Towards Augmented Reality Applications in a Mobile Web Context

Antti Karhu, Arto Heikkinen, Timo Koskela
Center for Internet Excellence
University of Oulu
Oulu, Finland
firstname.lastname@cie.fi

Abstract— **Mobile augmented reality (MAR) applications are becoming reality due to the rapid evolution of mobile devices and their capacities. In this paper, we introduce MARW, a development framework for MAR applications on a web browser. Use of the web browser technology enables running the same MAR applications on all mobile platform and operating system configurations. The performance of the implemented MARW prototype was evaluated with several experiments using multiple mobile device and web browser combinations. Based on the experimentation results, a maximum number of 12 frames per second were achieved with a high-end tablet device in vision-based tracking using markers. It was also discovered that the performance differences between web browsers are significant in terms of MAR application performance.**

Keywords—AR; 3D; web browser; web services; html5, javascript; performance;

I. INTRODUCTION

Augmented Reality (AR) is an interactive technology that allows enhancing the view of the physical real world with computer-generated digital information [1][2]. Due to the advancement of graphics processing capacity and the embedded location and orientation sensors, mobile devices (smart phones and tablets) have recently become a very attractive platform for AR applications [3]. Today, many commercial mobile AR (MAR) applications and application development frameworks already exist (e.g. Wikitude, Layar and Nokia City Lens), but currently, they all require installation of a standalone MAR application that needs to be tailored for all supported platform and operating system configurations. This does not only introduce an additional cost for service providers, but may also lead to a prolonged adoption of novel MAR applications [4]. Rapidly developing web technologies provide tools for implementing MAR applications that run on cross-platform supported web browsers [5]. However, web technologies have not yet been integrated together in order to provide a complete application development framework for MAR applications. Furthermore, implementing MAR applications on the web browser introduces some performance challenges that have not yet been sufficiently addressed.

This paper presents a framework for implementing MAR applications on a cross-platform supported web browser (hence called MARW). MARW relies on the recent and yet drafted web standards including HTML5[1], WebRTC[2], XML3D [6] and Xflow [5]. To examine the feasibility of the implemented MARW prototype, performance of sensor and vision-based registration and tracking functionality was evaluated with several mobile device and web browser combinations. Based on the evaluation results, the performance challenges for implementing MAR applications on a web browser were identified and discussed. This information can be used for guiding the research and development efforts towards improving the efficiency of web technologies from the standpoint of novel MAR applications.

The rest of the paper is organized as follows: Section 2 presents the related work on MAR applications in the web context, Section 3 introduces MARW and its components in detail, Section 4 describes the experimental setup for evaluating the performance of the MARW prototype, Section 5 presents the results of the experimentation, Section 6 discusses the performance challenges of implementing MAR applications on a web browser and concludes the paper.

II. RELATED WORK

In the earlier research, hundreds of different MAR applications have been presented ranging from library guides [7] to various shopping assistants [8]. However, all these MAR applications have been developed as standalone applications for a specific platform and operating system configuration. Only recently, some application development frameworks for MAR applications have been proposed that either (1) extensively rely on web technologies, but yet implement a platform-specific AR browser (e.g. [3][9]) or (2) operate entirely on a web browser [5][10]. It is important to emphasize that even though the AR browsers use web technologies, they do not inherently integrate the 3D content into HTML

[1] HTML5: A vocabulary and associated APIs for HTML and XHTML, W3C draft 12 December 2013: http://www.w3.org/html/wg/drafts/html/CR/

[2] WebRTC 1.0: Real-time Communication Between Browsers, W3C draft 10 September 2013: http://www.w3.org/TR/2013/WD-webrtc-20130910/

978-1-4799-5073-7/14 $31.00 © 2014 IEEE
DOI 10.1109/NGMAST.2014.36

document object model (DOM) as is done with MARW using XML3D. As a result, the 3D content commonly used in MAR applications cannot be directly accessed and modified using common JavaScript libraries such as jQuery.

When compared to MARW, other web browser based MAR frameworks [5][10] provide only vision-based registration and tracking, whereas MARW also provides an option for sensor-based registration and tracking. In addition, MARW provides support for fetching data from 3rd party Web Services, which is lacking in [5] and [10]. Moreover, MARW does not enforce use of any specific data structures, which improves its applicability. It is crucial for MAR frameworks to enable use of existing web content as fluently as possible to facilitate MAR application development. The shorter the time needed for the MAR application development, the shorter the time-to-market and the lower the monetary expenses.

