected. The results can be found in the [Test Report Chapter 6.3.2 Applied Light Estimation Values: Figure 36: Applied light estimation values](#).

3. When working with reflective objects such as metallic cars, reflection probes can increase the immersion even further by applying realtime realworld reflections to reflective surfaces. This can be achieved by using reflection probes. The requirement for reflection probes to return realtime reflections is that the object's textures support the metallic workflow. The reflectivity and light response of the surface are modified by the metallic and smoothness level of the texture. The result of applied reflection probes can be seen in the [Test Report Chapter 6.3.3 Reflection Probes](#) and following.

*Figure 18* demonstrates the final result of applied light estimation and reflection probes.



*Figure 18:* Final result Light Estimation on iPhone X (left) and Samsung Galaxy Tab3 (right) in natural lighting

### 5.2.6 Occlusion

A big part of creating convincing AR experiences involve the usage of occlusion. Occlusion means that real-world geometry should visually hide virtual geometry and inversely. High-end AR systems like the Magic Leap and Microsoft HoloLens have a lot of computational resources dedicated and feature a special depth camera to mesh the real-world environment to a level that mobile AR is simply not capable of doing at this time. The most recent version of ARKit3 however has found a way to occlude people through a machine learning algorithm instead of spatial understanding (*Figure 20: People Occlusion in ARKit3* [\[Apple, 2019\]](#)). By default virtual content is rendered on top of the camera image.

ARKit 3 is using machine learning to recognize people in the frame and creates a separate layer for these pixels. This process is called segmentation. It also needs to take the distance of people from the object into account. ARKit3 uses advanced machine learning to perform an additional distance calculation step. With this distance the rendering order can be adjusted

and thus it is ensured that both people standing behind and in front of a virtual object are occluded. This is performed on every frame in real time to enable a smooth experience. This also works for half occluded people. At least an A12 processor is required to support this feature. On the available test devices this feature is not supported due to their limited processing power. Since the devices also do not feature a true depth camera they do not have any spatial awareness to facilitate occlusion naturally like the HoloLens. A possible way to overcome this is to “fake” occlusion by simply using a shader that renders the material of found planes transparent as *Figure 21* demonstrates.



*Figure 21: Plane Occlusion Shader*

Since this result is achieved with a simple shader it functions on any device. As evident, the major drawback of this method is firstly that detected planes are rendered transparent and no longer give the user an indication of detected planes. Another drawback is that planes are never tracked 100% identical with the real world resulting in strange overlapping as seen in image 3.

For this use case this method is not suitable and has not been implemented in the final prototype. While people occlusion has been implemented in the final prototype it is not possible to test its functionality at this time.

### 5.2.7 Transparent Light and Shadow Receiver Shader

In some use cases in mobile augmented reality it might be required to project light and shadows onto transparent geometry. In this particular project, when the car's headlights turn on the light should be reflected on the ground. Since an opaque plane underneath the object would break the immersion of the object being anchored in the real world, the plane onto which the shadow and lighting is cast needs to be transparent. Rendering shadows and light onto transparent geometry requires a special custom shader that is not included in the Unity AR project by default. In forward rendering, multi-light shaders use a separate pass for each pixel light in the scene. The shader therefore needs two defined passes. The Base Pass renders the main directional light in the scene, responsible for the shadow of the car. The second pass gets called once for each additional light, and is additively blended with the previous passes. With the help of this custom shader the following result can be achieved:



*Figure 22: Custom shader to render light and shadow on transparent geometry*

With the help of the light estimation values received from the previous step the headlights are enabled and disabled automatically once the environment light falls under a certain threshold. This shader functions on all devices.

## 6. Discussion

As mentioned previously, a crucial aspect of multi-platform support is the computational power of the device. It has been confirmed in the testing that there are some optimization methods that improve the quality of tracking even in devices with less processing power. It is generally advisable to apply these recommendations to any project to maximize the device's potential of tracking the environment. Firstly, the tests confirmed that the lower the poly count and count of separate objects the better is the tracking quality and persistence of virtual content especially on lower end devices. This is due to the high processing power required to render high poly models with a lot of separate objects on screen. The exact amount of polygons that should be used is unique for each use case. In design applications in which more model details should be present a higher polygon count might be more appropriate. It should however always be the goal to reduce the amount as much as possible. Another aspect to consider is that the research results clearly confirm the hypothesis that lighting and surface texture of the environment that the application is used in are equally important and have to be considered carefully. As the test report shows, even higher end devices fail to deliver convincing results in unfavourable conditions, which is dark light and a solid colored surface that does not provide sufficient feature points to track. When working in good lighting conditions on structured surfaces the device's camera quality is directly related to the accuracy of plane detection. Together with the reduction of lighting power, the detection time of the planes increases. Similarly with a reduction in structure on the surface the detection of planes decreased dramatically. The application areas for the AR solutions should therefore be considered carefully before the development. These methods combined provide a decent solution to supporting a wider range of devices in the tracking process. ARFoundation attempts to bridge the gap of feature availability by providing a layer of abstraction, meaning common functionalities supported by both platforms can be integrated on the respective platform through a single shared code basis. It is not possible however to support ARKit 3

specific features on Android devices simply because the underlying technology is too distinct. ARFoundation overcomes this issue by simply omitting features that are not supported on the respective device. While this is very convenient for a quick prototyping flow, this can result in unexpected behaviour such as the model not being lit correctly.

Some of these features such as occlusion can be tried to be replicated by custom shaders to attempt a more unified approach that supports even older Android and iPhone devices as well. These solutions are only substitutes however and might not provide a consistently adequate solution.

It is evident that Apple's ARKit3 capabilities are tremendously more advanced than ARCore's solutions. When trying to create multi platform applications this can become an issue when core functionalities are only supported on iOS devices. It is important to remark that the Android test device's specifications were unfortunately by no means en par with the hardware technology of the iOS devices. It should therefore be considered that a more recent Android device with similar hardware specifications might have yielded more comparable results. The literature research nonetheless clearly confirms that Apple is currently providing the most consistent AR innovations with many of them being exclusive to iOS devices.

Working in a multi platform approach at this stage of technology would in the majority of use cases mean compromising functionality in favour of supporting a wider range of devices. The implication of this should be considered carefully for each use case. In terms of staying on top of creating cutting edge solutions, according to the research and test results iOS devices are clearly the preferred choice of development in terms of their hard-and software advancements.

## 7. Conclusion

The aim of the research was to give an overview of the challenges and possible approaches to a multi-platform development process and how it can be realized under the aspect of different hardware and software requirements. Through extensive testing and comparison of different target devices the research succeeded to establish a guideline for cross-platform development decisions. The research confirms that the AR support and innovations are directly related to the quality of the build-in hardware, namely the quality of the rear facing camera but more importantly the power of the central processing unit. Devices on the market today ship with a variety of different hardware prerequisites. Especially on the Android system the hardware differs greatly as opposed to the somewhat cohesive iOS devices. This makes a true multi platform approach for both systems difficult to realize. While there are nowadays many unifying solutions to a consistent multi platform AR framework, they can only provide a base layer of abstraction for both operating systems.

The research showed that devices with less than a hexa-core processor, 2GB of RAM and a 12MP camera did not yield any usable results for basic reliable AR tracking.

Even though it might be technically possible to use models as low as an iPhone 6S for markerless use cases, the quality of the experience will suffer greatly. Furthermore, many cutting edge features such as occlusion and motion tracking introduced recently will not be available as the software does not support this. This deficit will only increase with future updates to the AR frameworks. According to Apple, at least an iPhone XS is required to support the newest ARKit3 features ([Apple, n.d.](#)). This can be translated to the following hardware specifications required for creating stable, state of the art experiences: Apple A12 Bionic Chip, hexa-core and 4GB of RAM. Therefore only iPhones starting with model XS,

iPhone XS Max, iPhone XR and later can be recommended. For Apple tablets the iPad Air 2019, iPad Mini 2019 and the iPad Pro 2018 and later meet these standards. The diversity of Android devices makes it difficult to create state of the art solutions while still supporting a maximum of devices. With the A12 bionic chip and their machine learning pipeline Apple has achieved a major update in the capabilities of iOS devices that Android cannot compete with at the moment. It is important to note that the test results do not accurately reflect the potential of Android devices as the provided Samsung Galaxy Tab 3 device is hardly comparable with the more modern iOS devices. When comparing the results with the literature research the overall advantage of the Apple systems is still distinct however.

While the in-depth testing on a range of devices ensured that the research was objective and iterations followed the test results, the test devices did not meet the aforementioned criteria. In a lot of cases the features could not be tested to their fullest potential. This leaves it up for debate whether this research actually reflects the full spectrum of AR capabilities at the current moment. On the other hand this simple use case does not require any complicated features. Therefore, within the use case the research is valid as it demonstrates how to create a more unified workflow with the help of ARFoundation. It succeeded to provide optimal platform support but also clearly showed the limitations of the devices. Overall the research managed to deliver a solid foundation of theoretical knowledge and practical guidelines of AR development to consider, in order to be able to manage resources for development more efficiently. With the help of this paper it is possible to make an informed decision on the best development approach for each use case and make accurate estimations about future trends and support of developed products. The mobile AR market however has not reached its full potential yet. As [Gurman \(2019\)](#) from Bloomberg suggests, Apple is planning to add the ToF sensor to the rear camera in 2020. This would enable iPhones to create spatial awareness maps similar to the Holonse. iPhones would then be able to perform true depth sensing and realistic occlusion, allowing for more accurate positional tracking and content positioning. It is certainly advisable to keep a close watch on Apple's future innovations as they currently seem to be both leading in terms of mobile hardware components as well as augmented reality functionalities.



## Appendix I

### Literature Sources

Alcala Toca, F. (2011). *Cross-Platform-Entwicklung unter iOS und Android: Technologieüberblick und Prototyp-basierte Bewertung*. Retrieved from Otto-von-Guericke-Universität Magdeburg.

Apple.(n.d.). *Introducing ARKit 3.5*. Retrieved from <https://developer.apple.com/augmented-reality/arkit/>

Apple Documentation. (2017). *iOS Device Compatibility Reference*. Retrieved from <https://developer.apple.com/library/archive/documentation/DeviceInformation/Reference/iOSDeviceCompatibility/DeviceCompatibilityMatrix/DeviceCompatibilityMatrix.html>

Apple. (2017). *iOS 11 brings powerful new features to iPhone and iPad this fall*. Retrieved from <https://www.apple.com/newsroom/2017/06/ios-11-brings-new-features-to-iphone-and-ipad-this-fall/>

Apple Inc. (n.d.) *Understanding World Tracking*. Retrieved from [https://developer.apple.com/documentation/arkit/understanding\\_world\\_tracking](https://developer.apple.com/documentation/arkit/understanding_world_tracking)

ARCore.(2019). *Using ARCore to light models in a scene*. Retrieved from <https://developers.google.com/ar/develop/unity/light-estimation>

Azuma, R. T. (1997). *A Survey of Augmented Reality*. *Teleoperators and Virtual Environments* 6, 355-385.

Chen, J., Cao, R., Wang, J. (2015). *Sensor-Aware Recognition and Tracking for Wide-Area Augmented Reality on Mobile Phones*. Retrieved from *Sensors* **2015**, 15(12), 31092-31107.

Chopra, S. (2018). *Introduction to Motion Tracking in ARCore....*. Retrieved from <https://medium.com/coding-blocks/introduction-to-motion-tracking-in-arcore-f3e584ce0ba0>

Cross, J. (2018). *iOS12: ARKit2 extends Apple's lead in mobile Augmented Reality*. Retrieved from <https://www.macworld.com/article/3278018/ios-12-arkit-2-apple-mobile-augmented-reality.html>

Prof. Daponte, P. (n.d.) *Smartphones and Augmented Reality for Measurement Applications*. Retrieved from University of Sannio.

Google Inc. (n.d.). *ARCore - Fundamental Concepts*. Retrieved from <https://developers.google.com/ar/discover/concepts>

Google Inc. (n.d.). *ARCore supported devices*. Retrieved from <https://developers.google.com/ar/discover/supported-devices#ios>

Google Inc. (n.d.). *Using ARCore to Light Models in a scene*. Retrieved from <https://developers.google.com/ar/develop/unity/light-estimation>

Google Inc. (2017). *System and Method for Concurrent Odometry and Mapping*. Retrieved from <https://patents.google.com/patent/US20170336511A1/en>

Grossi, M. (2019). *A sensor-centric survey on the development of smartphone measurement and sensing systems*. Retrieved from Measurement Journal of the International Measurement Confederation(IMEKO), Elsevier, 135, 572-592.

GSMarena (n.d.) *Device Hardware Specifications*. Retrieved from <https://www.gsmarena.com>

Gurman, M. (2019). *Apple Is Planning 3-D Cameras for New iPhones in AR Push*. Retrieved from <https://www.bloomberg.com/news/articles/2019-01-30/apple-is-said-to-prep-new-3-d-camera-for-2020-iphones-in-ar-push>

Hum3D (2018). *Seat Ateka FR 2018 3D model*. Retrieved from <https://hum3d.com/3d-models/seat-ateca-fr-2018/>

IAV.(2020). Retrieved from <https://www.iav.com/>

IKEA. ( 2019). *Neue AR-App IKEA Place*. Retrieved from <https://ikea-unternehmensblog.de/article/2019/ikea-place-app>

Jakl, A. (2018). *Basics of AR: SLAM*. Retrieved from <https://www.andreasjakl.com/basics-of-ar-slam-simultaneous-localization-and-mapping/>

Mourcou, Q., Fleury, A., Franco, C., Klopčič, F., & Vuillerme, N. (2015). *Performance evaluation of smartphone inertial sensors measurement for range of motion*. Retrieved from *Sensors*, 15(9), 23169.

Niantec. (2016). *Pokemon GO*. Retrieved from <https://www.pokemongo.com/de-de/>

SpikeSource, Intel Corporation. (2005). *Business Readiness Rating for Open Source - A Proposed Open Standard to Facilitate Assessment and Adoption of Open Source Software*. BRR 2005 - RFC 1

Palladino, T. (2019). *Here's What You Need to Run ARKit 3 on Your iPhone or iPad*. Retrieved from <https://mobile-ar.reality.news/news/heres-what-you-need-run-arkit-3-your-iphone-ipad-0198424/>

Patterson, P. (2017). *Handheld AR App Development with Unity*. Retrieved from Coursera.com

Samsung. (n.d.). *What is ToF camera technology on Galaxy and how does it work?* Retrieved from <https://www.samsung.com/global/galaxy/what-is/tof-camera/>

Schart, D. (2014). *Fünf Anwendungsfelder für Augmented Reality in Automotive*. <http://www.wearear.de/automotiveaugmented-reality/>

Trumpf. (n.d.). *Maschinen und Anlagenplanung with augmented reality*. Retrieved from <https://www.re-flekt.com/de/portfolio-item/augmented-reality-anlagen-produktions-planung-trumpf>

Unity. (n.d.). *Unitys AR Ecosystem ARFoundation ARCore and ARKit*. Retrieved from <https://blogs.unity3d.com/2018/12/18/unitys-handheld-ar-ecosystem-ar-foundation-arcore-and-arkit/>

Unity. (n.d.). *About ARFoundation*. Retrieved from <https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@1.0/manual/index.html#glossary>

Unity Manual. (n.d.). *Designing UI for Multiple Resolutions*. Retrieved from <https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/HOWTO-UIMultiResolution.html>

XinReality. (n.d.). *ARKit*. Retrieved from <https://xinreality.com/wiki/ARKit>

Yudiantika, A. R. (2015). *The Development of Mobile Augmented Reality Quiz Visualization Methods Based on Markerless Tracking for Museum Learning Application*. The 10th International Forum on Strategic Technology 2015

Ziegler, E. (2010). *Real-time markerless tracking of objects on mobile devices*. University of Koblenz and Landau

## Appendix II

### Tables

AR Device Support		
iOS	Device	Supported Features
<b>ARKit1</b>	iPhone6s and later all iPad Pro Models iPad 5th generation ( <a href="#">Apple Documentation, 2017</a> )	<ul style="list-style-type: none"> <li>• Face tracking (only iPhone X and later)</li> <li>• Plane Recognition</li> <li>• Image Recognition</li> <li>• Light Estimation (<a href="#">XinReality, n.d.</a>)</li> </ul>
<b>ARKit2</b>	iPhone 6s and later all iPad Pro models iPad 5th generation iPad 6th generation iPad Air 2019 iPad Mini 2019 Latest edition of iPad Pro (both the 11-inch and 12.9-inch models ( <a href="#">Cross, 2018</a> )	<ul style="list-style-type: none"> <li>• Persistent experiences</li> <li>• Shared experiences</li> <li>• Improved face tracking</li> <li>• More realistic rendering</li> <li>• 3D object detection (<a href="#">Cross, 2018</a>)</li> </ul>
<b>ARKit3</b>	iPhone XS iPhone XS Max iPhone XR iPad Air 2019 iPad Mini 2019 Latest edition of iPad Pro (both the 11-inch and 12.9-inch models ( <a href="#">Palladino, 2019</a> )	<ul style="list-style-type: none"> <li>• People Occlusion</li> <li>• Motion Capture</li> <li>• Multiple faces tracking</li> <li>• Simultaneous front and rear camera world tracking</li> <li>• Collaborative sessions between two or more users (<a href="#">Palladino, 2019</a>)</li> </ul>



