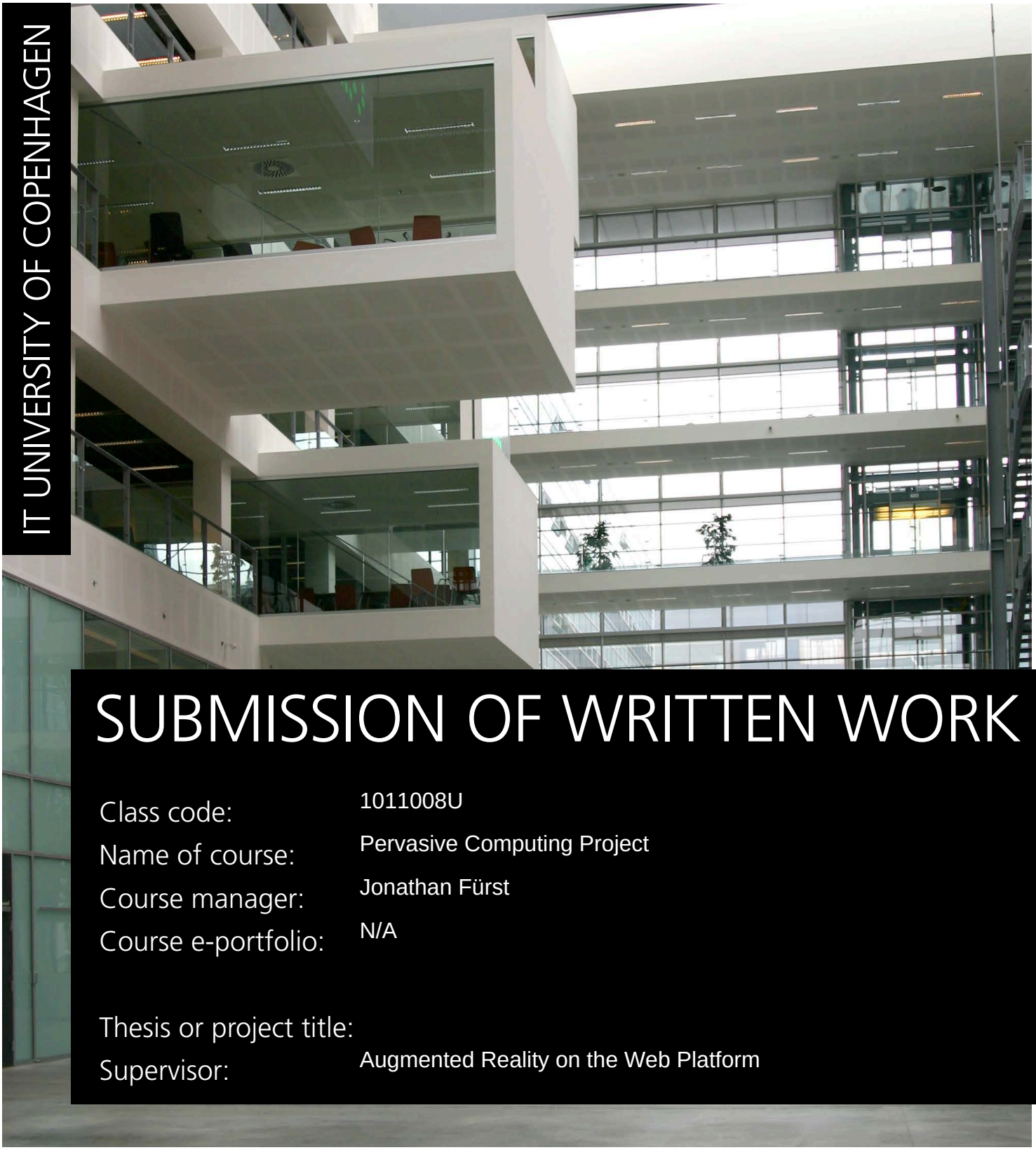
IT UNIVERSITY OF COPENHAGEN



SUBMISSION OF WRITTEN WORK

Class code:	1011008U
Name of course:	Pervasive Computing Project
Course manager:	Jonathan Fürst
Course e-portfolio:	N/A
Thesis or project title:	
Supervisor:	Augmented Reality on the Web Platform

	Full Name:	Birthdate (dd/mm-yyyy):	E-mail:
1.	Theodor Lindekaer	17/12-1991	tcli@itu.dk
2.			@itu.dk
3.			@itu.dk
4.			@itu.dk
5.			@itu.dk
6.			@itu.dk
7.			@itu.dk

Augmented Reality on the Web Platform

A case study for COOP

Theodor C. L. Lindekaer
IT University of Copenhagen
Rued Langgaards Vej 7
Copenhagen, Denmark
tcli@itu.dk

ABSTRACT

This paper provides insights into the usage of the web platform for creating augmented reality applications. The research was initiated on the basis of a case study provided by COOP - a major retail chain in Scandinavia. The case outlined the desire for COOP to explore the domain of augmented reality in order to provide their customers with a better shopping experience.