III. Framework for Augmented Reality Application Development in Mobile Web Context

The objective of MARW is to provide a high-level application programmer's interface (API) stack, which is used for implementing various kinds of MAR applications. The framework utilizes several existing web technologies including HTML5, WebRTC and XML3D. In Fig. 1, the API stack of MARW is presented. Next, the operation of each MARW component is described in detail.

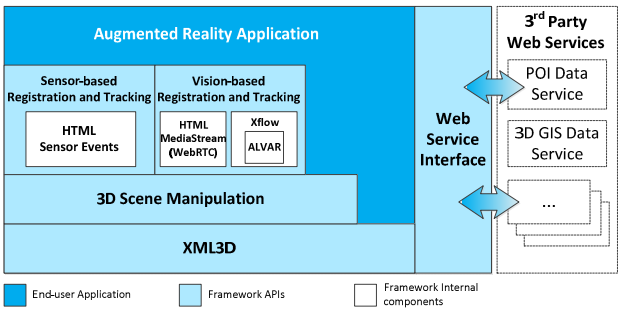

Figure. 1. The API stack of MARW.

A. Sensor-based Registration and Tracking

The Sensor-based Registration and Tracking component is used to sense relevant information from the real-world surroundings. It provides a single uniform API allowing access to the following W3C specifications: Geolocation, DeviceOrientation, DeviceLight, and DeviceProximity. The Sensor-based Registration and Tracking component allows programmers to easily bind call-back functions into the different HTML5 sensor events for processing the sensor event data. Geolocation API is used to determine the location of the mobile device. DeviceOrientation API is used to sense how much the mobile device is leaning side-to-side, front-to-back and the direction it is facing. DeviceProximity API is used to sense the distance between the mobile device and the physical object nearby. DeviceLight API is used to sense the light intensity level in the surrounding environment.

B. Vision-based Registration and Tracking

The Vision-based Registration and Tracking component is used for registering visual features and for detecting them in a video stream obtained from the camera of the mobile device as shown in Fig. 2. This is realized by combining the HTML MediaStream interface, ALVAR[3] and Xflow.

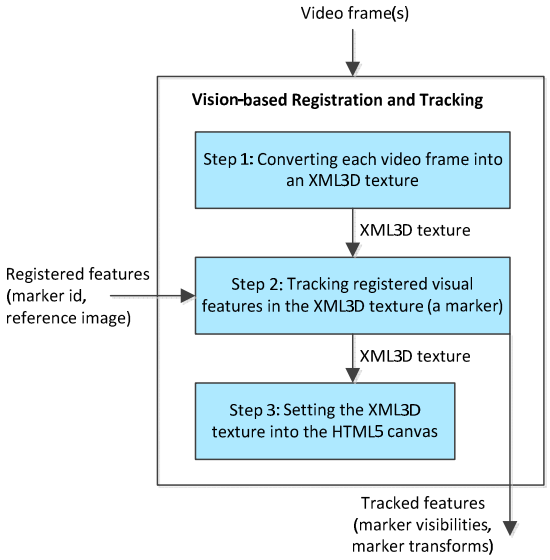

Figure. 2. The steps in the vision-based registration and tracking.

The HTML MediaStream interface provides access to local devices that can generate multimedia stream data. In this case, the multimedia stream is a video stream and the local device is the camera of the mobile device. ALVAR is a library for creating AR applications. MARW uses the JavaScript version of ALVAR which runs on web browsers. At the moment, the functionality of the JavaScript version is limited to marker-based tracking, which detects both fiducial[4] and image[5] markers as well as their transformation information. Fiducial markers are registered using their numerical identifiers and the image markers using reference images.

Xflow is a general purpose data processing tool designed to perform high-performance data processing tasks in a web context [5]. Xflow allows defining data processing flows using DOM elements. In MARW, the ALVAR functionality is encapsulated into a Xflow element. Hence, the ALVAR can be easily placed into the DOM structure.