Android	Device	Features
<b>ARCore</b>	<p><b>Android</b> For a full list of supported android devices please refer to the official device support documentation for ARCore: <a href="https://developers.google.com/ar/discover/supported-devices#android_play">https://developers.google.com/ar/discover/supported-devices#android_play</a></p> <p><b>iOS</b> iPhone 5S and later All iPad Pros iPad 5th Generation iPad Air + Air 2 iPad Mini 2, 3, and 4 (<a href="#">Google Inc., n.d.</a>)</p>	<ul style="list-style-type: none"> <li>• Motion Tracking</li> <li>• Environmental Understanding (Plane recognition)</li> <li>• Light Estimation (<a href="#">Google Inc., n.d.</a>)</li> </ul>
ARFoundation	Device	Features
	<p><b>Android</b> Same as ARCore support</p> <p><b>iOS</b> same as ARKit support</p>	Same as ARKit + ARCore

Table 2: Feature Availability and Supported Devices

Use Case Specific Device Recommendation	
Platform	Device
<b>iOS phone</b>	iPhone XS and later
<b>iOS Tablet</b>	iPad Pro 2018 and later
<b>Android Devices</b>	Minimum 3GB RAM Minimum Hexa-Core Processor Minimum 12MP Camera

Table 3: Use case specific device recommendations

## Appendix III

### Image Sources

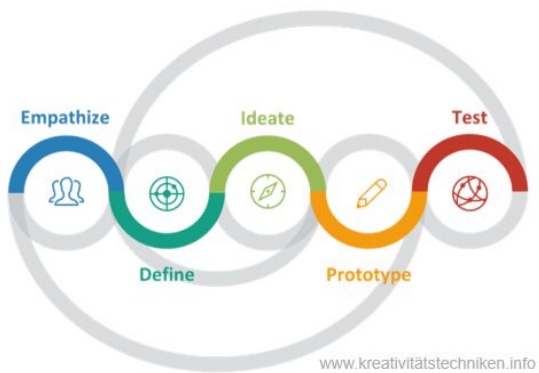


Figure 1: Design Thinking Model

Kreativitätstechniken. (n.d.). *Design Thinking*. Retrieved from

<https://xn--kreativittstechniken-jzb.info/kreativitaetsframeworks/design-thinking/>

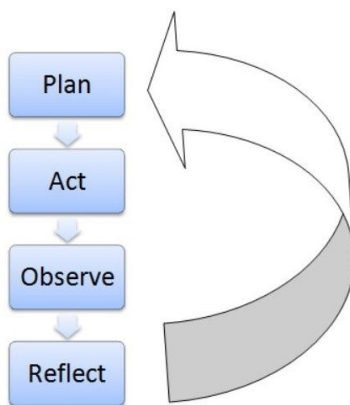


Figure 2: Action Research Model

Research-Methodology. (n.d.). *Action Research Model*. Retrieved from

<https://research-methodology.net/research-methods/action-research/>



Figure 3: Overview of Augmented Reality Cases.

Milgram, P. (1995). *Augmented reality: a class of displays on the reality-virtuality continuum*. Retrieved from Proceedings Volume 2351, Telemanipulator and Telepresence Technologies



Figure 4: Example of marker based AR

Salah-ddine, K. (2019). *Augmented reality types and popular use cases*. Retrieved from [https://www.researchgate.net/figure/example-of-marker-based-AR\\_fig1\\_332543647](https://www.researchgate.net/figure/example-of-marker-based-AR_fig1_332543647)



Figure 5: Example of QR Code

QR Code Generator. (n.d.). *QRCode Generator*. Retrieved from <https://www.qrcode-generator.de/>

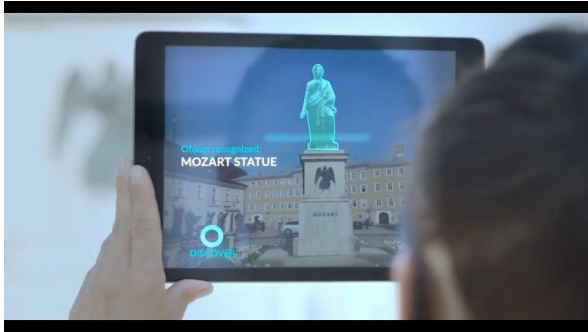


Figure 6: Example of object recognition

Wikitude. (2019). *Object & Scene Tracking: Augmented Reality Use Cases and How-to*. Retrieved from <https://www.wikitude.com/blog-object-scene-tracking-augmented-reality-use-cases-and-how-to/>



Figure 10: Tracked landmarks in an image and their location in a mapped view.

Davison, A. J., Reid, I. D., Molton, N. D., Stasse, O. (2007). *MonoSLAM: Real-time single camera SLAM*. Retrieved from IEEE Transactions on Pattern Analysis & Machine Intelligence, 6, 1052-1067.

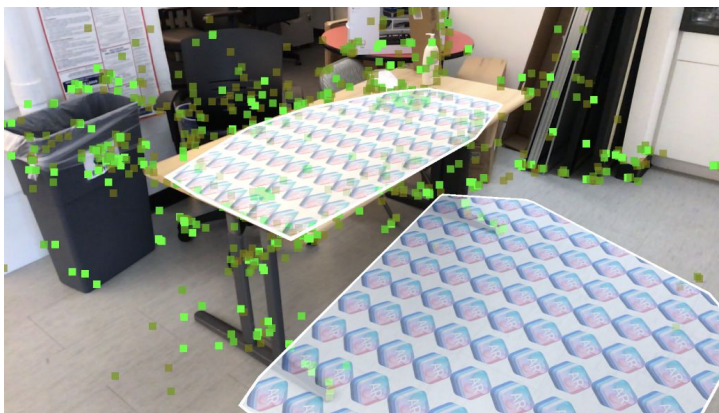


Figure 11: AR Feature Point Clustering

Mukherjee, P. (2018). *Saving ARKit Planes and meshes across multiple sessions using PlacernoteSDK*. Retrieved from <https://medium.com/placernote/saving-arkit-planes-and-meshes-across-multiple-sessions-using-placenotes-dk-fd3ebaa86d2a>

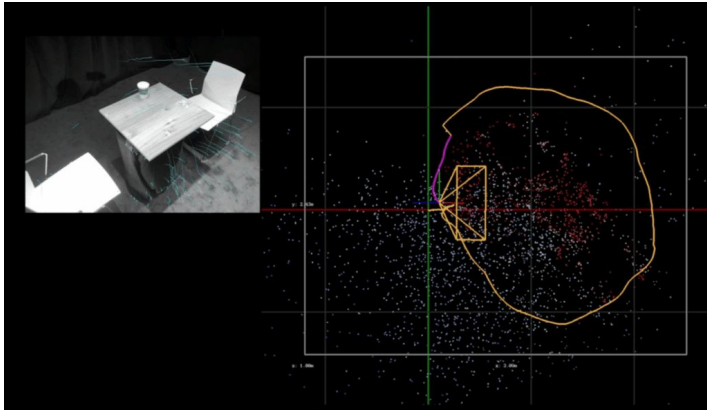


Figure 12: SLAM performed by ARKit, as demonstrated at WWDC 2018  
Apple. (2018). *WWDC 2018*. Retrieved from <https://developer.apple.com/videos/play/wwdc2018/610/>

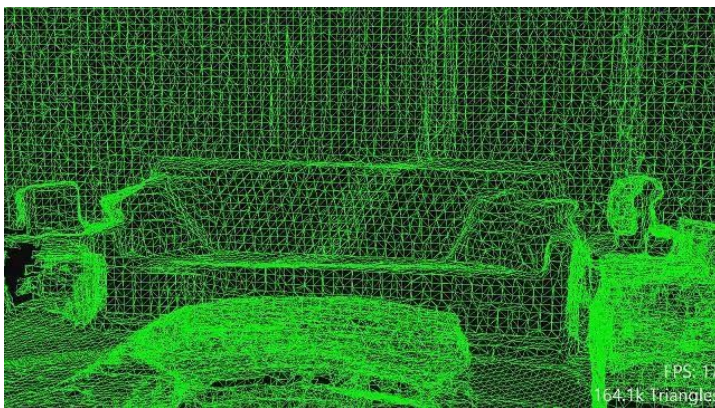


Figure 13: Spatial mapping mesh covering a room  
Microsoft. (2018). *Räumliche Abbildung*. Retrieved from <https://docs.microsoft.com/de-de/windows/mixed-reality/spatial-mapping>

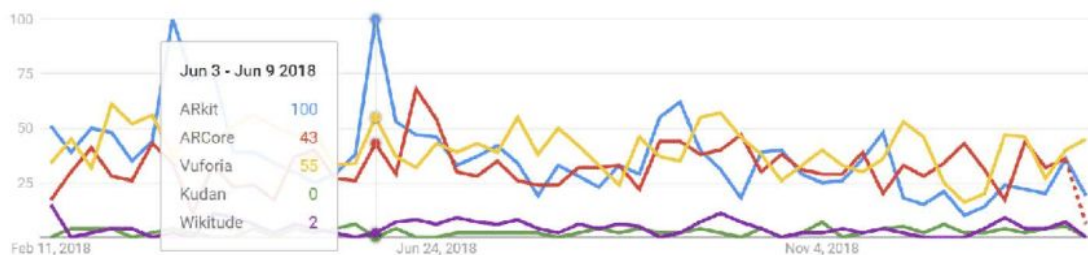


Figure 14: Worldwide interest in AR platforms  
(Numbers represent search interest relative to the highest point on the chart for the given region and time. A value of 100 is the peak popularity for the term. A value of 50 means that the term is half as popular. A score of 0 means there was not enough data for this term.)  
Google Inc. (2018). *Popular AR/VR Frameworks*. Retrieved from [trends.google.com](https://trends.google.com)

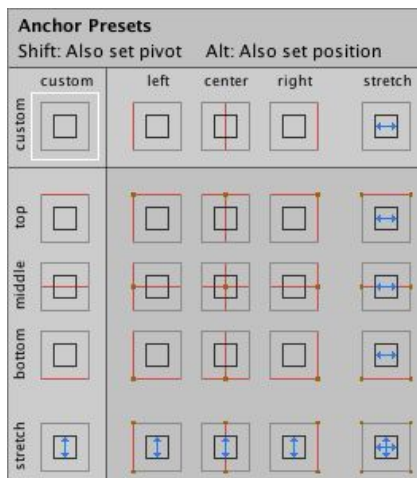


Figure 16: Image of Rect Transform Anchor Preset Settings.

Unity. (n.d.) *Manual: RectTransform*. Retrieved from

<https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/class-RectTransform.html>

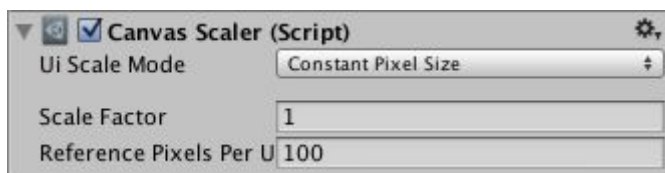


Figure 17: Image of Canvas Scaler Settings

Unity. (n.d.) *Manual: RectTransform*. Retrieved from

<https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/class-RectTransform.html>



Figure 20: People Occlusion in ARKit3

Apple. (2019). *WWDC 2019 Keynote - Apple*. Retrieved from

[https://www.youtube.com/watch?v=psL\\_5RIBqnY](https://www.youtube.com/watch?v=psL_5RIBqnY)



## Appendix IV

### Student Contribution Chart

Assets	Provided by	Comment
Seat Ateca High Poly	Company	
Seat Ateca Low Poly	Student	Optimized Poly/Object count for mobile AR, changed model setup to support animations in Maya 3D
Truck Model	Company	
Cicada	Unity Asset Store	
UI	Provided by	
UI Structure	Student	The student was free to create a unique structure for how the user interacts with the application with no restrictions from the company. This entails all UI elements in the app.
UI Buttons and Elements	Student	The student created all UI Elements and Buttons in Photoshop. Icons were taken from Flaticon.com
Animations	Provided by	Comment
Animations	Student	The student created all animations and interactions in Unity including UI animations, Light driven animations and model animations
Shaders	Provided by	Comment
AR URP Shadow + Lighting Shaders	Student	In order to render light and shadow in AR space on a transparent plane the student wrote a custom shader for the universal rendering pipeline

<b>Scripting</b>	<b>Provided by</b>	<b>Comment</b>
<b>UI configuration</b>	<b>Student</b>	The student setup all UI and button interactions
<b>Information database</b>	<b>Student</b>	To dynamically display custom instructions for each car part the student set up a customizable database structure
<b>Object interaction (rotate, scale, place model)</b>	<b>Student</b>	Scripts for interaction with objects in AR space
<b>Hotspot System</b>	<b>Student</b>	Student created a hotspot system indicating interactions on the car in AR space that fade in/out when occluded by other objects
<b>AR specific features</b>	<b>Provided by</b>	<b>Comment</b>
<b>Light Estimation Recognition</b>	<b>ARFoundation</b>	The system detects light values from the environment based on sensor data
<b>Application + Configuration of Light Estimation</b>	<b>Student</b>	The student wrote scripts to read and process the data to output the values in the UI and apply them to the virtual world. The values are interpreted to drive animations
<b>Plane/Feature Point Recognition</b>	<b>ARFoundation</b>	Once setup, the system's algorithms detect feature points and planes in the environment
<b>Plane/Feature Point Recognition Setup + Configuration</b>	<b>Student</b>	The student setup the AR system in Unity, defined custom materials, custom prefabs and interaction scripts

Table 4: Student Contribution Chart



## Appendix V



An approach to multi-platform augmented reality  
development for mobile devices

# Test Report

Anna-Marie Richter  
Create Media and Technologies  
Saxion University of Applied Sciences

## Table of content

<b>Figure Index</b>	<b>54</b>
<b>Introduction</b>	<b>56</b>
<b>1. Test Plan</b>	<b>56</b>
1.1 Test assignment	56
1.1.1 Introduction and brief project description	56
1.1.2 Commissioning party and stakeholders	57
1.1.3 Test assignment and scope	57
1.2 Test strategy	57
1.2.1 Test Devices	57
1.2.2 Acceptance criteria	58
<b>2. Cross Platform Development Frameworks</b>	<b>58</b>
2.1 Requirements analysis	59
2.2 Scope	59
2.3 Evaluation Model / Test Method	59
2.4 Criteria for the pre-selection of frameworks	60
2.5 Evaluation of the frameworks	63
2.5.1 Functionality	64
2.5.2. Quality/Cost	65
2.5.3 Performance	66
2.5.4. Documentation	67
2.5.5. Community	68
2.5.6 Usability	69
2.5.7 Adoption	69
2.5.8 Support	70
2.6 Decision Matrix	71
2.7 Summary and outlook	72
<b>3. ARFoundation Tracking and Persistence</b>	<b>73</b>
3.1 ARFoundation Plane Tracking	73
3.1.1 Test Cases	73
3.1.2 Test Conditions	74
3.1.3 Test Procedures	74
3.1.4 Test Results	75

3.1.4.1 ARFoundation	75
3.1.5 Evaluation	78
3.2 ARFoundation Object Persistence	80
<b>ARFoundation Object Persistence</b>	<b>80</b>
3.2.1 Test Cases	80
3.2.2 Test Conditions	80
3.2.3 Test Procedures	80
3.2.4 Test Results	81
3.2.4.1 ARFoundation Model Persistence with Plane Tracking	81
3.2.4.2 ARFoundation Model Persistence with Anchor Points	88
3.2.5. Evaluation	92
<b>4. Model Optimization</b>	<b>96</b>
4.1 Test Cases	96
4.2 Scope	96
4.3 Test Conditions	97
4.3.1 Environment	97
4.3.2 Models	97
4.4 Test Procedures	98
4.5 Test Results	99
4.5.1 Truck	99
4.5.2 Seat Ateca	102
4.5.3 Seat Ateca Low Poly	104
4.5.4 Cicada	106
4.6 Evaluation	109
<b>5. Multi Platform UI</b>	<b>110</b>
5.1 Test Cases	110
5.2 Scope	110
5.3 Test Procedure	111
5.4 Results	111
5.4.1. Iteration I - Not optimized UI	111
5.4.2 Iteration II	113
5.4.3 UI Iteration III	114
5.5 Evaluation	115
<b>6. Light Estimation</b>	<b>116</b>

6.1 Test Cases	116
6.2 Scope	117
6.3 Test Results	117
6.3.1 Light Estimation Values	117
6.3.2 Applied Light Estimation Values	118
6.3.3 Reflection Probes	120
6.4 Evaluation	121
<b>7. AR Light and Shadows Support Shaders</b>	<b>123</b>
7.1 Approach	123
7.2 Results	124
<b>8. Final Test Results</b>	<b>124</b>
8.1 Test Cases	124
8.2 Test Results	125
8.3 Evaluation	128
<b>Appendix I</b>	<b>129</b>
Requirements Traceability Matrix	129
<b>Appendix II</b>	<b>132</b>

## Figure Index

### Chapter 1

*Figure 1:* Test device comparison table

### Chapter 2

*Figure 2:* Augmented Reality SDK Comparison.