Part of the research efforts have been spent exploring the current available technologies for augmented reality that are available to the application developer. Once the technological landscape has been sufficiently explored the author continued by developing a sample application to showcase the capabilities of current web technologies. Simultaneously with the development of the web based augmented reality application a feature-equivalent application was developed on the IOS platform.

Benchmarking the performance of the web based AR application against the native IOS alternative the research showed that the native application performed better. It was observed how accessing media (video stream) is time consuming on both platforms. The web based solution does, despite worse performance, have some appealing characteristics such as cross-platform development.

Keywords

Pervasive computing, augmented reality, performance measurement, mobile application development

1. INTRODUCTION

Augmented reality (AR) is technological paradigm centered around technologies that enable an enhancement or augmentation of the real physical world. Often confused with virtual reality (VR) which deals with the complete replacement of the physical world with a virtual equivalent, AR deals exclusively with the blending of virtual objects into the physical world. A commonly used example of an AR application is an information overlay (screen, picture, spreadsheet or similar) projected on physical plane such as a wall or floor. This technique relies on software being able to carry out plane detection algorithms.

2. CASE STUDY

The foundation for this paper is a case study proposed by the major Scandinavian retail chain, COOP. The case study outlined the need for COOP to explore the domain of AR applications in order to provide their customers with a better shopping experience. Currently the conveying of information from store to consumer happens through the textual medium. Labels on products describe characteristics such as nutritional content, expiration date and *product story*. The latter defined as the storytelling used to describe the production process to the consumer with the purpose of convincing them to purchase. The case study material emphasized the opportunity to use AR to provide a richer story telling about the product than currently possible in the textual medium.

3. METHODOLOGY

As a starting point for the research of this paper an analysis of the user environment was conducted. The analysis was based on a typical shopping experience through a COOP supermarket. The shopping experience was not strictly planned, but the author emphasized every step in the decision making process and paid special attention to the characteristics of these. The intention of the work was to pinpoint the current habitual patterns of a shopping customer and map these to the requirements of an AR application in this space. The result of this work is displayed in figure 1. The preliminary findings from this work helped in the process of defining the research design and put an explicit focus on the performance of the AR application.

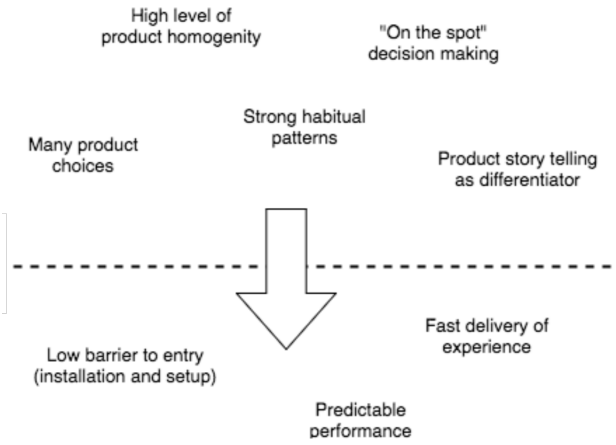


Figure 1: A mapping from current user experience to characteristics of the technical implementation

During the preliminary research efforts were put into visualizing the user interaction flow of a potential AR application in the context of a supermarket. The solution needs to be easily implemented and be applicable to a variety of products. Since a supermarket offer a great variety of heterogeneous goods, the AR application must be flexible and not rely heavily on certain product characteristics. In order to meet this demand the author investigated the possibility of adding a marker directly to the product packaging. The solution is illustrated in figure 2. The marker also contains a QR code in order to create a dynamism - with the same marker being placed on every product, the AR application is not able to differentiate the model displayed. Using a QR code with encoded information the application can upon scanning be aware of which model to display once the marker has been detected. The solution thus relies on two different types of detection - detecting and reading the QR code and detecting the marker.