C. 3D Scene Manipulation

The 3D Scene Manipulation component is an interface between the Sensor and Vision-based Registration and Tracking components and the XML3D component. For example, a MAR application can use the orientation of a

[3] http://virtual.vtt.fi/virtual/proj2/multimedia/alvar/index.html

[4] A fiducial marker is defined as a square box which is divided into a grid. The outside cells of the grid are always black and the inside cells can be either white or black. The inside cells are used for encoding the data in a binary form.

[5] An image marker is very similar to a fiducial marker, however, the resolution of its grid is larger and the inside cells are selected to form a shape or a logo.

mobile device to manipulate the virtual camera, or use the detected markers and the GPS coordinates for placing 3D objects in the 3D scene. However, the data obtained from Sensor and Vision-based Registration and Tracking components is not in a form that the XML3D component can use directly. The GPS location is provided in the form of latitude and longitude pair on the WGS84 coordinates system. The device orientation is provided in the form of a set of intrinsic Tait-Bryan angles of type Z-X'-Y''. The marker transform is provided in the form of 4x4 transform matrix. In order to use this information it must be first converted into right-handed, three-dimensional Cartesian coordinate system used by the XML3D component. The 3D Scene Manipulation component provides the required methods for making the format conversions.

D. XML3D

The XML3D component contains the representation of the 3D content rendered on top of the video stream obtained from the camera of the mobile device. XML3D is a declarative representation of 3D content for HTML, fully integrated into DOM. XML3D encapsulates the 3D content into different types of nodes (mesh, light, etc.), which comprise a 3D scene as a graph like structure inside the DOM. Each node within this 3D scene graph is also a node in a web page's DOM tree representation, and can be accessed and modified like a common DOM element. Because of the DOM integration, each scene graph node receives HTML events (on-click, on-mouse-over etc.) similar to regular HTML elements. XML3D is a proposal for extension to HTML5 [6]. However, XML3D is not natively supported in web browsers yet, hence MARW emulates it using xml3d.js, which is a Polyfill implementation of XML3D [11].

E. Web Service Interface

Use of Web Services is a way of communication between a data provider (a 3rd party Web Service) and a data requester (a MAR application). The Web Service Interface component manages the communication between a MAR application and the 3rd Party Web Services using standard web technologies. The communication takes place either using RESTful API over HTTP, or through a WebSocket, for instance, when the requested data is acquired frequently. The Web Service Interface component has utility functions for initiating REST queries and for listening to Web sockets. It also provides automatic parsing of JSON data.

F. The 3rd Party Web Services

The 3rd Party Web Services provide the content, such as text, images, audio, videos and 3D objects, displayed in a MAR application. The content usually contains spatial information (i.e. content metadata) such as the GPS coordinates, which enables the content to be retrieved and visualized correctly in a MAR application. For example, a 3rd Party Web Service may be a point-of-interest (POI) or a 3D GIS Data Service, which provide functionality for searching content from a specific location based on GPS coordinates.

G. Augmented Reality Application

The API stack architecture is modular and each of its components is independent. Therefore, one can use only the components needed in a specific MAR application. The modularity is demonstrated briefly by explaining which components are needed in three MAR applications developed using MARW.

In the first MAR application, a virtual model of a house is placed on top of the surrounding environment as illustrated in Fig. 3. The user can examine the interior and exterior space of the virtual house, and also view the surroundings of the house through its windows by moving the mobile device. Components used in this MAR application are Sensor-based Registration and Tracking, 3D Scene Manipulation and XML3D.

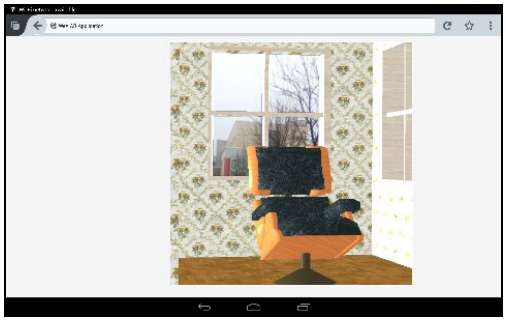

Figure. 3. A virtual model of a house on top of the surrounding environment.

In the second MAR application, POIs such as nearby restaurants and cafes are visualized as 3D pointer objects on top of the surrounding environment. When the mobile device is directed towards one of the 3D pointer objects, more information is shown to the user. Components used in this MAR application are Sensor-based Registration and Tracking, 3D Scene Manipulation, XML3D, Web Service Interface and 3rd Party Web Services.