*Figure 3:* 12 categories of the BRR model

*Figure 4:* Framework functionality evaluation

*Figure 5:* Framework quality/cost evaluation

*Figure 6:* Framework performance evaluation

*Figure 7:* Framework documentation evaluation

*Figure 8:* Framework community evaluation

*Figure 9:* Framework usability evaluation

*Figure 10:* Framework adoption evaluation

*Figure 11:* Framework support evaluation

*Figure 12:* Final decision matrix

### Chapter 3

*Figure 13:* ARFoundation plane tracking test conditions

*Figure 14:* ARFoundation plane tracking results

*Figure 15:* Plane tracking on Samsung Galaxy Tab 3

*Figure 16 + 17:* Plane detection on unstructured surfaces

*Figure 18:* ARFoundation object persistence with plane tracking test results

*Figure 19:* ARFoundation object persistence with anchor points test results

*Figure 20:* Samsung Galaxy Tab 3 Tablet ARFoundation

*Figure 21:* iPhone 7 + ARFoundation

*Figure 22:* iPhone X + ARFoundation

*Figure 23:* iPhone 7 positioning throughout test cases

*Figure 24:* iPhone X positioning throughout test cases

*Figure 25:* iPhone 7 tracking in dim artificial lighting

## **Chapter 4**

*Figure 26: Model Optimization Environment Conditions*

*Figure 27: Test Models*

*Figure 28: Test Results Truck*

*Figure 29: Test Results Seat Ateca*

*Figure 30: Test Results Seat Ateca Low Poly*

*Figure 31: Test Results Cicada*

*Figure 42: Light estimation on separate objects (door and body)*

## **Chapter 5**

*Figure 32: UI Iteration I*

*Figure 33: UI Iteration II*

*Figure 34: UI Iteration III*

## **Chapter 6**

*Figure 35: Light estimation values*

*Figure 36: Applied light estimation values*

*Figure 37: Reflection Probes*

*Figure 38: Reflective car material*

*Figure 39: A range of metallic values from 0 to 1 (with smoothness at a constant 0.8 for all samples)*

## **Chapter 7**

*Figure 40: Custom shader to render light and shadow on transparent geometry*

## **Chapter 8**

*Figure 40: Final Test Results*

## **Appendix I**

*Figure 41: Functional Requirements*

*Figure 42: System Requirements*

## Introduction

This test report complements the graduation report. It focuses on testing prototype iterations by collecting test results from multiple iteration stages to obtain a complete and reliable quality assessment for every phase of the project and confirm the course of action. Based on the project needs, the following topics will be described per iteration test phase:

1. **Test assignment:** introduction and brief iteration description, commissioning party and stakeholders, test assignment and scope.
2. **Test strategy:** describing the test process, quality characteristics, components, acceptance criteria, and test levels.
3. **Test evaluation and quality control:** analyze and interpret the results, ensuring that they correlate with the requirements.

The testing will include the features and system components that must be tested before product delivery. The tests are carried out according to **must and should** requirements. Each test iteration will include a number of test cases. Test cases include the requirements that are being tested, the steps taken during the test procedure, and the acceptance criteria. Once the testing is complete, the results are presented for each test case. The results are interpreted briefly and a general indication of actions for improvements that have to be taken are mentioned.

## 1. Test Plan

### 1.1 Test assignment

#### 1.1.1 Introduction and brief project description

The objective of the graduation assignment is to create a mobile virtual showroom application with the underlying markerless augmented reality technology. It is specified that the application must be built in a multi-platform approach using ARFoundation and the Unity Engine.

The application contains multiple features to demonstrate and configure a car model.

This test plan aims to test all AR components in regards to their usability, stability and performance on multiple platforms throughout all iteration stages in order to ensure that the prototype functions properly and meets the stakeholder's expectations.

#### 1.1.2 Commissioning party and stakeholders

The commissioning party of the graduation assignment is the automotive company IAV. The graduation student is the supplier. The stakeholders include Lars Grotehenne (Scrum



Master/Company Coach), Terence Geldner(Company Coach), Mark Schipper (Saxion graduation coach).

### 1.1.3 Test assignment and scope

The assignment and the scope consist of the following tests: AR frameworks, AR tracking stability test, AR model optimization, AR features and multi-platform development optimizations. The type of testing and criteria are defined individually for each project iteration.

Each test iteration will feature up to 3 repetitions based on the test's complexity.

This paper focuses on markerless tracking solutions and frameworks. Marker based solutions are not subject to this research. Due to the current Covid-19 circumstances this thesis will focus only on the technical prerequisites and recent advancements in the field of mobile augmented reality.

This approach is to ensure that testing does not require any other people and can be facilitated by the student instead. Subject of discussion will be AR frameworks, optimization processes when handling cross platform development and AR features. The prototype has to be created in Unity3D. The supporting AR framework used will be chosen based on the research results. The prototype has the purpose of supporting and demonstrating the technical findings of this research. It does not attempt to provide a finished user experience solution. The actual content and presentation of the app is not subject to this research. To confirm the usability of an augmented user manual, additional user research in regards to the content and UI structure would have to be performed. In the current situation this is not possible.

For this project the multi-platform approach only relates to iOS and Android devices. The testing is limited to the devices provided by the company. Because of the device limitations the most recent AR features will not be included in the testing or the final prototype and instead only be mentioned in theory as they are not supported on any of these devices. There is no budget for development or testing.

All development and testing was done in home office.

## 1.2 Test strategy

### 1.2.1 Test Devices

The test devices provided by the company include:

Device	Released	Processor	Cores	RAM	Screen Resolution	Camera
iPhone 7	2016	Apple A10X Fusion	4	2GB	750 x 1334 pixels, 16:9 ratio (~326 ppi density)	12 MP

iPhone X	2017	Apple A11 Bionic	6	3GB	1125 x 2436 pixels, 19.5:9 ratio (~458 ppi density)	12 MP
iPad Pro	2017	Apple A10X Fusion	6	4GB	1668 x 2224 pixels, 4:3 ratio (~265 ppi density)	12 MP
Samsung Galaxy Tab 3	2013	Intel Atom	2	1GB	600 x 1024 pixels, 16:9 ratio (~170 ppi density)	3.15 MP

Figure 1: Test device comparison table ([GsmArena, n.d.](#))

### 1.2.2 Acceptance criteria

The acceptance criteria are identified for each test case separately.

In order for a test to be considered approved, the following criteria for the individual test elements must be met:

- **PASSED:** amount of *passed* evaluations **100%**;
- **NOT RECOMMENDED:** amount of *passed* evaluations **> 50%**;
- **NOT PASSED:** amount of *passed* evaluations **< 50%**;

## 2. Cross Platform Development Frameworks

Unit Requirement <a href="#">[SR01-M]</a>	Cross-Platform Frameworks	Description <i>Cross-Platform support</i>
Acceptance Criteria	Must be deployable to iOS and Android mobile and tablet devices supporting AR	
State	MET	

## 2.1 Requirements analysis

The use case for this application is the creation of a maintenance/manual app based on the markerless tracking technology. The base requirements for this scenario are the following:

- 1) Fast and reliable tracking of 3D objects in AR space
- 2) Recognition in a range of lighting conditions
- 3) Physically correct application of shaders and rendering methods like lighting and shadows
- 4) Integration of most recent AR features
- 5) Integration of a 2D UI system to display information

Based on this Unity 3D will be used as an engine that supports complex rendering processes on 3D graphics. In the following AR frameworks that integrate with Unity will be regarded.

## 2.2 Scope

Because of personal and time limitations only frameworks that provide a Unity integration can be considered as getting familiar with a new development environment would break the scope of this project. As defined with the stakeholders the app needs to support both iOS and Android devices.

Furthermore it has been defined that only a markerless plane tracking approach should be used for the project. The frameworks listed below have already been chosen in conformity with the scope.

Unity has worked closely with Apple and Google over the years to integrate the **ARKit** and **ARCore** SDKs into its library. In 2018, Unity went public with a unifying API for both platforms with a package called AR Foundation. Because ARFoundation integrates both ARCore and ARKit functionalities with the recent update as well as ARKit only being supported through ARFoundation as of ARKit 3, ARCore and ARKit will not be considered individually.

## 2.3 Evaluation Model / Test Method

The evaluation of the augmented reality frameworks follows the Business Readiness Rating Model (BRR) ([SpikeSource, Intel Corporation, 2005](#)) which is considered an open standard for the evaluation of open source frameworks but can also be used for the comparison of proprietary software.

The model is divided into 4 phases in which the separate software framework components are gradually evaluated.

### 1) Phase 1: Quick Assessment Filter:

Definition and application of the criteria on the established list of all possible software products previously collected for a first pre-selection of frameworks. ([SpikeSource, Intel Corporation, 2005](#))

### 2) Phase 2: Target Usage Assessment:

Weighting and prioritization of the 12 categories and their metrics on which the evaluation of the frameworks happen. ([SpikeSource, Intel Corporation, 2005](#))

### 3) Phase 3: Data Collection & Processing:

The normalized metrics are now applied to the previously collected data and calculated against the weighting factors of the individual metrics.

By default a scale from 1-5 is used for this in which 1 is defined as not acceptable and 5 is defined as outstanding. [\(SpikeSource, Intel Corporation, 2005\)](#)

### d) Phase 4: Data Translation:

From the sum of evaluations for the single metrics an overall score is created per category.

Finally, these category ratings are accounted for with the weighting of the single categories and in a decision matrix the BRR point score is determined for each software product. [\(SpikeSource, Intel Corporation, 2005\)](#)

The data collection in phase 3 is realized through desk and literature research by gathering information from the official website and documentation of the respective framework.

## 2.4 Criteria for the pre-selection of frameworks

The currently available augmented reality SDKs according to Socialcompare [\(Socialcompare, 30. April 2020\)](#) are listed in *Figure 2: Augmented Reality SDK Comparison*.

To have a manageable amount of frameworks to evaluate the following pre-selection criteria have been defined in relation to the scope and purpose of the project:

- 1) Cross-Platform: All selected frameworks are providing at least one SDK for iOS and Android for a platform independent development to reduce development and maintenance costs of hybrid applications.
- 2) Natural Feature Tracking: The selected framework supports a markerless tracking approach based on natural feature tracking
- 3) Free (trial): The selected framework is either open source or provides a free trial license in order to be able to work with it during this project
- 4) Unity 3D Integration: the selected framework provides an integration for the Unity 3D engine

The rows highlighted in red in *Figure 2: Augmented Reality SDK Comparison* are the frameworks that fulfil these requirements.

No	Framework	Cross-Platform	NFT	Free Trial Available	Unity 3D
1	ALVAR				

2	AndAR				
3	ARToolkit	x	x	x	
4	ARMedia		x		x
5	Aurasma		x		
6	Awila				
7	BazAR	x	x		
8	Beyond Reality Face	x		x	
9	BeyondAR		x		
10	Catchoom		x		x
11	Cortexia				
12	D'Fusion		x		x
13	DeepAR.ai				
14	Designers ARToolkit (DART)				
15	DroidAR		x		
16	EasyAR	x	x		x
17	flare		x		
18	FLARToolkit	x			
19	Goblin XNA	x			
20	Google ARCore	x	x	x	x
21	HERE Mobile SDK		x		x
22	HoloBuilder		x		x

23	HOPPALA				
24	iPhone ARKit	In combination with ARFoundation	x	x	x
26	instantreality				
27	Koozyt		x		
28	LibreGeoSocial				
29	LinceoVR		x		
30	linkme				
31	Luxand FaceSDK				
32	Magic Face				x
33	MAXST AR	x	x	x	x
34	Minerva				
35	mixare				
36	Morgan				
37	MXR Toolkit				
38	NyARToolkit	x			x
39	ObviousEngine		x		x
40	omniar.com				
41	osgART		x		
42	Kudan AR Engine	x	x		x
43	Vuforia	x	x	x	x
44	Robocortex		x		x

45	SLARToolkit				
46	snaptell				
47	String				x
48	Studierstube				
49	Unity ARFoundation	x	x	x	x
50	Vidinoti PixLive		x		
51	Viewdle				
52	Wikitude	x	x		x
53	Xloudia		x		x
55	Zenitum Feature Tracker		x		

Figure 2: Augmented Reality SDK Comparison. [\(Socialcompare, 30. April 2020\)](#)

Identified as potential frameworks have been EasyAR, ARCore, MaxST AR, Kudan AR Engine, Vuforia, Unity ARFoundation and Wikitude. Since ARCore and ARKit functionalities are already wrapped by the Unity ARFoundation SDK as well as Wikitude, in the following only the ARFoundation SDK will be considered in place of ARCore and ARKit. It is worth noting that both ARCore and ARKit are either open source or do not require commercial licensing. It is advisable however to analyze the working environment in which the SDK will be integrated separately in regards to their licensing costs.

## 2.5 Evaluation of the frameworks

The 12 categories of the BRR model are listed and sorted in *Figure 3: 12 categories of the BRR model* based on their importance and provided with their appropriate weightings. Only the first 8 categories are subject of this investigation. The other categories are not relevant for this use case.

Grade	Category	Weighting
1	Functionality	<b>20%</b>
2	Quality/cost	<b>20%</b>



3	Performance	<b>20%</b>
4	Documentation	<b>15%</b>
5	Community	<b>10%</b>
6	Usability	<b>5%</b>
7	Adoption	<b>5%</b>
8	Support	<b>5%</b>
9	Scalability	<b>0%</b>
10	Architecture	<b>0%</b>
11	Security	<b>0%</b>
12	Professionalism	<b>0%</b>

Figure 3: 12 categories of the BRR model ([SpikeSource, Intel Corporation, 2005](#))

In the following, the metrics and evaluations of the single categories are highlighted. Every metric will be evaluated with 1-5 points. Subsequently the points are added with the weighting factor (%) to calculate the total score of a metric.

### 2.5.1 Functionality

The metrics in the category Functionality are only assigned 5 points if the functionality exists or 1 point if the functionality does not exist. All of the frameworks support the tracking of natural features.

All frameworks except Kudan support the recognition of planes. Kudan instead focuses on creating anchor points to track an object's position. Therefore Kudan receives 1 point for the plane tracking but 5 points for anchor points. All other frameworks except EasyAR support anchor points as well and are assigned 5 points. Spatial mapping on mobile devices is only supported by Kudan and EasyAR and are therefore assigned 5 points. All platforms except Wikitude provide a form of occlusion and receive 5 points. Only Unity's ARFoundation as a wrapper for ARCore and ARKit provides the light estimation feature and is evaluated with 5 points.

No	Metric	%	EasyAR	Unity ARFoundation	MAXS T AR	Kudan AR Engine	Vuforia	Wikitude

1.1	Plane Tracking	25%	1,25	1,25	1,25	0,25	1,25	1,25
1.2	Anchor points	25%	0,25	1,25	1,25	1,25	1,25	1,25
1.3	Spatial mapping	25%	1,25	0,25	0,25	1,25	0,25	0,25
1.4	Realtime Occlusion	15%	0,75	0,75	0,75	0,75	0,75	0,15
1.5	Light Estimation	10%	0,1	0,75	0,1	0,1	0,1	0,1
Evaluation in points			3,6	4,25	3,6	3,6	2	3

Figure 4: Framework functionality evaluation

### 2.5.2. Quality/Cost

With 9 minor releases within 12 months and 12 documented bug fixes Wikitude SDK for Unity is updated with new features frequently. Vuforia, MaxST and EasyAR do not indicate the date of the updates they do however provide extensive bug fixes and updates for the Unity SDK that are very well documented and will therefore receive 4 points. Kudan for Unity has last been updated in July 2019 and will therefore receive only 1 point for its actuality. In terms of licensing costs, Unity provides ARFoundation within the standard Unity license and can therefore be considered the best option for this use case and is assigned 5 points. EasyAR charges \$39 per month, Vuforia \$42 and MaxST similarly \$50. Kudan and Wikitude can only be assigned 1 point in terms of cost because they charge the highest amount within the chosen frameworks, namely \$125 per month for Kudan and \$190 a month for Wikitude.

No	Metric	Description	%	EasyAR	Unity ARFoundation	MAXST AR	Kudan AR Engine	Vuforia	Wikitude
2.1	Number of minor releases in the	This measures how up to date the SDK is kept	35 %	1,4	1,75	1,4	0,35	1,4	1,75

	last 12 months								
2.2	Number of bugs fixed in last 6 months	This measures how quickly bugs are fixed to prevent security issues	35 %	1,4	1,75	1,4	0,35	1,4	1,75
2.3	Licencing Cost		30 %	0,9	1,5	0,9	0,3	0,9	0,3
Evaluation in points				3,7	5	3,7	1	3,7	3,8

Figure 5: Framework quality/cost evaluation

### 2.5.3 Performance

The following tests have been performed in optimal natural lighting conditions on a structured surface on an iPhone 7. If the object stays persisted 5 points are assigned. If the object moves away from its original position the framework is assigned 1 point. All frameworks provide almost equally good tracking capabilities between 4-5 points. The most reliable tracking is provided by Wikitude, Vuforia and ARFoundation.