Figure 2: The full UX flow from scanning of QR code to rendering of 3D model

The preliminary is strictly necessary because of the novelty of AR applications. Currently many AR applications are just simply just proof of concept projects that are trying to push the limits of the technologies. In the case of this research there is an emphasis on the user's ability to successfully derive tangible value from the application. However small in scale and scope, the author was developing the project in an iterative manner as described by Gabbard et al. [3]. Figure 3 shows the approach to developing AR applications.

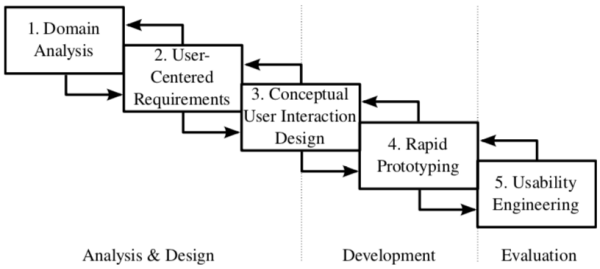


Figure 3: Iterative working process for AR development as proposed by Gabbard

4. MOTIVATION

Before getting into the details of the research conducted for this paper, this section is exclusively devoted to explaining the initial motivation for investigating the web platform as a candidate for AR applications.

4.1 Developer centric perspective

As described by Jobe[5], in his paper on the comparison between web and native apps, the proliferation of mobile devices has created a surge of application developers developing applications for mobile devices. Due to the widespread adoption of the web platform, many developers are working exclusively on the web platform with technologies such as HTML, CSS, JS etc. With an ever increasing fraction of mobile traffic coming from mobile devices, greater emphasis was put on providing a satisfactory experience on a mobile client. One of the first attempts to empower web developers to access native features (e.g. GPS and camera) took shape as the Cordova project[1].

In the fashion of open source, a multitude of projects were introducted as layers on top of Cordova to improve the developer experience. In recent years some of the dominant technological leaders, such as Facebook, have contributed greatly to the empowering of the web developer on the native platforms. One such example is through the React Native[2] (RN) project. RN applies a Javascript abstraction on top of native code - the end result is a fully native app that is orchestrated trough a *bridge* to the Javascript runtime. Google has taken another approach to developing for mobile devices; the company has coined the term *Progressive Web App* (PWA). In order to qualify as a PWA a web application has to implement certain caching methods, structure code according to guiding principles and leverage the capabilities of web workers.

Despite the convergence between web and native apps, Jobe argues that the decision to develop under either paradigm has its distinct advantages and drawbacks. Developing on the web platform gives the ability to "write once, run anywhere", since HTML, CSS and Javascript is uniformly available across all browsers. If any browser vendor is lacking behind the general cycle of feature adoption, many features are available as polyfills. Additionally the web platform is by far the largest ecosystem of libraries, tools and other assets available to developers, which gives great flexibility to the individual application developer. However, Jobe highlights how high performance applications might suffer on the web platform due to the restrictions of the Javascript runtime. Native platforms, such as IOS, is able to run lower level such as C++, which is not possible directly on the web platform (only using an additional transpilation step in the build process).

4.2 User centric perspective

A report published by the industry analysis corporation *ComScore* in 2014 provided evidence for an increase in "app installation fatigue" - only one third of mobile phone users install new apps every month. Pairing this with a decline in app loyalty[3], the incentive for developing and distributing native apps seems to be decreasing. Additionally web based applications are favored due to the notion of "no installation" as the application is publicly accessible directly from any browser. However in the case of application with great bundle size the initial download of the source code can easily become a costly operation ultimately acting as an entry

[1]Cordova project: https://cordova.apache.org/
[2]React Native project: https://facebook.github.io/react-native/
[3]77% of daily active users are lost within the first 3 days after install according to Quettra Mobile Intelligence[1]

barrier.

5. RESEARCH DESIGN

The research design of this paper was greatly influenced by the exploratory character of the case study providing the initial starting point. Commonly native mobile application development platforms such as Android and IOS are used to provide AR experiences to the user. However, due to recent developments on the web platform, AR application development using solely web technologies has become possible. The core question of the author is thus: *Is the web platform a viable alternative to native platforms for the development of AR applications?*. The exploration of the available technologies infused the author with a belief that web technologies are capable of performing equally well as native alternatives - thus the hypothesis *An AR experience based on web technologies can provide a similar user experience compared to a native solution.* In order to quantify the performance evaluation undertaken to compare the different solution, the author decided to focus explicitly on the time spent on certain core operation such as framework start up time, media access time and marker localization time.