In the third MAR application, a model of an airplane is rendered on top of an identified marker as shown in Fig. 4. When the user touches the airplane it flies off and the user can follow the airplane by turning the mobile device. Components used in this MAR application are Sensor-based Registration and Tracking, Vision-based Registration and Tracking, 3D Scene Manipulation and XML3D.

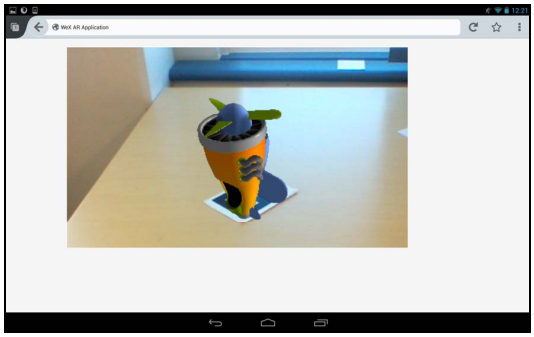

Figure. 4. A virtual aeroplane ready to fly off after a user touches it.

IV. Experimental Setup

To evaluate the performance of MARW and to identify the performance bottlenecks of running MAR applications on a web browser, four different experiments were conducted; three with vision-based registration and tracking and one with sensor-based registration and tracking.

A. Devices and Tools

For vision-based registration and tracking experiments, the devices used were a laptop computer (HP EliteBook 2760p: Intel i5-2540M, 4GB RAM, Intel HD Graphics 3000, Win8), a tablet device (HP Slatebook x2: Tegra 4, 2GB RAM, Android 4.2.2), and a smartphone (Motorola Droid Razr I: Intel Atom Z2460, 1GB RAM, Android 4.1.2.). The web browsers used were Mozilla Firefox Nightly (32.0a1) and Google Chrome Beta (35.01). For sensor-based registration and tracking experiment, the same tablet device was used with the both web browsers.

B. Methodology

For evaluating the performance of Vision-based Registration and Tracking component, the following three questions were examined:

(1) How is the processing time distributed between different steps in the marker-based tracking, when using different mobile device and web browser combinations?

(2) How much does the reference image size affect the processing time in the marker-based tracking, when using different mobile device and web browser combinations?

(3) Are there significant differences in performance between alternative marker-based tracking technologies developed for web browsers?

For evaluating the performance of Sensor-based Registration and Tracking component, the following question was also examined:

(4) How significant is the variation of sensor values when obtained using different web browsers?

For (1) and (2), the processing time in the marker-based tracking was measured in all three steps that were shown in Fig. 2. In the measurements, three different sets of markers were used. The first set contained five 3x3 fiducial markers, the second set five 5x5 fiducial markers, and the third set five image markers. Each image marker was also registered with different levels of accuracy, having 16x16, 32x32, 64x64 or 128x128 pixels. The time elapsed in each step was measured by recording the current time before and after the call to the respective step in the JavaScript code. To ensure the same test conditions for all mobile device and web browser combinations, the tracking process was carried out using 15s of recorded video (resolution: 720x480, frames per second: 30) presenting the marker sets. For each marker set, an average frame processing time was calculated using the recorded video.

For (3), the processing time in the marker-based tracking was also measured for JSARToolkit[6] (a popular JavaScript library for vision-based registration and tracking) in a similar way as for ALVAR in (1) and (2). However, the measurements were carried out solely with 5x5 fiducial markers that are the only markers supported by JSARToolkit.

For (4), the variation of sensor values in the sensor-based tracking was examined by measuring mobile device acceleration. First, the tablet device was kept stationary, and second, it was carried while walking a straight line until 800 samples were collected. While walking the tablet device was held in an approximately 45 degrees angle. The acceleration data were obtained using the HTML5 device motion events.

V. Results

Next, the results of the four experiments conducted with MARW are presented. Based on the results, the defined research questions are elaborated.

A. Vision-based Registration and Tracking Experiments

In Fig. 5 and Fig. 6, the processing times with fiducial and image markers are presented with different mobile device and web browser combinations. As the performance difference between 3x3 and 5x5 fiducial markers as well as between 32x32 and 64x64 image markers was rather insignificant, 3x3 fiducial markers and 32x32 image markers were left out from Fig. 5 and Fig. 6. In addition, 128x128 image markers were left out from Fig. 6, since they were clearly too heavy to detect with mobile devices.