No	Metric	Description	%	EasyAR	Unity ARFoundation	MAXST AR	Kudan AR Engine	Vuforia	Wikitude
3.1	Placing a 3D object on a plane	Object sits right on surface	20%	1	1	1	1	1	1
3.2	Moving back and forth from the object in a straight horizontal line	Object stays in place	20%	0,8	1	0,8	1	1	1

3.3	Moving the camera up and down	Object stays in place	<b>20%</b>	<b>0,8</b>	<b>1</b>	<b>0,8</b>	<b>1</b>	<b>1</b>	<b>1</b>
3.4	Walking around the object	Object stays in place	<b>20%</b>	<b>0,8</b>	<b>1</b>	<b>0,8</b>	<b>0,8</b>	<b>1</b>	<b>1</b>
3.5	Re-focussing the object after tilting camera away	Object stays in place	<b>20%</b>	<b>0,8</b>	<b>1</b>	<b>0,8</b>	<b>0,8</b>	<b>1</b>	<b>1</b>
Evaluation in points				<b>4,2</b>	<b>5</b>	<b>3,6</b>	<b>4,2</b>	<b>5</b>	<b>5</b>

Figure 6: Framework performance evaluation

#### 2.5.4. Documentation

All frameworks are providing extensive instructions on how to setup an AR app in the Unity 3D engine and receive for metric 3.1 5 points each. The documentations for ARFoundation and Wikitude are very detailed and provide extensive instructions on how to implement individual features. They each receive 5 points in metric 3.2. All other frameworks provide at least some form of documentation and setup instructions, they are however not nearly as extensive. Only MaxST, Kudan and Vuforia do not provide any source code examples at all and receive 1 point for metric 3.5. All frameworks except for EasyAR provide some form of demo application and thus receive full points. Even though Wikitude does provide a demo application they charge a fee for 499€ for it and will thus be weighted with 1 point. Extensive source code documentation is only provided by Unity ARFoundation, Wikitude, EasyAR and ARCore. They therefore receive 5 points each for metric 3.4. It was not possible to trace the source code documentation for Vuforia, MAXST and Kudan. They will therefore receive 1 point for the metric. Clearly, the highest score with 5 points goes to ARFoundation in the category of documentation.

No	Metric	%	EasyAR	Unity ARFoundation	MAXST AR	Kudan AR Engine	Vuforia	Wikitude
4.1	Instructions for creation of an AR App in Unity available	35%	1,75	1,75	1,75	1,75	1,75	1,75
4.2	Instructions for usage of features available	30%	0,9	1,5	1,2	0,75	0,75	1,5
4.3	Demo App Available	25%	0,25	1,25	1,25	1,25	1,25	0,25
4.4	Source Code documentation available	10%	0,5	0,5	0,1	0,1	0,1	0,5
Evaluation in points			3,4	5	4,3	3,85	3,85	4

Figure 7: Framework documentation evaluation

#### 2.5.5. Community

EasyAR and Kudan receive 2 points. Even though a forum is available the last entry dates back a couple of months. ARCore and Unity ARFoundation receive 4 points. Even though they don't provide a forum of their own they have both a gitHub forum as well as a very active StackOverflow participation.

Vuforia, Wikitude, ARCore and ARFoundation receives 5 points because their community is very active with the last entry from a couple of days ago and provide community based video tutorial

No	Metric	%	EasyAR	Unity ARFoundation	MAXST AR	Kudan AR Engine	Vuforia	Wikitude
5.1	Developer Forum Available	100%	2	5	1	2	5	5

Figure 8: Framework community evaluation

### 2.5.6 Usability

The points are assigned according to the following setup times:

5 points: < 10 minutes

4 points: 10 min - 20 min

3 points: 20 min - 30 min

2 points: 30 min - 1 hours

1 points: > 1 hours

In order to enable a free trial license for Wikitude a complex setup with a registration process is required. Therefore Wikitude only receives 2 points. AR was slightly complex with setting up the free license and is therefore evaluated with 4 points. Similarly MaxST and Kudan required separate SDK downloads and manual installation. Vuforia provides a very simple setup and is assigned 5 points. ARFoundation is easily set up through the Unity package manager and assigned 5 points.

No	Metric	%	EasyAR	Unity ARFoundation	MAXST AR	Kudan AR Engine	Vuforia	Wikitude
6.1	Setup time for Unity integration	100%	4	5	4	4	5	2

Figure 9: Framework usability evaluation

### 2.5.7 Adoption

The distribution and popularity of the frameworks in the industry is difficult to measure. It however should still be considered to give a general indication on how the market adaptability is.

Some companies do not announce which framework has been used to realize their applications. Similarly frameworks can be denied to publicize applications on their reference list. Nevertheless metric 7.1 has been defined because it gives a quick overview on the potential distribution.

From the more than 900 listed applications listed on Vuforia's website it can be speculated this framework is the most popular one on the current market. It is valued with 5 points, followed by Wikitude with 100 apps listed on their website. For Unity's ARFoundations no specifically declared apps are listed. However Unity itself is a very popular engine frequently used in the industry. It is therefore still assigned 2 points. EasyAR, MaxST and Kudan receive 3 points for displaying some reference applications. Furthermore the average monthly search

requests for the frameworks have been considered. Keyword-tools.org has provided the following values:

Vuforia 60.500  
Wikitude 6.600  
ARFoundation 4.400  
easyAR 2.400  
Maxst 1.000  
Kudan AR 320

Based on this Vuforia has been assigned the highest rating of 5 points, followed by Wikitude and ARFoundation with 4 points and easyAR and Maxst with 3 points. Kudan AR with the lowest average search requests per month receives 2 points.

No	Metric	%	EasyAR	Unity ARFoundation	MAXST AR	Kudan AR Engine	Vuforia	Wikitude
7.1	Amount of Reference Applications	80%	2,4	1,6	2,4	2,4	4	4
7.2	Amount of monthly search requests	20%	0,6	0,8	0,4	0,6	1	0,8
Evaluation in points			3	2,4	2,8	3	5	4,8

Figure 10: Framework adoption evaluation

### 2.5.8 Support

All frameworks except for EasyAR provide some form of technical support. While Kudan does provide support as well, it has been evaluated with 1 point because of the fee of \$500 per ticket. All other frameworks provide email support, extensive tutorials and a broad FAQ section.



No	Metric	Description	%	EasyAR	Unity ARFoundation	MAXST AR	Kudan AR Engine	Vuforia	Wikitude
7.1	Technical support available	Professional support that helps fine-tune for the local deployment and troubleshooting	100 %	1	5	5	1	5	5

Figure 11: Framework support evaluation

## 2.6 Decision Matrix

The results of the categories are calculated with the previously established weightings and compared in the following table. From a maximum of 5 points in the BRR model Unity ARFoundation reached the highest score of 4,72 for reaching good grades in almost all categories.

By wrapping both the popular open source frameworks ARKit and ARCore ARFoundation provides an extensive and reliable set of features, combined with excellent documentation and a very active community. Considering the licensing costs ARFoundation provides the best cost to performance ratio because ARFoundation is included in the Unity licensing cost which is required anyways when working commercially with the Unity 3D Engine. ARFoundation is followed closely by Wikitude with 4,05 points. While Wikitude provides a similarly excellent set of features and performance, the high licensing costs are its main drawback. Even though Vuforia is a popular choice in the AR market its prime focus are marker based projects. With a score of 3,967 it is still a valid choice.

EasyAR, MaxST and Kudan did not show outstanding results especially in regards to actuality and documentation. While they deliver a good performance and a standard set of features, they do not have a community as active as Unity or Wikitude. The very good results of all frameworks reflects that a multi platform development is very much possible with all of them. For the use case of this project the decision on the framework for a markerless cross-platform development of an AR app based on the results of the decision matrix goes to the ARFoundation framework.

No	Metric	%	EasyAR	Unity ARFoundation	MAXST AR	Kudan AR Engine	Vuforia	Wikitude
1	Functionality	20 %	0,72	0,85	0,72	0,72	0,4	0,6
2	Quality/Cost	20 %	0,74	1	0,74	0,2	0,74	0,76
3	Performance	20 %	0,48	1	0,72	0,84	1	1
4	Documentation	15 %	0,51	0,75	0,654	0,577	0,577	0,6
5	Community	10 %	0,2	0,5	0,1	0,2	0,5	0,5
6	Usability	5%	0,2	0,25	0,2	0,2	0,25	0,1
7	Adoption	5%	0,15	0,12	0,14	0,15	0,25	0,24
8	Support	5%	0,05	0,25	0,25	0,05	0,25	0,25
<b>Total Score</b>			<b>3,05</b>	<b>4,72</b>	<b>3,524</b>	<b>2,937</b>	<b>3,967</b>	<b>4,05</b>

Figure 12: Final decision matrix

## 2.7 Summary and outlook

In the scope of this evaluation 55 augmented reality SDKs have been found of which 6 could potentially be suitable for a markerless application in a business environment. The individual prioritization and weighting of the categories of the BRR model enabled an adjusted evaluation for the specific business case of this project. Already in the individual category ratings it was eminent that the final decision would most likely go to one of the subjectively three most popular frameworks in the industry, namely ARFoundation, Vuforia and Wikitude. They achieved predominantly very good results. The BRR model does however hold some weakness when evaluating specifics that cannot be measured directly such as market adoption or an active community. The attempt was to create an evaluation matrix that somehow makes the semantic evaluations more tangible on a scientific level. Furthermore it would be interesting to measure the effectiveness of the overall business case of porting an automotive service and instructions manual to the AR space in terms of time reduction, improved user engagement, understanding and brand identification. The interconnectedness

of the real and the virtual world has increased over the last years and will continue to grow especially in the industrial automotive environment. With a steady, rapid increase of the development of new AR technologies and SDKs it can be expected that not all of the aforementioned frameworks will provide a continuous development. The top 3 exposed frameworks provide the momentarily highest market share and have been developed and refined in their features at a steady rate in the past and present. They can therefore safely be implemented in business applications today ensuring a very high probability of remaining their accuracy even in the near future.

### 3. ARFoundation Tracking and Persistence

#### 3.1 ARFoundation Plane Tracking

Unit Requirement <a href="#">[SR06-M]</a>	ARFoundation plane tracking	Description <i>Detecting Surfaces</i>
Acceptance Criteria	System must recognize planes reliably to ensure the virtual content behaves as expected	
State	MET	

##### 3.1.1 Test Cases

One of the key elements of augmented reality is the device's "understanding" of the surrounding space. The main component in markerless vision based tracking is surface detection. Well-mapped surfaces allow for realistic placement of virtual objects in real space. The quality of the surface detection translates to the quality of the entire application. If the accuracy is too low virtual models will hang in the air, move, or appear in different places which breaks the immersion.

The testing is done in ARFoundation. The ARFoundation framework as well as ARCore and ARKit offer detection of flat horizontal surfaces (floors, countertops) and vertical surfaces (walls).

Throughout this project only horizontal planes are considered.

The following points are subject of observation:

- Accuracy of planes detected in relation to the total surface area
- Working in various lighting conditions
  - Impact of brightness and color of light on virtual objects daylight, incandescent light (60 W bulb)
- Working in unfavourable conditions

- Impact of a shape on plane mapping: flat surface, single-colored, devoid of pattern and texture such as plane tables,  
flat surface with a pattern such as wooden tables

### 3.1.2 Test Conditions




Lighting	Structure	Reference
Bright natural light	Structured Flooring/Carpet	
Bright natural light	Unstructured solid colored table with minor feature points	
Artificial light	Structured wooden table	

Figure 13: ARFoundation plane tracking test conditions

### 3.1.3 Test Procedures

The test application was deployed on the device. The test surface and lighting conditions were prepared for each step. To measure the accuracy of plane detection 20 measurements were taken for each platform and device to map a test surface measuring around 90 cm x 60 cm. The measured parameter was the average percentage of coverage of the test plane by the mapped surface. This measure was mostly based on sensible estimation. The surface mapping was executed following the general recommendation of slow circular motions.

In order to be evaluated as **PASSED** >90% coverage is required.  
Up to 50% coverage is evaluated as **NOT RECOMMENDED**.  
Anything below 50% is evaluated as **NOT PASSED**.

### 3.1.4 Test Results

#### 3.1.4.1 ARFoundation

Accuracy of Plane Tracking			
Priority	High	Description	Accuracy of recognizing and tracking planes in different conditions
Date	18/03/2020	Performed by	Student
Test Software	ARFoundation	Test Devices	Apple iPhone 7 Apple iPhone X Samsung Tablet Galaxy Tab 3
TEST PROCEDURE			
Condition		Bright natural light, Structured flooring/carpet	
Device		iPhone 7	
Action		Expected	Actual
1. Average coverage of a 90cm x 60cm surface in %		100%	93%
Passed			
Device		iPhone X	
Action		Expected	Actual
1. Average coverage of a 90cm x 60cm surface in %		100%	94%
Passed			

Device	Samsung Tab 3	
Action	Expected	Actual
1. Average coverage of a 90cm x 60cm surface in %	100%	53%
Not Recommended		
Condition	Bright natural light, Unstructured uni colored table with minor features points	
Device	iPhone 7	
Action	Expected	Actual
1. Average coverage of a 90cm x 60cm surface in %	100%	0%
Failed		
Device	iPhone X	
Action	Expected	Actual
1. Average coverage of a 90cm x 60cm surface in %	100%	0%
Failed		
Device	Samsung Tab 3	
Action	Expected	Actual
1. Average coverage of a 90cm x 60cm surface in %	100%	0%
Failed		
Condition	Artificial light Structured wooden table	
Device	iPhone 7	
Action	Expected	Actual
1. Average coverage of a 90cm x 60cm surface in %	100%	91%
Passed		

Device	iPhone X	
Action	Expected	Actual
1. Average coverage of a 90cm x 60cm surface in %	100%	92%
Passed		
Device	Samsung Tab 3	
Action	Expected	Actual
1. Average coverage of a 90cm x 60cm surface in %	100%	54%
Not Recommended		

Figure 14: ARFoundation plane tracking results

### 3.1.5 Evaluation

The tests showed the tendency of the ARFoundation platform to cover a slightly bigger area of the tested plane. The application in ARFoundation will in most cases be able to cover the surface to a satisfactory degree (over 90%) when using a new device with a high quality camera. It is evident that the Samsung device has failed to deliver usable results and can not be considered a development choice. Due to hardware limitations it was not possible to test on a more recent Android device. Considering an optimal usage condition it can be concluded that using markerless AR on an unstructured surface that does not provide any feature points can not be recommended and will not yield any results regardless of the quality of the device.



Figure 15: Plane tracking on Samsung Galaxy Tab 3

Together with the reduction of lighting power, the detection time of the planes increased. Similarly with a reduction in structure on the surface the detection of planes decreased dramatically. The accuracy of tracking stayed above 90% for the higher quality iOS devices regardless of the lighting condition as long as a structured surface was present, while the Samsung device showed major difficulties in any condition except bright natural light and a highly structured surface. All devices failed to detect plain unstructured surfaces under all conditions. The general recommendations from Apple and Google can be confirmed. Independently from the device’s camera quality and internal processing abilities it is evident that solid surfaces without notable feature points (patterns or structural differences) are not recognized by the device at all. The following screenshots illustrate this:



Figure 16 + 17: Plane detection on unstructured surfaces

Plain white surfaces of the table are not recognized, while the areas containing some form of disturbance are clearly marked as planes. This could be observed throughout all devices. This is also reflected in the preceding test results table.

3.2 ARFoundation Object Persistence

Unit	Description	
Requirement	ARFoundation Object Persistence	Anchoring virtual objects on detected surfaces
<u>[SR03-M]</u>		
Acceptance Criteria	Framework must support reliable anchoring of virtual object so they stay in place without moving	



State	MET
-------	-----

### 3.2.1 Test Cases

In order to create a convincing experience, virtual objects placed in the environment should stay in their places at all times. Testing is facilitated under 3 different realistic setups and 3 different target devices. The following points are subject of observation:

- Positioning of 3D content on planes
  - Working in various lighting conditions
    - Impact of brightness and color of light on virtual objects [daylight, incandescent light (60 W bulb)]
  - Work in unfavourable conditions
    - Impact of a shape on plane mapping: flat surface, single-colored, devoid of pattern and texture such as plane tables, flat surface with a pattern such as wooden tables
- Work in motion (rapid movement, different angles, losing track of the object)

### 3.2.2 Test Conditions

See test conditions chapter 3.1.2

### 3.2.3 Test Procedures

The testing is done in 10 trials per device.

The result in the table below reflects the average result from all trials.