6. LITERATURE REVIEW

Studying marker finding is not a new phenomenon in the literature. Several studies have looked extensively at techniques and tools to detect markers. It can thus be considered a well established branch of computer vision research. Of importance for this paper is the work of Hirzer [4], who provides an introduction to the use of ARToolkit and conducts a series of experiments aimed at improving the reliability of the marker finding process. Other papers assess AR from the perspective of HCI and looks into the interaction pattern with AR applications - on of such studies is by Ducher [2]. As described in the methodology section, the suggested approach for the flow of the AR application was to embed a QR code on a marker. This is easily achieveable, but introduces a small degree of redundancy - why not use the QR code itself to serve as a marker? This was achieved by Hong et al.[6], who with an Android setup successfully used a QR code is a marker for a 3D overlay.

7. TECHNOLOGIES

This section will outline the technologies that enable development of AR applications on the web platform. An overview is provided in figure 4. All of the following technologies are part of the library *AR.js*.

7.1 ARToolkit

In order for AR application to anchor virtual elements in the physical world, anchor points are required. Such anchor points can be, e.g. planes or markers. In this paper marker finding is the exclusive focal point. In the space of open source ARToolkit is a widely known library used to detect markers, create a virtual camera inheriting the characteristics of the physical camera and subsequently be able to place an anchored virtual object in the physical world. The complete process is summarized in figure 5.

ARToolkit is written in C++ and thus therefore not directly available for usage on the web platform. Across browsers only Javascript (JS) is able to executed during run time. Using the tool of Emscripten[4] ARToolkit becomes available for use in the browser - the official repository for ARToolkitJS is maintained by the ARToolkit organization.

Figure 4: Web technologies required to proving an AR experience on the web platform

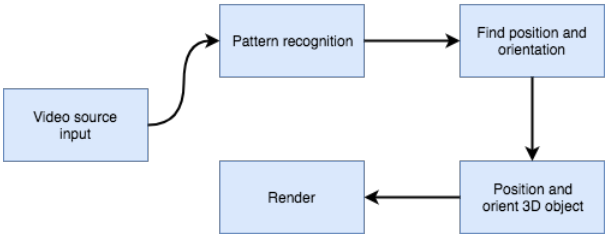


Figure 5: ARToolkit from video input to 3D rendering

ARToolkit is able to detect markers in the video frames that are fed to it. Such markers can take a variety of shapes, sizes and formats. A commonly used marker is the *Hiro Marker Pattern* (see figure 6) - this has also been the marker used throughout this paper. A strong characteristic of the Hiro Marker is the thick black border that helps computer vision algorithms to clearly separate the marker from its surroundings.

The great advantage of ARToolkit in a research context is the fact that the software is available on both IOS, Android and (using an additional compilation step) the web.

Figure 6: A Hiro marker

7.2 WebRTC

[4]A LLVM-to-JavaScript compiler

WebRTC is a suite of technologies that augment that capabilities of browsers by enabling media streaming and direct P2P connections between browsers. In the context of this paper, the media streaming capabilities have been used. Using the *UserMedia* interface the AR application is able to prompt the user for media access; once accepted a full video stream is available. The developer is subsequently able to handle each video frame programmatically. For the experiments of this paper the frames were piped to ARToolkit for processing to detect marker patterns.

7.3 ThreeJS

The last piece of a web based AR application comes from 3D model rendering. For this purpose WebGL is a de-facto technology. Several wrapper libraries have been created in order to provide an easier API for the underlying WebGL. One of such libraries is *Three.js*. With Three.js it is possible to create 3D models of all geometric shapes as well as loading and rendering more complex models from e.g. STL files.

8. EXPERIMENTS

This section will outline the experimental work of this paper. Firstly, some preliminary experiments were done to access the performance impact of the web based AR application - this is in particular interesting because the web based application relies on fetching resources over the HTTP protocol in order to function, while the native version requires only a single download of binaries.