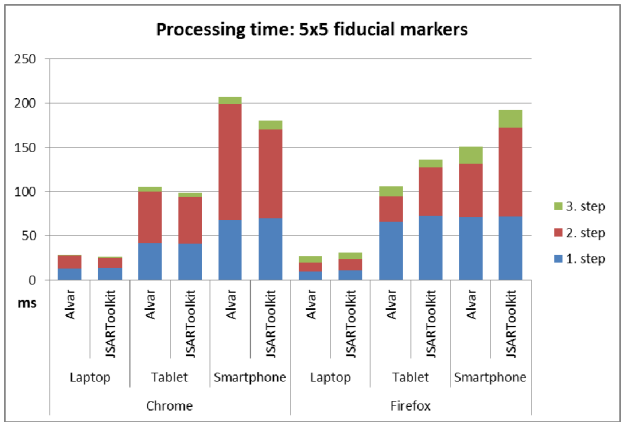

Figure. 5. Processing times with fiducial markers (lower values are better).

For (1), Fig. 5 and Fig. 6 indicate that with Chrome, the performance bottleneck is the second step (i.e. marker identification). With the laptop, the difference in the processing times between the first (i.e. video frame conversion) and the second step is minimal, but it increases significantly with the tablet and the smartphone. When comparing the processing times in the second step, Firefox performed generally better than Chrome. The explanation for this is that the JavaScript version of ALVAR was converted from C++ to asm.js using

[6] http://fhtr.org/JSARToolKit/

Emscripten. asm.js is a strict subset of the JavaScript language, intended to have performance characteristics closer to that of native code than standard JavaScript. Although both web browsers support asm.js, its performance is better with Firefox[7].

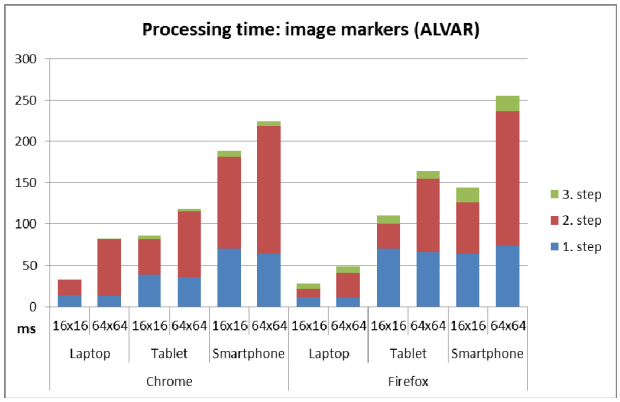

Figure. 6. Processing times with image markers (lower values are better).

Fig. 5 and Fig. 6 also indicate that with Firefox, the performance bottleneck shifts from the second step to the first step when using less complex markers (e.g. fiducial 5x5, image 16x16) with the tablet and the smartphone. This phenomenon is probably related to the mechanisms how the video frame is converted into the XML3D texture using the canvas 2D context, which should be hardware-accelerated on both web browsers. However, Chrome and Firefox may use the OpenGL API differently when implementing hardware-acceleration, for instance, Chrome could be using some extensions to enhance its performance. Another reason could be that there is a difference in how the web browsers handle the video codec. For instance, is the decoding process done entirely in the GPU or is the CPU involved in some stages such as in the color conversion.

TABLE I. THE NUMBER OF FRAMES PER SECOND IN EACH EXPERIMENT (HIGHER VALUES ARE BETTER).

	Laptop		Tablet		Smartphone	
	Chrome	Firefox	Chrome	Firefox	Chrome	Firefox
Fiducial 3x3	38	37	11	10	5	8
Fiducial 5x5	34	37	10	9	5	7
Fiducial 5x5 (JSARToolkit)	38	32	10	7	6	5
Image 16x16	30	36	12	9	5	7
Image 32x32	26	30	10	8	5	6
Image 64x64	12	21	8	6	4	4
Image 128x128	8	14	5	4	2	2

For (2), Fig. 6 indicates that the size of the reference image used in the registration correlates with the processing time required for image marker detection as expected. As seen in Table I, the number of frames per second (FPS) stays interactive (>30) up to 32x32 image markers on Chrome and up to 64x64 image markers on Firefox with the laptop. The values for FPS were acquired by calculating the inverse of the average processing time needed for all three steps for a single frame. In this case, the time used for 3D rendering was considered negligible. As an example, the rendering of a Utah teapot (~10k vertices) on top of a marker took less than 5ms with all devices used in the experiments. With the tablet and the smartphone, the drop in FPS is not as drastic as with the laptop. This is due to the fact that the processing time needed for the first step is dominating with the tablet and the smartphone. However, with these two mobile devices FPS is already low with the smallest (16x16) image markers.