The steps taken are as follows:

- Creating a basic setup for plane recognition and object placement in ARFoundation
- Deploying the app on all aforementioned test devices
- Preparing the environment for different test conditions
- Open the app on a device and follow the recommended approach on finding planes
- Place object on plane
- Run through different test scenarios to test tracking stability
- Repeat with different device

### 3.2.4 Test Results

#### 3.2.4.1 ARFoundation Model Persistence with Plane Tracking

#### Accuracy of Tracking

<b>Priority</b>	<b>High</b>	<b>Description</b>	<i>Ensuring that 3D content placed in the world stays persistently in place from all viewpoints under different conditions.</i>
<b>Date</b>	<i>10/04/2020</i>	<b>Performed by</b>	<i>Student</i>
<b>Test Software</b>	<i>ARFoundation</i>	<b>Test Devices</b>	<i>Apple iPhone 7 Apple iPhone X Samsung Tablet Galaxy Tab 3</i>
<b>TEST PROCEDURE</b>			
<b>Condition</b>	Bright natural light, Structured flooring/carpet		
<b>Device</b>	<b>iPhone 7</b>		
<b>Action</b>	<b>Expected</b>	<b>Actual</b>	
1. Placing a 3D object on the plane	Object sits right on top of surface	<b>As expected</b>	
2. Moving back and forth from the object in a straight horizontal line	Object stays in place without moving	<b>As expected</b>	

3. Moving the camera up and down	Object stays in place without moving	<b><i>As expected</i></b>
4. Walking around the object	Object stays in place without moving	<b><i>As expected</i></b>
5. Re-focussing the object after tilting camera away	Object stays in place without moving	<b><i>As expected</i></b>
<b>Passed</b>		
<b>Device</b>	<b>iPhone X</b>	
<b>Action</b>	<b>Expected</b>	<b>Actual</b>
1. Placing a 3D object on the plane	Object sits right on top of surface	<b><i>As expected</i></b>
2. Moving back and forth from the object in a straight horizontal line	Object stays in place without moving	<b><i>Minor jittering when camera refocuses</i></b>
3. Moving the camera up and down	Object stays in place without moving	<b><i>Minor jittering when camera refocuses</i></b>
4. Walking around the object	Object stays in place without moving	<b><i>As expected</i></b>
5. Re-focussing the object after tilting camera away	Object stays in place without moving	<b><i>Minor jittering when camera refocuses</i></b>
<b>Not recommended</b>		

Device	Samsung Tab 3	
Action	Expected	Actual
1. Placing a 3D object on the plane	Object sits right on top of surface	<i>Slightly floating</i>
2. Moving back and forth from the object in a straight horizontal line	Object stays in place without moving	<i>In most cases as expected, frequently jumps and drifts</i>
3. Moving the camera up and down	Object stays in place without moving	<i>In most cases as expected, frequently jumps and drifts</i>
4. Walking around the object	Object stays in place without moving	<i>Random drifting and jumping</i>
5. Re-focussing the object after tilting camera away	Object stays in place without moving	<i>Random drifting and jumping</i>
<b>Not passed</b>		
Condition	Bright natural light, Unstructured uni colored table with minor features points	
Device	iPhone 7	
Action	Expected	Actual
1. Placing a 3D object on the plane	Object sits right on top of surface	<i>Minor floating due to feature points</i>

2. Moving back and forth from the object in a straight horizontal line	Object stays in place without moving	<b><i>As expected</i></b>
3. Moving the camera up and down	Object stays in place without moving	<b><i>As expected</i></b>
4. Walking around the object	Object stays in place without moving	<b><i>As expected</i></b>
5. Re-focussing the object after tilting camera away	Object stays in place without moving	<b><i>As expected</i></b>
<b>Not Recommended</b>		
<b>Device</b>	<b>iPhone X</b>	
<b>Action</b>	<b>Expected</b>	<b>Actual</b>
1. Placing a 3D object on the plane	Object sits right on top of surface	<b><i>Minor floating due to feature points</i></b>
2. Moving back and forth from the object in a straight horizontal line	Object stays in place without moving	<b><i>Minor jittering due to camera refocusing</i></b>
3. Moving the camera up and down	Object stays in place without moving	<b><i>As expected</i></b>
4. Walking around the object	Object stays in place without moving	<b><i>As expected</i></b>

5. Re-focussing the object after tilting camera away	Object stays in place without moving	<b>Minor jittering due to camera refocusing</b>
<b>Not recommended</b>		
<b>Device</b>	<b>Samsung Tab 3</b>	
<b>Action</b>	<b>Expected</b>	<b>Actual</b>
1. Placing a 3D object on the plane	Object sits right on top of surface	<b>Minor floating</b>
2. Moving back and forth from the object in a straight horizontal line	Object stays in place without moving	<b>Major jumping and drifting</b>
3. Moving the camera up and down	Object stays in place without moving	<b>Major jumping and drifting</b>
4. Walking around the object	Object stays in place without moving	<b>Minor drifting</b>
5. Re-focussing the object after tilting camera away	Object stays in place without moving	<b>Major jumping</b>
<b>Not Passed</b>		
<b>Condition</b>	Artificial light Structured wooden table	
<b>Device</b>	<b>iPhone 7</b>	
<b>Action</b>	<b>Expected</b>	<b>Actual</b>

1. Placing a 3D object on the plane	Object sits right on top of surface	<b><i>Slightly floating</i></b>
2. Moving back and forth from the object in a straight horizontal line	Object stays in place without moving	<b><i>Minor jumping and jittering</i></b>
3. Moving the camera up and down	Object stays in place without moving	<b><i>As expected</i></b>
4. Walking around the object	Object stays in place without moving	<b><i>As expected</i></b>
5. Re-focussing the object after tilting camera away	Object stays in place without moving	<b><i>As expected</i></b>
<b>Not recommended</b>		
<b>Device</b>	<b>iPhone X</b>	
<b>Action</b>	<b>Expected</b>	<b>Actual</b>
1. Placing a 3D object on the plane	Object sits right on top of surface	<b><i>As expected</i></b>
2. Moving back and forth from the object in a straight horizontal line	Object stays in place without moving	<b><i>As expected</i></b>
3. Moving the camera up and down	Object stays in place without moving	<b><i>As expected</i></b>

4. Walking around the object	Object stays in place without moving	<b><i>As expected</i></b>
5. Re-focussing the object after tilting camera away	Object stays in place without moving	<b><i>As expected</i></b>
<b>Passed</b>		
<b>Device</b>	<b>Samsung Tab 3</b>	
<b>Action</b>	<b>Expected</b>	<b>Actual</b>
1. Placing a 3D object on the plane	Object sits right on top of surface	<b><i>Major floating</i></b>
2. Moving back and forth from the object in a straight horizontal line	Object stays in place without moving	<b><i>Object frequently drifts and jumps</i></b>
3. Moving the camera up and down	Object stays in place without moving	<b><i>Object sometimes drifts and jumps</i></b>
4. Walking around the object	Object stays in place without moving	<b><i>Object sometimes drifts and jumps</i></b>
5. Re-focussing the object after tilting camera away	Object stays in place without moving	<b><i>Object sometimes drifts and jumps</i></b>
<b>Not Passed</b>		

Figure 18: ARFoundation object persistence with plane tracking test results



3.2.4.2 ARFoundation Model Persistence with Anchor Points

Accuracy of Tracking			
Priority	High	Description	Ensuring that 3D content placed in the world stays persistently in place from all viewpoints under different conditions.
Date	11/04/2020	Performed by	Student
Test Software	ARFoundation	Test Devices	Apple iPhone 7 Apple iPhone X Samsung Tablet Galaxy Tab 3

TEST PROCEDURE		
Condition	Bright natural light, Structured flooring/carpet	
Device	iPhone 7	
Action	Expected	Actual
1. Placing a 3D object on the plane	Object sits right on top of surface	<i>As expected</i>
2. Moving back and forth from the object in a straight horizontal line	Object stays in place without moving	<i>Minor drifting</i>
3. Moving the camera up and down	Object stays in place without moving	<i>As expected</i>
4. Walking around the object	Object stays in place without moving	<i>Minor drifting</i>
5. Re-focussing the object after tilting camera away	Object stays in place without moving	<i>As expected</i>
Passed		
Device	iPhone X	
Action	Expected	Actual
1. Placing a 3D object on the plane	Object sits right on top of surface	<i>As expected</i>
2. Moving back and forth from the object in a straight horizontal line	Object stays in place without moving	<i>As expected</i>

3. Moving the camera up and down	Object stays in place without moving	<b><i>As expected</i></b>
4. Walking around the object	Object stays in place without moving	<b><i>As expected</i></b>
5. Re-focussing the object after tilting camera away	Object stays in place without moving	<b><i>As expected</i></b>
<b>Passed</b>		
<b>Device</b>	<b>Samsung Tab 3</b>	
<b>Action</b>	<b>Expected</b>	<b>Actual</b>
1. Placing a 3D object on the plane	Object sits right on top of surface	<b><i>Slightly floating</i></b>
2. Moving back and forth from the object in a straight horizontal line	Object stays in place without moving	<b><i>Jumping and drifting</i></b>
3. Moving the camera up and down	Object stays in place without moving	<b><i>Jumping and drifting</i></b>
4. Walking around the object	Object stays in place without moving	<b><i>Jumping and drifting</i></b>
5. Re-focussing the object after tilting camera away	Object stays in place without moving	<b><i>Jumping and drifting</i></b>
<b>Not passed</b>		
<b>Condition</b>	Low Artificial light Structured wooden table	
<b>Device</b>	<b>iPhone 7</b>	

Action	Expected	Actual
1. Placing a 3D object on the plane	Object sits right on top of surface	<b><i>Floats slightly above</i></b>
2. Moving back and forth from the object in a straight horizontal line	Object stays in place without moving	<b><i>Minor jumping</i></b>
3. Moving the camera up and down	Object stays in place without moving	<b><i>Minor jittering</i></b>
4. Walking around the object	Object stays in place without moving	<b><i>Minor jittering</i></b>
5. Re-focussing the object after tilting camera away	Object stays in place without moving	<b><i>As expected</i></b>
<b>Not recommended</b>		
Device	iPhone X	
Action	Expected	Actual
1. Placing a 3D object on the plane	Object sits right on top of surface	<b><i>As expected</i></b>
2. Moving back and forth from the object in a straight horizontal line	Object stays in place without moving	<b><i>As expected</i></b>
3. Moving the camera up and down	Object stays in place without moving	<b><i>As expected</i></b>
4. Walking around the object	Object stays in place without moving	<b><i>As expected</i></b>

5. Re-focussing the object after tilting camera away	Object stays in place without moving	<b><i>As expected</i></b>
<b>Passed</b>		
<b>Device</b>	<b>Samsung Tab 3</b>	
<b>Action</b>	<b>Expected</b>	<b>Actual</b>
1. Placing a 3D object on the plane	Object sits right on top of surface	<b><i>Major floating</i></b>
2. Moving back and forth from the object in a straight horizontal line	Object stays in place without moving	<b><i>Major jumping</i></b>
3. Moving the camera up and down	Object stays in place without moving	<b><i>Object sometimes drifts and jumps</i></b>
4. Walking around the object	Object stays in place without moving	<b><i>Object sometimes drifts and jumps</i></b>
5. Re-focussing the object after tilting camera away	Object stays in place without moving	<b><i>Object sometimes drifts and jumps</i></b>
<b>Not Passed</b>		

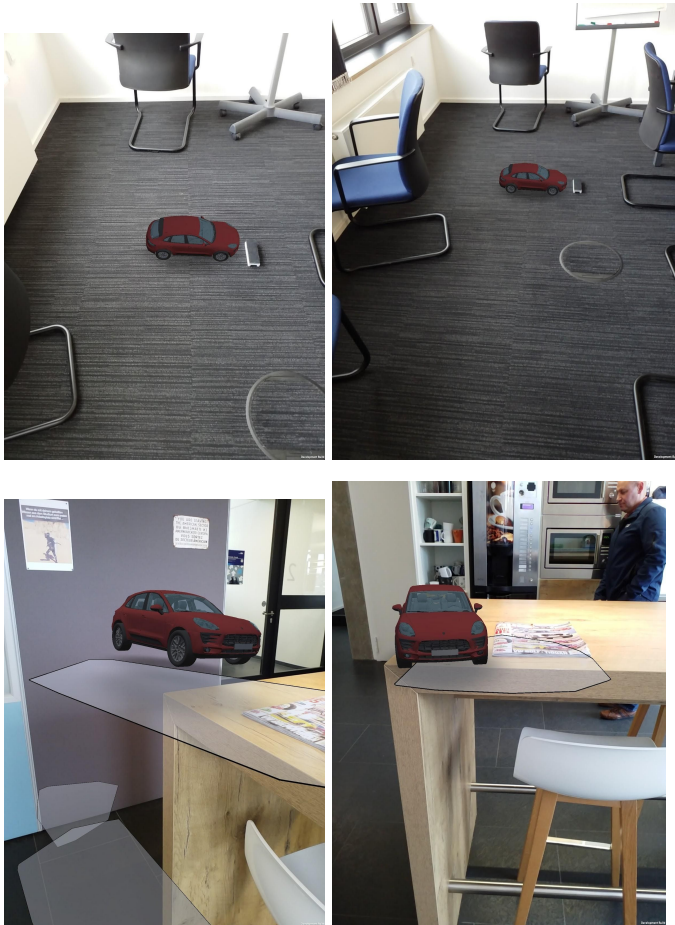
Figure 19: ARFoundation object persistence with anchor points test results

### 3.2.5. Evaluation

Throughout all test iterations and setups the Samsung tablet yields a very unstable result. The object rarely stays in its position and constantly drifts across the plane.

This is certainly mostly due to the poor camera quality.

Because of the hardware limitations it is up for speculations if a more recent Android device would return more reliable results.



*Figure 20: Samsung Galaxy Tab 3 Tablet ARFoundation*

On more structured surfaces in natural lighting conditions the failure rate was, as expected, considerably lower than unstructured surfaces. Both iPhone 7 and X on the other hand succeeded in placing the content on the table accurately even in difficult low feature point conditions.



*Figure 21: iPhone 7 + ARFoundation*

*Figure 22: iPhone X + ARFoundation*

It is noteworthy that the slightly older iPhone 7 tracks the object with considerably less jittering than the iPhone X. The iPhone X was constantly re-focussing the camera and thus briefly lost track of the feature points. The disturbance however is only minor and does not impact the overall performance. Even though the Samsung Tab 3 was able to place the car on the table as well, the result was not convincing. The model was floating above the table and did not stay anchored in place whatsoever. In general it can be said that AR tracking works best in natural lighting conditions throughout all devices. The more prominent the feature points, the more accurate and reliable the tracking. Better camera quality has a direct impact on the accuracy of tracking as can be seen in the Samsung Tab 3 example.



*Figure 23: iPhone 7 positioning throughout test cases*



*Figure 24: iPhone X positioning throughout test cases*

Surprisingly, the iPhone X running on iOS 11.3 did show some minor jittering as the camera tried to refocus, resulting in minor positional change of the object. This however did not disturb the overall experience too much. The iPhone 7 as the older iPhone model running on

iOS 11 excelled at tracking the object reliably and without positional changes throughout all test cases in natural lighting. The anchoring did become unstable in artificial lighting conditions as was the case for the other devices as well:



Figure 25: iPhone 7 tracking in dim artificial lighting

It can be concluded that even though the exact same app was deployed on 3 different devices, the results differ greatly based on the hardware prerequisites. The Samsung Galaxy Tab 3 tablet turned out to be unsuited for augmented reality use cases and cannot be recommended. In both ARFoundation and ARCore test sessions the object is not anchored properly. This is most likely due to its poor camera quality and by no means a drawback of ARFoundation as opposed to ARCore. This is furthermore confirmed when regarding the results of the other test devices:

The iPhone X finds planes fast even in low lighting conditions. However when it comes to tracking it showed minor issues in refocusing the camera. These issues are not preventing a smooth experience however. The iPhone 7 yields by far the most reliable result in all test cases. The object stayed anchored on the plane securely at all times. This confirms that the ARFoundation framework is suitable for the requested use case of an AR showroom, provided the environment reflects the recommended standards for AR. For an optimal anchoring of the object the environment should be naturally lit and the surface to be tracked should provide enough distinguishable feature points to be picked up by the camera. Furthermore the device used should be no older than 2017.

### Results ARPlanes vs ARAnchors

There was no noticeable difference between the two methods. The official documentation of ARFoundation however recommends to use plane tracking whenever possible as it requires less resources to anchor the object onto its position.



## 4. Model Optimization

Unit Requirement <a href="#">[SR04-M]</a> <a href="#">[SR05-M]</a>	Model Optimization for AR	Description <i>Improve Model Tracking</i>
Acceptance Criteria	Model that is used must be optimized to a degree that ensures smooth reliable tracking. Model used must be optimized to reduce the size of the app	
State	MET	

### 4.1 Test Cases

Model optimization plays an important role in mobile development. Especially in markerless tracking the model needs to be low poly and optimized to a degree that ensures smooth, easy rendering without requiring too much computational resources that would impact the tracking quality. In the following, different models and degrees of optimization have been tested to see to what degree a model has to be optimized. Note that the result varies heavily with the CPU and GPU power of a device as well as available storage space. To see the specifications of the test devices please refer to chapter 1.2.1 Test Devices

### 4.2 Scope

For ease of testing only optimal lighting conditions are assumed. The tests are facilitated under natural lighting on a structured horizontal surface. All previously established requirements to ensure smooth tracking are applied.

The following points are subject of observation:

- Tracking stability of the model in relation to poly count

4.3 Test Conditions

4.3.1 Environment

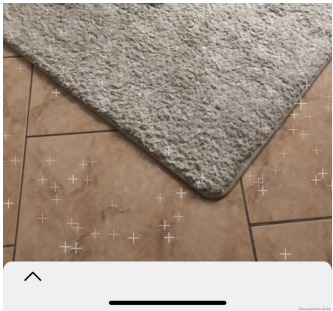
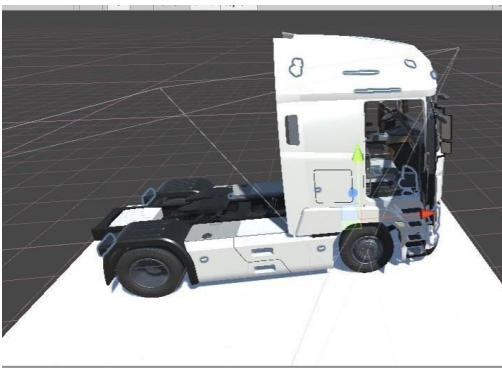
Light Condition	Surface	Reference
Bright natural light	Light structured carpet	

Figure 26: Model Optimization Environment Conditions

4.3.2 Models

Model Name/Source	Poly Count/Object count	Reference	Mesh
Truck (IAV intern)	5.345.600 700 objects		No Mesh image available, computer not powerful enough to open the FBX in Maya




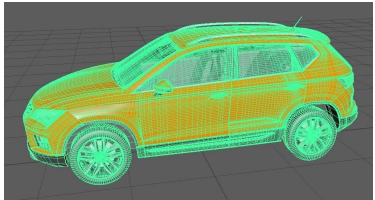

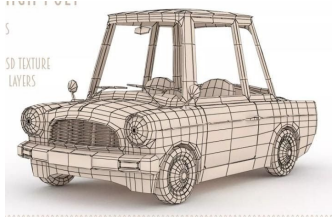
Seat Ateka high poly (hum3D provided by IAV)	1.338.000 240 objects		
Seat Ateka reduced (optimized by student)	500.000 15 objects		
Cicada cartoon car (Unity asset store for comparison)	6691 Poly 14 objects		

Figure 27: Test Models

#### 4.4 Test Procedures

The test application was deployed on the device.  
The test application contains no additional features but placing an object on a plane.  
The tests were facilitated in the same optimal lighting and tracking conditions.