8.1 Performance on the Web

Performance of web based applications is commonly based on load time. This covers the time spent from sending the initial HTTP request until all assets have been downloaded. Load time matters because the user has a limited attention span; the metrics department from the browser vendor Mozilla has set a limit of 2 seconds [7]. An auxiliary experiment was carried out to access the initial load time of the web based AR application. Firstly an analysis of the bundle was carried out. This showed that AR.js has a total bundle size of almost 2MB. Comparing this other widespread libraries (e.g. React or JQuery) AR.js is bigger with a factor of almost 40. The comparison to UI libraries might not be strictly fair, as AR.js contains code for much more complex computations. However, the perception of the end user is greatly influenced by what they have previously experienced. A load time experiment was carried out 10 times with caching disabled - this resulted in an average load time of 8.3 seconds. However considered the widespread (and default) usage of caching techniques on the web platform the performance impact of bundle size can be mitigated effectively on subsequent loads.

8.2 Main experiments

Two separate main experiments have been carried out. One run entirely on web technologies and the other on IOS. In order to carry out a comparative analysis the experiments were to the greatest extent identical to isolate the effect, in terms of performance, of the underlying platform. The experiments are best described as performance testing - more specifically a series of events are timed over the duration from start to detecting marker (the sequence is outline in figure 7).

Figure 7: The sequence of events from JS execution to detecting a marker

Both experiments were run on the same hardware - an iPhone 7 running IOS 11. Prior to the release of IOS 11 WebRTC was not available in the IOS version of Safari. This support is crucial since the AR application needs to access user media. The iPhone was fixated on a tripod with zero degree tilt. The camera was placed opposite to a Hiro marker 25 centimeters away. The Hiro marker was attached to a plain colored surfaced and was not obscured or covered in any way. The experiments were carried out in bright daylight in a well-lit room.

8.3 Web

The experiment running on the web platform was repeated 50 times. The events accounted for in the experiment are ARToolkit initialization, media source initialization, finding of marker and three.js renderer being in a ready state. Results are shown in figure 9.

8.4 IOS

The experiment running on IOS was repeated 50 times. The events accounted for in the experiment are ARToolkit initialization, media source initialization and finding of marker. It was not possible for the author to find an event equivalent to the ready state of the three.js renderer and this event was therefore excluded. Results are shown in figure 10.

8.5 Evaluation of results

Looking at the experimental results one observes how the IOS platform shows better performance in terms of time required to detect the marker. Comparing the average from the two platforms, one sees that IOS is faster by a factor of

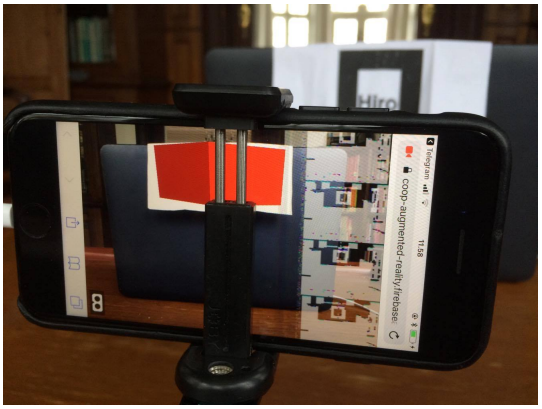


Figure 8: The experimental setup here running in the Safari mobile browser. ARToolkit is running in debug mode and therefore showing the black and white representation of each camera frame

Figure 9: Web experiment

1.5. Interestingly the time spent accessing media is almost equivalent being 800 ms - on this metric alone, the web based application is up to par with the native alternative. This result may also be serve as evidence for a performant implementation of WebRTC; a clear step towards feature compatibility between the web and native platforms. It has to noted that accessing media is a very costly operation on both platforms - it contributes directly to the time spent detecting the marker, since no video frames are sent to ARToolkit for processing.

Of particular interest to the author is the data collected on the initialization of the ARToolkit wrapper. In the IOS implementation the library is initialized after 50 ms on average, while the web based implementation requires four times as much time. Observing a difference of such great magnitude on a relatively simple operation may be a sign of the bottleneck inherent in the Javascript runtime. This was to be expected, since Javascript is higher level language and not as "close to the metal" as C++.