For (3), Fig. 5 indicates that JSARToolkit performs slightly better on Chrome and ALVAR, in turn, on Firefox. With the laptop the difference is marginal, but it increases with the tablet and the smartphone. The difference occurs in the second step, and therefore, it is most likely related to the performance of the web browsers' JavaScript engine.

B. Sensor-based Registration and Tracking Experiments

In Fig. 7 and Fig. 8, the acceleration data samples with Chrome and Firefox are presented.

For (4), when the tablet was carried while walking, the average acceleration was the same (0.11 m/s^2) with both web browsers. However, the standard deviation with Chrome (1.30m/s^2) was clearly higher than with Firefox (0.93m/s^2). As indicated by Fig. 7 and Fig. 8, the data samples provided by Firefox are heavily filtered at some stage, probably using a moving average filter. The filtering is either conducted by Firefox itself or Firefox uses different sensor APIs provided by the Android operating system.

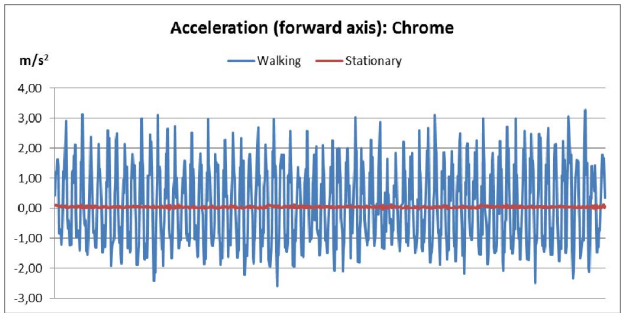

Figure. 7. 800 acceleration data samples collected with Chrome.

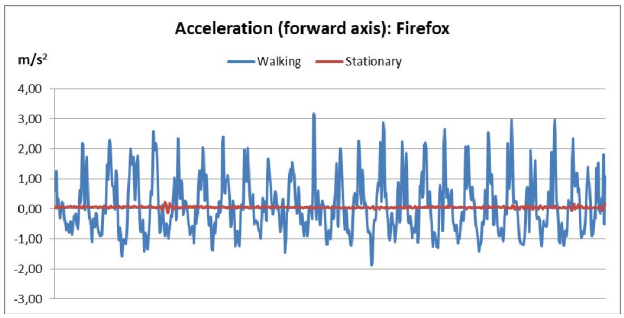

Figure. 8. 800 acceleration data samples collected with Firefox.

[7] Asm.js benchmarks: http://kripken.github.io/mloc_emscripten_talk/sloop.html#/7, http://people.mozilla.org/~lwagner/gdc-pres/gdc-2014.html#/34

Regardless of the fact whether the tablet was carried while walking or kept stationary, it appeared that Firefox provided acceleration data samples at a higher frequency than Chrome. Based on our measurements, it seems that the HTML5 motion event interval is set to 50ms with Chrome (Beta 35.01) and to 20ms with Firefox (Nightly 32.0a1), although Firefox officially reports the HTML5 motion event interval as 100ms. To investigate this phenomenon further, orientation data samples were also collected when the tablet was kept stationary. In this case, Firefox provided data samples at a significantly higher frequency. The greater difference probably results from the fact that Chrome does not publish a new orientation data sample if the change in the orientation data is considered insignificant.

VI. Discussion and Conclusions

Due to the huge number of different mobile devices and their operating system combinations, the cross-platform supported web browser technology provides an ideal development platform for MAR applications. However, it should be noted that web browsers are not equal in terms of their performance which becomes easily evident with mobile devices. As a consequence, this makes the optimization of development frameworks for MAR applications challenging on web browsers. In addition, web browser technology also introduces performance issues such as the single threadness of JavaScript execution that are not present with native mobile applications.