## 4.5 Test Results

### 4.5.1 Truck

Model Optimization			
Priority	High	Description	Ensuring that 3D content placed in the world stays persistently in place from all viewpoints under different conditions.
Date	15/04/2020	Performed by	Student
Test Software	ARFoundation	Test Devices	Apple iPhone 7 Apple iPhone X Samsung Tablet Galaxy Tab 3
TEST PROCEDURE			
Condition		Bright natural light, Structured flooring/carpet	
Device		iPhone 7	
Action		Expected	Actual

1. Placing a 3D object on the plane	Object sits right on top of surface	<b><i>As expected</i></b>
2. Moving back and forth from the object in a straight horizontal line	Object stays in place without moving	<b><i>Major drifting and jittering</i></b>
3. Moving the camera up and down	Object stays in place without moving	<b><i>Major drifting and jittering</i></b>
4. Walking around the object	Object stays in place without moving	<b><i>Major drifting and jittering</i></b>
5. Re-focussing the object after tilting camera away	Object stays in place without moving	<b><i>Major drifting and jittering</i></b>
<b>Not Passed</b>		
<b>Device</b>	<b>iPhone X</b>	
<b>Action</b>	<b>Expected</b>	<b>Actual</b>
1. Placing a 3D object on the plane	Object sits right on top of surface	<b><i>As expected</i></b>
2. Moving back and forth from the object in a straight horizontal line	Object stays in place without moving	<b><i>Major drifting and jittering</i></b>
3. Moving the camera up and down	Object stays in place without moving	<b><i>Major drifting and jittering</i></b>
4. Walking around the object	Object stays in place without moving	<b><i>Major drifting and jittering</i></b>

5. Re-focussing the object after tilting camera away	Object stays in place without moving	<b><i>Major drifting and jittering</i></b>
<b>Not Passed</b>		
<b>Device</b>	<b>Samsung Tab 3</b>	
<b>Action</b>	<b>Expected</b>	<b>Actual</b>
1. Placing a 3D object on the plane	Object sits right on top of surface	<b><i>Slightly floating</i></b>
2. Moving back and forth from the object in a straight horizontal line	Object stays in place without moving	<b><i>Jumping and drifting</i></b>
3. Moving the camera up and down	Object stays in place without moving	<b><i>Jumping and drifting</i></b>
4. Walking around the object	Object stays in place without moving	<b><i>Jumping and drifting</i></b>
5. Re-focussing the object after tilting camera away	Object stays in place without moving	<b><i>Jumping and drifting</i></b>
<b>Not passed</b>		

Figure 28: Test Results Truck

#### 4.5.2 Seat Ateca

Model Optimization			
<b>Model</b>	<i>Seat Ateca high poly</i>	<b>Description</b>	<i>Ensuring that 3D content placed in the world stays persistently in place from all viewpoints under different conditions.</i>
<b>Date</b>	<i>02/03/2020</i>	<b>Performed by</b>	<i>Student</i>
<b>Test Software</b>	<i>ARFoundation</i>	<b>Test Devices</b>	<i>Apple iPhone 7 Apple iPhone X Samsung Tablet Galaxy Tab 3</i>
TEST PROCEDURE			
<b>Condition</b>	Bright natural light, Structured flooring/carpet		
<b>Device</b>	<b>iPhone 7</b>		
<b>Action</b>	<b>Expected</b>	<b>Actual</b>	
1. Placing a 3D object on the plane	Object sits right on top of surface	<b><i>As expected</i></b>	
2. Moving back and forth from the object in a straight horizontal line	Object stays in place without moving	<b><i>Minor drifting</i></b>	
3. Moving the camera up and down	Object stays in place without moving	<b><i>Minor drifting</i></b>	
4. Walking around the object	Object stays in place without moving	<b><i>As expected</i></b>	

5. Re-focussing the object after tilting camera away	Object stays in place without moving	<b>Minor drifting</b>
<b>Not Passed</b>		
<b>Device</b>	<b>iPhone X</b>	
<b>Action</b>	<b>Expected</b>	<b>Actual</b>
1. Placing a 3D object on the plane	Object sits right on top of surface	<b>As expected</b>
2. Moving back and forth from the object in a straight horizontal line	Object stays in place without moving	<b>Minor drifting</b>
3. Moving the camera up and down	Object stays in place without moving	<b>As expected</b>
4. Walking around the object	Object stays in place without moving	<b>As expected</b>
5. Re-focussing the object after tilting camera away	Object stays in place without moving	<b>Minor drifting</b>
<b>Not Recommended</b>		
<b>Device</b>	<b>Samsung Tab 3</b>	
<b>Action</b>	<b>Expected</b>	<b>Actual</b>
1. Placing a 3D object on the plane	Object sits right on top of surface	<b>Slightly floating</b>
2. Moving back and forth from the object in a straight horizontal line	Object stays in place without moving	<b>Jumping and drifting</b>
3. Moving the camera up and down	Object stays in place without moving	<b>Jumping and drifting</b>
4. Walking around the object	Object stays in place without moving	<b>Jumping and drifting</b>



5. Re-focussing the object after tilting camera away	Object stays in place without moving	<b><i>Jumping and drifting</i></b>
<b>Not passed</b>		

Figure 29: Test Results Seat Ateca High Poly

#### 4.5.3 Seat Ateca Low Poly

Model Optimization			
Model	Seat Ateka low poly	Description	Ensuring that 3D content placed in the world stays persistently in place from all viewpoints under different conditions.
Date	05/05/2020	Performed by	Student
Test Software	ARFoundation	Test Devices	Apple iPhone 7 Apple iPhone X Samsung Tablet Galaxy Tab 3
TEST PROCEDURE			
Condition		Bright natural light, Structured flooring/carpet	
Device		iPhone 7	
Action		Expected	Actual
1. Placing a 3D object on the plane		Object sits right on top of surface	As expected
2. Moving back and forth from the object in a straight horizontal line		Object stays in place without moving	As expected

3. Moving the camera up and down	Object stays in place without moving	<i>As expected</i>
4. Walking around the object	Object stays in place without moving	<i>As expected</i>
5. Re-focussing the object after tilting camera away	Object stays in place without moving	<i>As expected</i>
<b>Passed</b>		
<b>Device</b>	<b>iPhone X</b>	
<b>Action</b>	<b>Expected</b>	<b>Actual</b>
1. Placing a 3D object on the plane	Object sits right on top of surface	<i>As expected</i>
2. Moving back and forth from the object in a straight horizontal line	Object stays in place without moving	<i>As expected</i>
3. Moving the camera up and down	Object stays in place without moving	<i>As expected</i>
4. Walking around the object	Object stays in place without moving	<i>As expected</i>
5. Re-focussing the object after tilting camera away	Object stays in place without moving	<i>As expected</i>
<b>Passed</b>		
<b>Device</b>	<b>Samsung Tab 3</b>	
<b>Action</b>	<b>Expected</b>	<b>Actual</b>
1. Placing a 3D object on the plane	Object sits right on top of surface	<i>As expected</i>

2. Moving back and forth from the object in a straight horizontal line	Object stays in place without moving	<b><i>Jumping and drifting</i></b>
3. Moving the camera up and down	Object stays in place without moving	<b><i>Slightly drifting</i></b>
4. Walking around the object	Object stays in place without moving	<b><i>Slightly drifting</i></b>
5. Re-focussing the object after tilting camera away	Object stays in place without moving	<b><i>Slightly drifting</i></b>
<b>Not passed</b>		

Figure 30: Test Results Seat Ateca Low Poly

#### 4.5.4 Cicada

Model Optimization			
<b>Model</b>	<b><i>Cicada high poly</i></b>	<b>Description</b>	<i>Ensuring that 3D content placed in the world stays persistently in place from all viewpoints under different conditions.</i>
<b>Date</b>	<i>17/03/2020</i>	<b>Performed by</b>	<i>Student</i>

Test Software	ARFoundation	Test Devices	Apple iPhone 7 Apple iPhone X Samsung Tablet Galaxy Tab 3
TEST PROCEDURE			
Condition	Bright natural light, Structured flooring/carpet		
Device	iPhone 7		
Action	Expected	Actual	
1. Placing a 3D object on the plane	Object sits right on top of surface	As expected	
2. Moving back and forth from the object in a straight horizontal line	Object stays in place without moving	As expected	
3. Moving the camera up and down	Object stays in place without moving	As expected	
4. Walking around the object	Object stays in place without moving	As expected	
5. Re-focussing the object after tilting camera away	Object stays in place without moving	As expected	
Passed			
Device	iPhone X		
Action	Expected	Actual	

1. Placing a 3D object on the plane	Object sits right on top of surface	<i>As expected</i>
2. Moving back and forth from the object in a straight horizontal line	Object stays in place without moving	<i>As expected</i>
3. Moving the camera up and down	Object stays in place without moving	<i>As expected</i>
4. Walking around the object	Object stays in place without moving	<i>As expected</i>
5. Re-focussing the object after tilting camera away	Object stays in place without moving	<i>As expected</i>
<b>Passed</b>		
<b>Device</b>	<b>Samsung Tab 3</b>	
<b>Action</b>	<b>Expected</b>	<b>Actual</b>
1. Placing a 3D object on the plane	Object sits right on top of surface	<i>As expected</i>
2. Moving back and forth from the object in a straight horizontal line	Object stays in place without moving	<i>Minor jittering</i>
3. Moving the camera up and down	Object stays in place without moving	<i>Minor jittering</i>
4. Walking around the object	Object stays in place without moving	<i>Minor jittering</i>

5. Re-focussing the object after tilting camera away	Object stays in place without moving	<b>Minor jittering</b>
<b>Not passed</b>		

Figure 31: Test Results Cicada

#### 4.6 Evaluation

It can be concluded that the lower the poly count and count of separate objects the better is the tracking quality and persistence especially in lower end devices. This is due to the high processing power required to render high poly models with a lot of separate objects on screen. Furthermore for an application that should be published the app size should not be neglected. By keeping the model size small the application will also be moderately sized. While the application with the Truck had 1,3GB in size, with the low poly Cicasa model it could be reduced to only 35MB. Because the CPU uses separate draw calls for single objects it is important to keep in mind to combine objects not required separately when optimizing the model. The Seat Ateca for example featured 240 separate objects. However only doors and wheels were required for separate animations, the other objects could be combined to a single mesh, reducing processing power required. The Samsung Galaxy Tab 3 as the lowest end device did not yield good results with any of the high poly models. The lower the poly and object count however the better the tracking experience was even on the not so powerful device. Another aspect to consider when optimizing models for AR use is the problem separate objects might cause with light estimation. Separate objects will be calculated and lit separately resulting in individual lighting conditions for each object.



Figure 42: Light estimation on separate objects (door and body)

It should therefore be considered carefully which objects are really required as separate objects for interaction to ensure even lighting.

## 5. Multi Platform UI

Unit Requirement <a href="#">[SR02-M]</a>	Multi Platform UI	Description <i>Dynamic UI scaling</i>
Acceptance Criteria	UI must adapt to all screen sizes and resolutions in Portrait mode	
State	MET	

Unit Requirement <a href="#">[SR01-S]</a>	Multi Platform UI	Description <i>Dynamic UI scaling</i>
Acceptance Criteria	UI must adapt to all screen sizes and resolutions in Landscape mode	
State	NOT MET	

### 5.1 Test Cases

A cross platform approach enables applications to be deployed on a wide range of devices throughout iOS and Android. This requires the UI system to be able to scale to all screen resolutions, both vertical and horizontal. There are currently hundreds of different devices with different screen sizes. There is no common way of fixing these issues in Unity. The approach has to be considered individually for each use case. There are however a number of general best practices recommended by Unity officials and its community of developers. The following documentation demonstrates and explains the iteration steps taken while following the general recommendations.

### 5.2 Scope

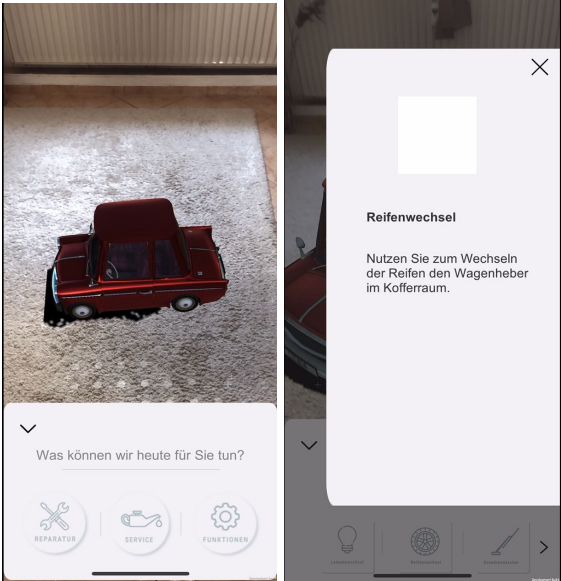
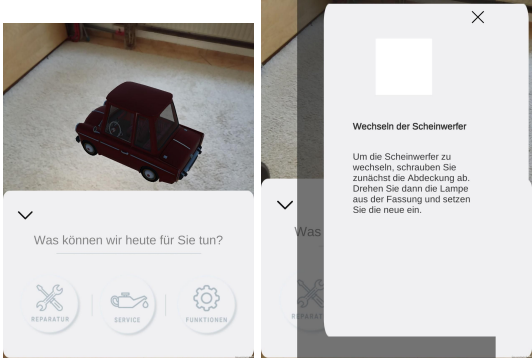
For this use case only the general approaches and best practices are applied technically. The content, design and positioning of the UI is secondary and would require user testing for verification and iteration in relation to the overall application content.

5.3 Test Procedure

For each iteration the app is deployed on the target device and run in both portrait and landscape mode. The iterations are evaluated based on the accurate positioning and scaling of the UI content.

5.4 Results

5.4.1. Iteration I - Not optimized UI

Not optimized UI Iteration I	
Portrait	
iPhone X	Samsung Tablet
	
Landscape	



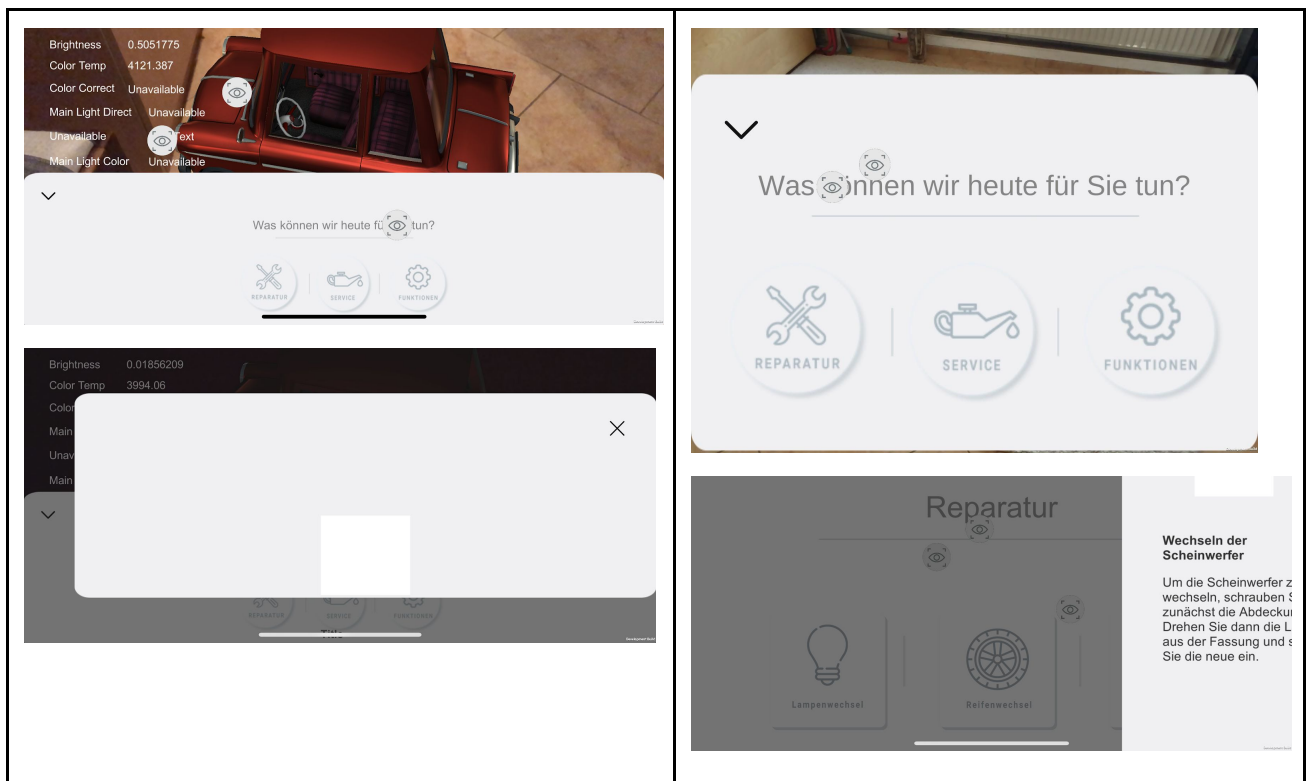


Figure 32: UI Iteration I

In the first iteration the UI was only spaced evenly on the iPhone X in portrait mode as this was the resolution while developing. No best practices were applied. UI orientation and positioning was created by setting pixel values none relative to the screen size. Therefore the content did not scale properly on any other device but the iPhone X.