9. IMPLEMENTATION DETAILS

The data gathering from the experiments was enabled by placing *hooks* throughout the software source code. After identifying the events of interest, small snippets of code were

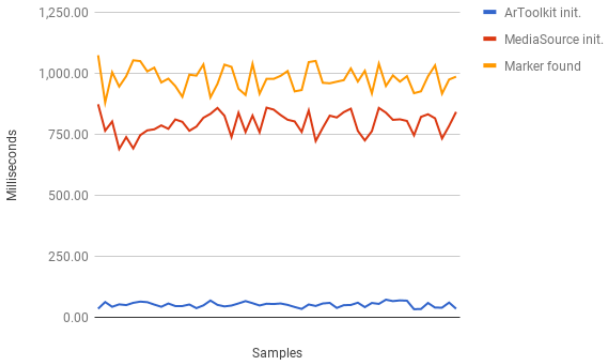


Figure 10: IOS experiment

	WEB			IOS		
	ART.	Media	Marker	ART.	Media	Marker
Mean	224	801	1632	56	758	982
Std.	30	76	15	10	65	58

Table 1: Experimental results all shown in milliseconds. Each event is calculated as the difference between the timestamp of the event and the initialization of the application.

inserted in the source code where certain events were either dispatched or handled. Great emphasis was put on test automation with purpose of achieving fast iteration cycles between each change in the codebase. The author decided to use Google Firebase (GFB) to store the gathered data. The benefit of GFB is the fact that it is document based (and thus do not require any schema definitions) and that it is available directly from the client side. The complete setup did therefore not require any server side programming.

10. SOURCES OF ERROR

There are various factors that need to be assessed when considering the reliability of the experimental results from this paper. Firstly, the experiments were carried out in a moderately controlled setting. Adjustments such as distance to marker and camera tilt could be accurately controlled; however the lighting conditions were not perfectly constant, as the experiment was not carried out in a laboratory. Had the lighting conditions been quantified it would have improved the replicability of the study. Additionally, the chosen lighting conditions might not accurately reflect the lighting conditions present in a COOP supermarket.

In the source code of respectively ARToolkitJS and ARToolkit the interface available to the developer differs. In the Javascript version a higher level event emitter interface is exposed, which makes it easy to carry out experimental work. The degree to which the same events have been captured accurately, and in an equivalent manner, on the native side is not certain - however, the author has to the greatest possible extent identified the most equivalent parts of the source codes to ensure validity.

The maturity of AR.js also has to be considered before making strict conclusions on the basis of its performance. The project is maintained by a small team and is updated frequently with new features - the project is largely a "proof

of concept" and might therefore not accurately reflect the true performance of the web platform, as the current implementation might not yet be optimized for performance.

11. CONCLUSION

This study has shown that the development of AR applications is certainly possible on both the web platform and on IOS. The primary purpose of the experimental work was to investigate whether or not an AR application on the web platform is able to provide a similar user experience compared to a native alternative. Looking strictly at the performance of the selected metrics this has turned out not to be true. The web based application is remarkably slower at finding the marker and thus is not able to provide the same fast experience.

However, the author has throughout the work on this paper become aware of some of the main benefits of working with the web platform. The iterations cycles when working on the web based application are much faster. Additionally, the ability to develop on all platforms simultaneously is a strong argument for choosing the web platform.

Considering the novelty of AR applications the author advices COOP to, despite the inferior performance, to select the web platform for their initial iterations. The low barrier to entry for the developer and the cross platform capabilities makes the web base solution an ideal candidate for testing out early ideas and getting feedback from the users. In the longer run when performance arises as a competitive factor, the native alternatives can no longer be neglected.

12. FUTURE RESEARCH OPPORTUNITIES

Many additional research opportunities exist in the field. In particular, it might prove fruitful to investigate whether or not ARToolkit compiled using WASM[5] instead of Emscripten. WASM is becoming widely accepted as the tool to use a wide variety of languages on the web platform and might therefore be an interesting path to pursue in the context of AR.

Given the fact that only the IOS platform has been investigated in this paper, a similar study carried out on Android would shed some additional light on the capabilities of the native platforms as a whole.

13. RESOURCES

The experimental setup for the project is available at https://github.com/lindekaer/ar-project-code

14. REFERENCES

[1] CloudFour. The business case for progressive web apps, July 2016.

[2] P. Ducher. Interaction with augmented reality. *Universitat Passau*, 2014.

[3] G. et al. Usability engineering: domain analysis activities for augmented-reality systems. *International Society for Optics and Photonics*, 2002.

[4] M. Hirzer. Marker detection for augmented reality applications. *Graz University of Technology*, 2008.

[5] W. Jobe. Native apps vs. mobile web apps. *Stockholm University*, 2013.

[6] R. Kong and H. Jeong. An augmented reality system using qr code as marker in android smartphone. *Engineering and Technology (S-CET)*, 2012.

[7] Mozilla. Firefox page load speed, March 2010.

[5]WASM project: http://webassembly.org/