5.4.2 Iteration II

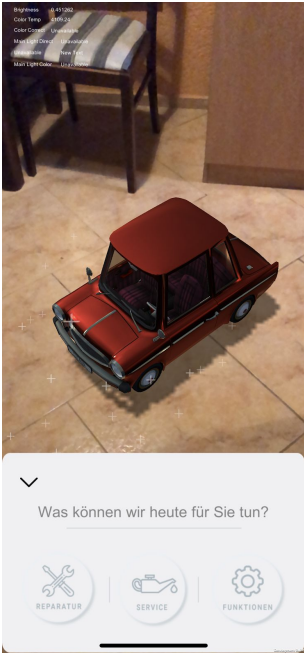

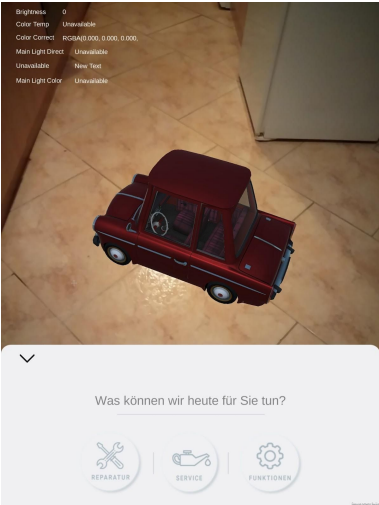
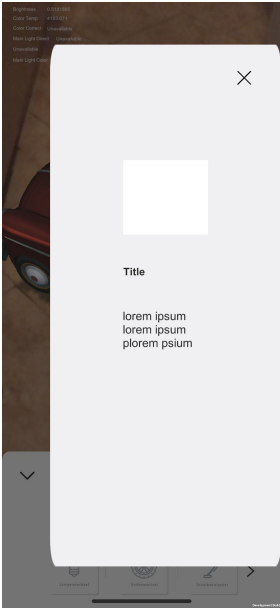
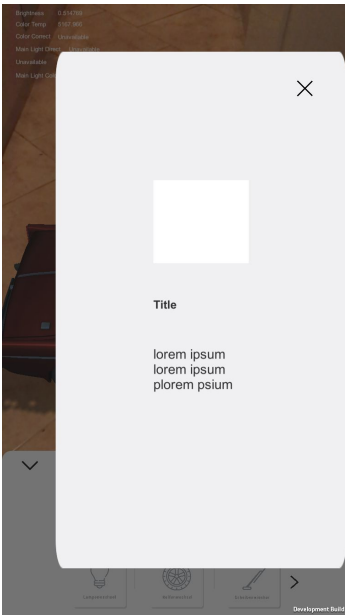
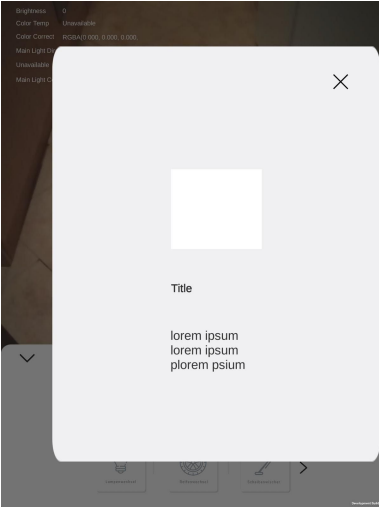
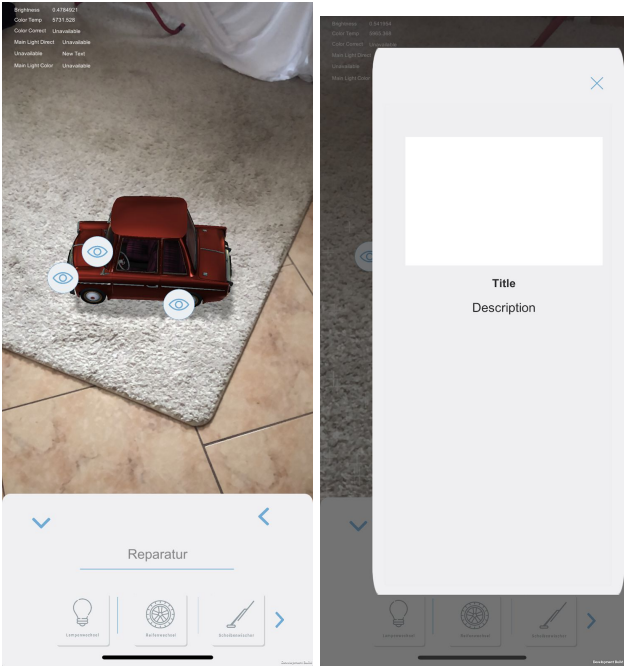
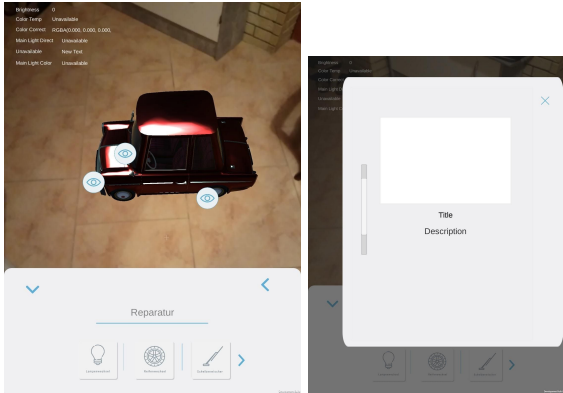
Optimized UI Iteration II		
Portrait		
iPhone X	iPhone 7	Samsung Tablet
 <p>UI mockup for iPhone X. The top status bar shows system settings like Brightness, Color Temp, and Color Correction. The main content area displays a 3D model of a red car on a tiled floor. Below the car is a white overlay with a close button (chevron down), the text "Was können wir heute für Sie tun?", and three circular icons labeled REPARATUR, SERVICE, and FUNKTIONEN.</p>	 <p>UI mockup for iPhone 7. The top status bar shows system settings. The main content area displays a 3D model of a red car with three eye icons above it. Below the car is a white overlay with a close button (chevron down), the title "Reparatur", and three icons: a lightbulb, a gear, and a pencil, followed by a chevron right. A "Development Build" label is at the bottom right.</p>	 <p>UI mockup for Samsung Tablet. The top status bar shows system settings. The main content area displays a 3D model of a red car. Below the car is a white overlay with a close button (chevron down), the text "Was können wir heute für Sie tun?", and three circular icons labeled REPARATUR, SERVICE, and FUNKTIONEN.</p>
 <p>UI mockup for iPhone X showing a modal dialog. The dialog has a close button (X) in the top right corner, a white square placeholder, the title "Title", and the text "lorem ipsum", "lorem ipsum", and "plorem psium".</p>	 <p>UI mockup for iPhone 7 showing a modal dialog. The dialog has a close button (X) in the top right corner, a white square placeholder, the title "Title", and the text "lorem ipsum", "lorem ipsum", and "plorem psium".</p>	 <p>UI mockup for Samsung Tablet showing a modal dialog. The dialog has a close button (X) in the top right corner, a white square placeholder, the title "Title", and the text "lorem ipsum", "lorem ipsum", and "plorem psium".</p>

Figure 33: UI Iteration II

The best practices as described in the thesis chapter 6.2.3 *UI Iteration* were applied. By positioning the UI with the help of anchors the content scales relative to the screen size. In landscape mode however the window still shrinks proportionally with the horizontal resolution causing the text box to not be fully visible. In order to solve this the content in landscape mode needed a *scroll rect* component to enable to see the whole content by scrolling the text box.

5.4.3 UI Iteration III

Optimized UI Iteration III	
Portrait	
iPhone X	Samsung Tablet
	
Landscape	

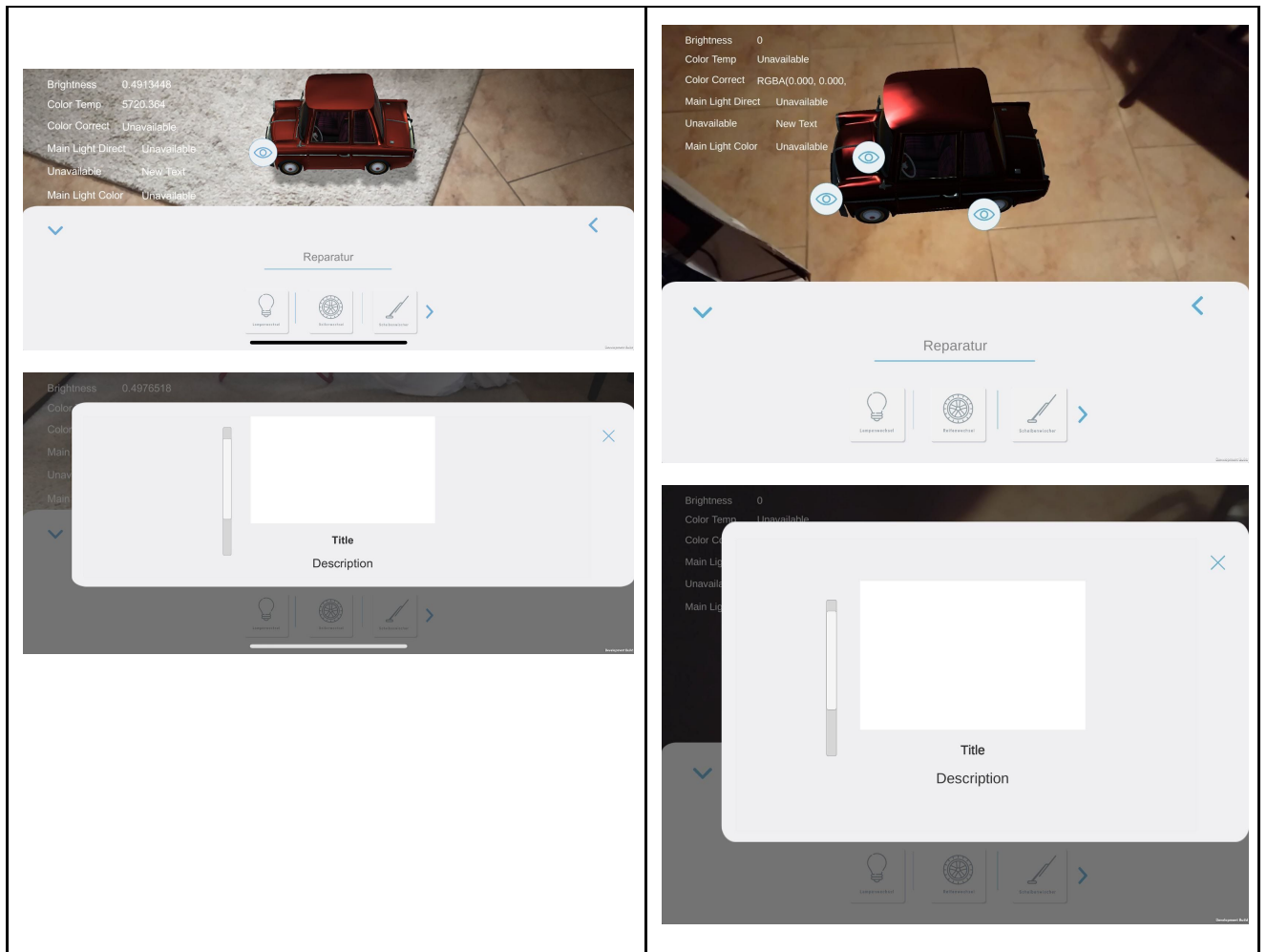


Figure 34: UI Iteration III

The content now scales proportionally in portrait and landscape mode on different devices. If the text box is not completely visible with the device's resolution a scrollbar enables scrolling the content.

## 5.5 Evaluation

Creating a responsive UI that adapts to all screen resolutions is challenging even in 2D applications.

The layout specifics and focus of the application need to be considered carefully for each use case and the UI design and layout chosen accordingly. In scenes where textual information is not the main focus it is easier to adapt to both portrait and landscape orientations as the buttons and objects only take up minimal space and can be anchored easily to the canvas and still provide equally good performance in both portrait and landscape mode. Generally, it is not advised to display important textual information directly in AR world space. It makes it difficult to focus on the text. Instead text should be displayed in a 2D UI. It is evident that for this use case a support for both portrait and landscape mode is not optimal as the textual information can not be read properly in landscape mode. The initially agreed upon system requirements [SR01-S] and [SR02-M] of the content having to scale in both portrait and

landscape mode has been revised in correspondence with the company. For this specific use case of displaying a lot of text a landscape orientation is suboptimal as the user can only see a small part of the text without scrolling.

Contrary to the intuitive positioning of most users of the device horizontally when interacting with AR apps, it has been decided to instead lock the orientation in portrait mode which makes it easier for the user to read the information instead.

## 6. Light Estimation

Unit Requirement <a href="#">[SR07-M]</a>	Light Estimation	Description <i>Dynamic lighting of a scene</i>
Acceptance Criteria	Feature dynamic lighting and shadow of the object based on the lighting of the real world to increase the immersiveness of a scene	
State	MET	

### 6.1 Test Cases

Lighting plays an important role in creating a realistic and convincing scene.

ARFoundation features Light Estimation as a way to adapt the virtual lighting to the real world lighting conditions. Testing is performed in 3 separate steps:

1. In the first step the general recognition of the real world environment lighting is tested on different devices by deploying the sample scene provided by Unity. The sample scene features a UI interface on which the detected values are outputted. The sample project can be found under:  
<https://github.com/Unity-Technologies/arfoundation-samples>
2. In a second step these received values are applied to the virtual lighting in the scene to sync the virtual and the real world light and make the virtual object appear to adapt to the lighting.
3. In a third step the impact of realtime reflection probes on realism is tested. Reflection probes serve as a way to project the real world camera feed onto reflective surfaces of the virtual object and make it appear as if it were reflecting the real environment

The light estimation feature is tested on a range of test devices to determine the usability for a cross platform approach.

### Acceptance criteria:

Virtual lighting of the object dynamically adapts with the real world lighting

## 6.2 Scope

The Light estimation feature is tested on the iPhone X, iPhone 7 and Samsung Tab3. The tests are facilitated under different lighting conditions to estimate how the virtual light adapts.

Since tracking persistence is secondary the surface used for tracking is not mentioned explicitly.

The following points are subject of observation:

- Response of the device in recognizing lighting conditions
- Response of the model to a change in lighting conditions
- Application of realtime reflections on reflective materials on the virtual model
- Adaption of color correction on the virtual model in different lighting conditions
- Overall adaptation of light estimation on different devices

## 6.3 Test Results

### 6.3.1 Light Estimation Values

The Sample project for LightEstimation can be found under:

<https://github.com/Unity-Technologies/arfoundation-samples>


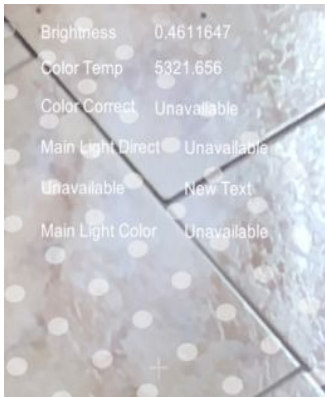



Light Estimation Values		
iPhone X	iPhone 7	Samsung Tablet
		

Figure 35: Light estimation values

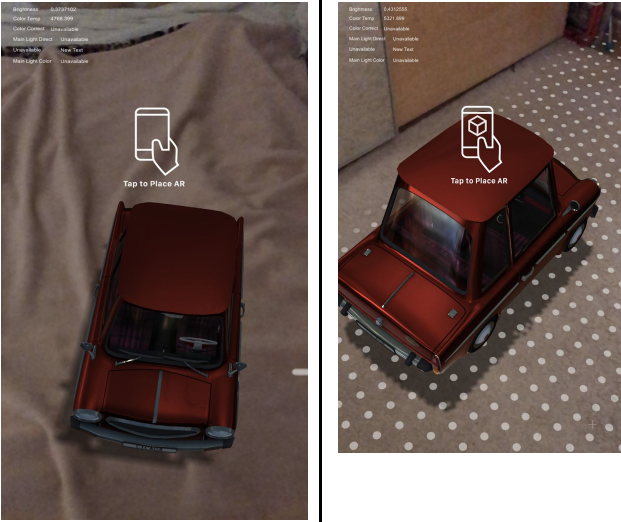
Both iPhone X and iPhone 7 recognize the prevailing lighting situation and output the detected values to the screen. The brightness is estimated between a value of 0 to 1, where 0 represents dark and 1 represents light. In a dark environment the brightness value on the iPhone X detected 0.398013.

In a lighter environment the iPhone 7 detected 0,4611647. The Samsung Tab 3 apparently does not support any form of light estimation and did not output any values. As a second value the iPhones detected the overall color temperature of the environment, with values < 5000 representing warm tones and values > 5000 representing cool tones. While the iPhone X detected 4654,793 in a relatively dark scene, the iphone 7 detected 5321,656 in a lighter scene. In both images cooler undertones can be observed. The Samsung Tab 3 did not yield any color temperature results, suggesting that this feature is unavailable on this device. The detected values for brightness and color temperature can now be applied to the virtual scene lighting to match it to the real world lighting and thus have the model be lit in correspondence with the environment.

### 6.3.2 Applied Light Estimation Values

Natural Bright Light			
	iPhone X	iPhone 7	Samsung Tablet
Brightness value	0,48	0,57	not supported
Color Temperature	5739	5735	not supported
Screenshot			



Natural Low Light			
	iPhone X	iPhone 7	Samsung Tablet
Brightness value	0,373	0,4312	not supported
Color Temperature	4766,399	5321	not supported
Screenshot			

Warm Artificial Light			
	iPhone X	iPhone 7	Samsung Tablet
Brightness value	0,48	0,422	not supported
Color Temperature	4109,24	4115,12	not supported



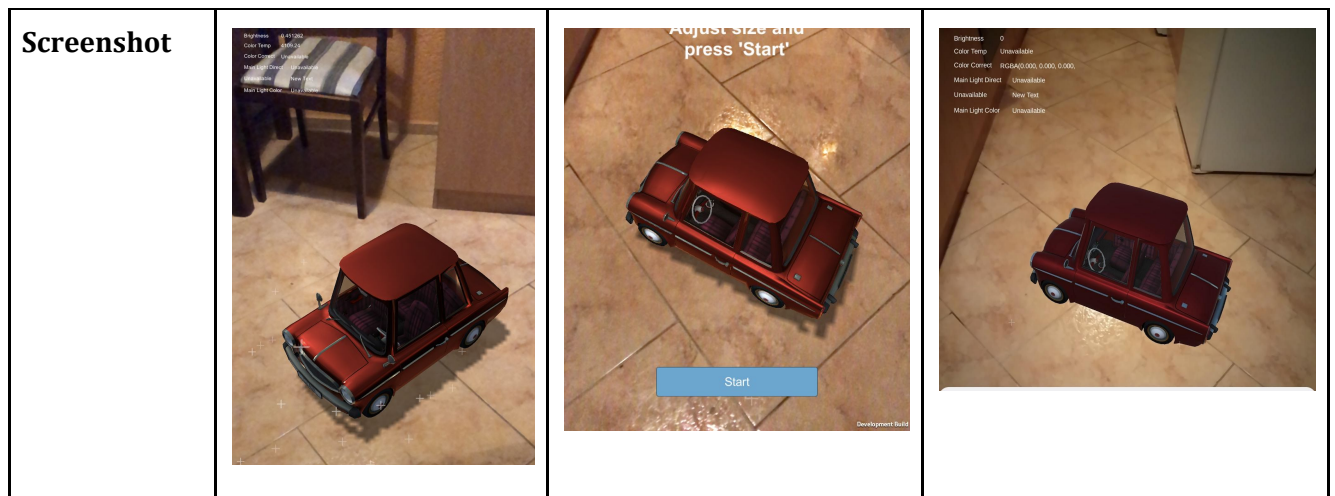




Figure 36: Applied light estimation values

The iPhones detect the values very well and thus the object seems lit correctly. In darker environments the object looks darker and in light environments the object looks well lit. It is also clear to see that in an environment in which color temp is detected as  $< 5000$ , meaning warm lighting, the object displays more yellow/orange hues because the value is applied to the scene lighting. In contrast, environments with detected color temperature values  $> 5000$  the object seems to be displayed in more blue undertones as the virtual light adapts to the environment. The model on the Samsung device appears unlit because the default value of 0 is applied to the virtual light, meaning the virtual brightness is set to 0.

### 6.3.3 Reflection Probes

iPhone X	
Brightness value	0,520059
Color Temperature	5739,587

<b>Screenshot</b>	
<b>iPhone 7</b>	
<b>Brightness Value</b>	<b>0,538698</b>
<b>Color Temperature</b>	<b>5916,422</b>
<b>Screenshot</b>	

*Figure 37: Reflection Probes*

Reflection probes add reflectiveness to reflective surfaces adding another layout of realism.

## 6.4 Evaluation

Light estimation adds an essential layer of realism to the virtual AR scene. It enables the virtual scene to dynamically change with the real world lighting conditions. The dynamic lighting yields convincing results in different light circumstances, helping to make the object blend with the environment. As a more practical application case, by receiving the brightness values on a scale of 0 - 1 it should be possible to also use these values to trigger actions such as playing animations or displaying content based on the real world lighting conditions. This would blend the borders between virtual content and the real world even more. When working with

reflective objects such as metallic cars, reflection probes can increase the immersion even further by applying realtime, realworld reflections to reflective surfaces. This can be achieved by using reflection probes. The texture for the reflection probes can be derived directly from the camera feed. The requirement for reflection probes to return realtime reflections is that the object's textures support the metallic workflow. The reflectivity and light response of the surface are modified by the metallic and smoothness level of the texture.

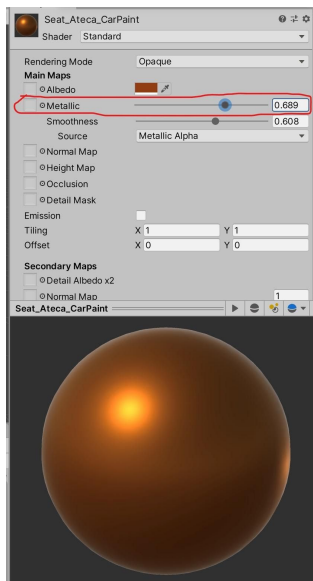


Figure 38: Reflective car material

As stated in the Unity documentation ([Unity Documentation, 2019](https://docs.unity3d.com/2019.4/Manual/MetallicWorkflow.html)): the metallic parameter of a material determines how “metal-like” the surface is. When a surface is more metallic, it reflects the environment more and its albedo colour becomes less visible. At full metallic level, the surface colour is entirely driven by reflections from the environment.

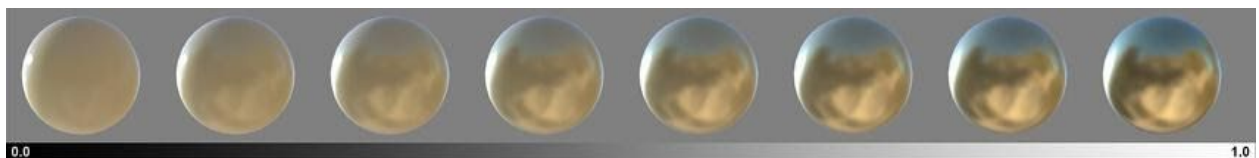


Figure 39: A range of metallic values from 0 to 1 (with smoothness at a constant 0.8 for all samples) ([Unity Documentation, 2019](https://docs.unity3d.com/2019.4/Manual/MetallicWorkflow.html))

The reflection probe can now reflect the texture created from the camera feed onto the model's metallic surface. The application of light estimation and realtime reflections in a cross-platform approach is questionable however. It is clear that iOS devices react very well to this feature. On the Android device no values could be detected at all. It stands to reason whether this is due to the specific device's limitation or if this feature does not function at all on Android devices. It was not possible to investigate this because of the limitation in test devices.

ARFoundation in theory also enables the estimation of the direction of the real world light. This value could be applied to the virtual light as well, resulting in the model being lit not only with the correct brightness and color temperature but also dynamically from the right angle, additionally resulting in a realistic casting of shadows. However this feature requires a true depth camera which at the moment is only available in iPhone X and later as a front facing camera. This means this feature can already be applied in face recognition applications or any other application using the front facing camera as the main camera.

## 7. AR Light and Shadows Support Shaders

Unit Requirement <a href="#">[SR08-M]</a>	Scene Lighting	Description <i>Dynamic lighting of a scene</i>
Acceptance Criteria	Feature an accurate representation of light and shadow the object causes onto the surface so the objects looks properly grounded in AR space	
State	MET	

### 7.1 Approach

In some use cases in mobile augmented reality it might be required to project light and shadows onto transparent geometry. In this particular project, when the car's headlights turn on the light should be reflected on the ground. Since an opaque plane underneath the object would break the immersion of the object being anchored in the real world, the plane onto which the shadow and lighting is cast needs to be transparent. Rendering shadows and light onto transparent geometry requires a special custom shader that is not included in the Unity AR project by default. In this chapter it will be discussed how such a shader can be created. In forward rendering, multi-light shaders use a separate pass for each pixel light in the scene. The shader therefore needs two defined passes. The Base Pass renders the main directional light in the scene, responsible for the shadow of the car. The second pass (the Add pass) gets called once for each additional light, and is additively blended with the previous passes. The base pass of this shader has been adapted from DanMiller's Mobile AR Shader ([Miller, 2019](#)). The base and add passes are marked using the LightMode tag. This tag tells Unity which pass to use for which. The "Forward" prefix on Add and Base identifies that these passes are for Forward rendering. The fallback to VertexLit allows to use the VertexLit shaders shadow passes. Without this the shader would not cast shadows properly.

## 7.2 Results



*Figure 40: Custom shader to render light and shadow on transparent geometry.*

In Unity the car is placed onto a geometric plane with a transparent material onto which the custom shader is applied. With the help of this shader it is not possible to create realistic lighting effects in AR space. In this example in Figure 40 the values for the average brightness received from the light estimation are applied to drive the animation for the headlights. If the environment brightness falls below 0.4 the headlights turn on automatically. This custom shader only supports the standard and lightweight rendering pipeline. It does not support the universal rendering pipeline.

## 8. Final Test Results

### 8.1 Test Cases

In a final summarizing test all features and iterations are evaluated in regards to their integration into a multi-platform project. The goal is to give an indication of which devices support more complex AR use cases.

## 8.2 Test Results

Plane Tracking			
Model	Seat Ateca Low Poly	Criteria	Planes are recognized reliably
Device	Result	State	
iPhone X	Feature points and planes are recognized fast and tracked accurately even in low light/low textured environments	MET	
iPhone 7	Feature points and planes are recognized fast and tracked accurately even in low light/low textured environments	MET	
iPad Pro 2017	Feature points are recognized even in low light/low textured environments but take more time	MET	
Samsung Galaxy Tab 3	Feature points take too long to be recognized. Planes are only tracked in optimal lighting conditions and heavily textured surfaces	NOT MET	
Model Persistence			
Model	Seat Ateca Low Poly	Criteria	Model stays anchored to the plane
Device	Result	State	
iPhone X	Object stays anchored in place provided enough feature points	MET	

	and sufficient lighting are present	
iPhone 7	Object stays anchored in place provided enough feature points and sufficient lighting are present	<b>MET</b>
iPad Pro 2017	Object stays in place in optimal environment conditions but drifts slightly in suboptimal conditions	<b>NOT RECOMMENDED</b>
Samsung Galaxy Tab 3	Model frequently drifts and jumps. Does not provide reliable tracking	<b>NOT MET</b>
<b>Light Estimation</b>		
<b>Model</b>	<b>Seat Ateca Low Poly</b>	<b>Criteria</b> Virtual lighting changes dynamically with the real world lighting conditions
<b>Device</b>	<b>Result</b>	<b>State</b>
iPhone X	Object lighting changes dynamically with the real world condition. Light direction not available yet	<b>MET</b>
iPhone 7	Object lighting changes dynamically with the real world condition. Light direction not available yet	<b>MET</b>
iPad	Object lighting changes dynamically with the real world condition. Light direction not available yet	<b>MET</b>

Samsung Galaxy Tab 3	No light estimation values detected	NOT MET	
Light Driven Shader Based Animations			
Model	Seat Ateca Low Poly	Criteria	The headlights are driven by the received value from average brightness. Shadow and Lights are projected accurately in AR space
Device	Result	State	
iPhone X	Animation is driven by light estimation value and lighting and shadow are correctly rendered onto transparent geometry	MET	
iPhone 7	Animation is driven by light estimation value and lighting and shadow are correctly rendered onto transparent geometry	MET	
iPad	Animation is driven by light estimation value and lighting and shadow are correctly rendered onto transparent geometry	MET	
Samsung Galaxy Tab 3	Since no light estimation is supported values cannot be used	NOT MET	
Dynamic UI			
Criteria		UI adapts to the screen of all test devices on portrait mode	
Device	Result	State	



iPhone X	Adapts to screen in portrait mode keeping aspect ratio and resolution of elements	<b>MET</b>
iPhone 7	Adapts to screen in portrait mode keeping aspect ratio and resolution of elements	<b>MET</b>
iPad	Adapts to screen in portrait mode keeping aspect ratio and resolution of elements	<b>MET</b>
Samsung Galaxy Tab 3	Adapts to screen in portrait mode keeping aspect ratio and resolution of elements	<b>MET</b>

Figure 41: Final Test Results

### 8.3 Evaluation

Device	Evaluation
iPhone X	<b>PASSED</b>
iPhone 7	<b>PASSED</b>
iPad Pro 2017	<b>PASSED</b>
Samsung Galaxy Tab 3	<b>NOT PASSED</b>

Figure 42: Final Device Evaluation

As seen from the table above, it is evident that the Samsung Galaxy Tab 3 did not pass the final test. The device does not yield stable results in the most basic markerless tracking and does not support advanced features such as light estimation. For state of the art use cases it is therefore not usable. The iOS devices all yielded good to very good results throughout all test cases, with minor difficulties in suboptimal environment conditions for the older iPad. When comparing the results with the device specification table in *Figure 1: Test device comparison chart* a relation can clearly be drawn between the device's hardware and its performance.

## Appendix I

### Requirements Traceability Matrix

{FR} - Functional Requirements

{SR} - Non-Functional/System Requirements

**Priority:** **M** - Must, **S** - Should, **C** - Could, **W** - Won't

#### Functional Requirements

The functional requirements define the functions and features the system should have. They are marked on their priority, depending on whether they are required by the thesis assignment and/or stakeholders and also they are marked implemented (MET) or not. Functional requirements do not specify the way of implementation however. Since no actual user testing is performed due to the Covid-19, the functional requirements are not subject to this test report. For the maintenance application the following functional requirements have been established in no particular order:

Req. Code	Req. Description	Status
<b>MUSTS</b>		
<b>[FR01-M]</b>	Implement scanning tutorial to introduce users to AR	<b>MET</b>
<b>[FR02-M]</b>	Placing a virtual object in AR space	<b>MET</b>
<b>[FR03-M]</b>	Positioning, scaling and rotating the virtual object in AR space	<b>MET</b>
<b>[FR04-M]</b>	Display textual information in 2D to convey information to the user	<b>MET</b>

<b>SHOULD</b>		
<b>[FR01-S]</b>	Display animations to support textual information graphically	<b>MET</b>
<b>[FR02-S]</b>	Hotspots indicating the position of interactables in AR space so the user knows the exact location of where to perform the action	<b>MET</b>
<b>[FR03-S]</b>	The app supports occlusion of people and objects	<b>NOT MET</b>
<b>COULD</b>		
<b>[FR01-C]</b>	Animations are driven dynamically by light values	<b>MET</b>

Figure 41: Functional Requirements

#### Non functional/System requirements

Non-Functional or system requirements specify criteria that can be used to judge the operation of a system, rather than specific behaviors. They address e.g. usability, performance, supportability, etc. The features tested in this test report relate to these system requirements:

Req. Code	Req. Description	Status
<b>MUSTS</b>		
<b>[SR01-M]</b>	Must be deployable to iOS and Android mobile and tablet devices supporting AR	<b>MET</b>
<b>[SR02-M]</b>	UI must adapt to all screen sizes and resolutions in Portrait mode	<b>MET</b>

<b>[SR03-M]</b>	Framework must support reliable anchoring of virtual object so they stay in place without moving	<b>MET</b>
<b>[SR04-M]</b>	Model that is used must be optimized to a degree that ensures smooth reliable tracking	<b>MET</b>
<b>[SR05-M]</b>	Model used must be optimized to reduce the size of the app	<b>MET</b>
<b>[SR06-M]</b>	System must recognize planes reliably to ensure the virtual content behaves as expected	<b>MET</b>
<b>[SR07-M]</b>	Feature realistic, dynamic lighting and shadow of the object based on the lighting of the real world to convey the illusion of realism	<b>MET</b>
<b>[SR08-M]</b>	Feature an accurate representation of light and shadow the object causes onto the surface so the objects looks properly grounded in AR space	<b>MET</b>
<b>SHOULD</b>		
<b>[SR01-S]</b>	UI must adapt to all screen sizes and resolutions in Landscape mode	<b>NOT MET</b>

Figure 42: Non-functional requirements

## Appendix II

### Literature Sources

Miller, D.(2019). *Mobile AR Shader*. Retrieved from <https://gist.github.com/DanMillerDev>

SpikeSource, Intel Corporation. (2005). *Business Readiness Rating for Open Source - A Proposed Open Standard to Facilitate Assessment and Adoption of Open Source Software*. BRR 2005 - RFC 1

Socialcompare (2020). *Augmented Reality SDK Comparison*. Retrieved from <http://socialcompare.com/en/comparison/augmented-reality-sdks>

Unity Documentation (2019) *Metallic mode: Metallic Parameter*. Retrieved from <https://docs.unity3d.com/Manual/StandardShaderMaterialParameterMetallic.html>

GsmArena(n.d.). *Device Specifications*. Retrieved from [https://www.gsmarena.com/samsung\\_galaxy\\_tab\\_3\\_7\\_0-5422.php](https://www.gsmarena.com/samsung_galaxy_tab_3_7_0-5422.php)