Based on our measurements and the prototype MAR applications, the performance of a mobile device is a not critical factor when sensor-based registration and tracking is used. In this case, the challenge lies in the accuracy of the sensor readings. While walking and carrying a mobile device, the standard deviation in the acceleration data, for instance, was significant especially with Chrome that did not enforce any pre-filtering on the acceleration data. This must be taken into account in the MAR application development.

The vision-based registration and tracking, in turn, requires a lot of processing power from the mobile device as shown in our measurements. With the current high-end tablet devices, the FPS of 10 is tolerable, however, not very interactive if the vision-based tracking is used continuously. It should also be noted that the marker-based registration and tracking, the only robust option available for web browsers today, is rather limited and also natural feature based (i.e. markerless) tracking needs to be developed further. In [10], natural feature based tracking was divided into detection and tracking phases. Based on the measurements conducted in [10], the natural feature detection is currently overly burdensome for mobile devices taking more than 500ms per frame. Once the natural features are detected, their continuous tracking takes 100ms per frame on average. However, if the tracked natural features are lost, the burdensome detection phase needs to be revisited.

Already today, web browsers are being developed towards the goal, where the execution of JavaScript becomes much faster. Good examples of this trend are the attempts to parallelize the JavaScript execution ([12] and SIMD.js). A considerable performance boost for MAR applications can also be achieved through WebCL if this JavaScript binding to OpenCL becomes supported by a growing number of web browsers. Thanks to Emscripten, this would enable use of well-optimized OpenCL-based C++ computer vision libraries such as OpenCV in web browsers without the need for manually porting them to JavaScript.

Finally, the work presented in this paper will be used as a part of the European Commission FI-WARE open cloud-based infrastructure for building Future Internet applications.

Acknowledgment

This research has been supported by the EU FI-PPP FI-WARE project and the CADIST3D project funded by the Academy of Finland.

References

[1] F. Zhou, H.B.-L. Duh, and M. Billinghurst, “Trends in Augmented Reality Tracking, Interaction and Display: A Review of Ten Years of ISMAR,” In Proc. IEEE International Symposium on Mixed and Augmented Reality, IEEE Press, 2008, pp. 193-202, doi: 10.1109/ISMAR.2008.4637362.

[2] T. Olsson, T. Kärkkäinen, E. Lagerstam, and L. Ventä-Olkkonen, “User evaluation of mobile augmented reality scenarios,” Journal of Ambient Intelligence and Smart Environments, vol. 4, pp. 29-47, 2012.

[3] B. MacIntyre, A. Hill, H. Rouzati, M. Gandy, and B. Davidson, “The Argon AR Web Browser and Standards-based AR Application Environment,” In Proc. IEEE International Symposium on Mixed and Augmented Reality, IEEE Press, 2011, pp. 65-74, doi: 10.1109/ISMAR.2011.6092371.

[4] T. Dahl, T. Koskela, S. Hickey, and J.M. Vatjus-Anttila, “A virtual world web client utilizing an entity-component model,” In Proc. 7th Intl. Conference on Next Generation Mobile Apps, Services and Technologies, IEEE Press, 2013, pp. 7-12, doi: 10.1109/NGMAST.2013.11.

[5] F. Klein, D. Rubinstein, K. Sons, F. Einabadi, S. Hermut, and P. Slussallek, “Declarative AR and image processing on the web with Xflow,” In Proc. 18th International Conference on 3D Web Technology, ACM Press, 2013, pp. 157-165, doi: 10.1145/2466533.2466544.

[6] K. Sons, F. Klein, D. Rubinstein, S. Byelozyorov, and P. Sludallek, “XML3D: Interactive 3D Graphics for the Web,” In Proc. 15th International Conference on Web 3D Technology, ACM Press, 2010, pp. 175-184, doi:10.1145/1836049.1836076.

[7] J. Hahn, “Mobile augmented reality applications for library services,” New Library World, vol. 113, iss. 9/10, pp. 429-438, 2012.

[8] J. Carmigniani, B. Furht, M. Anisetti, P. Ceravolo, E. Damiani, and M. Ivkovic,”Augmented reality technologies, systems and applications,” Journal of Multimedia Tools and Applications, vol. 51, iss. 1, pp. 341-377, 2011.

[9] Y. You, and V.-V. Mattila, “Visualizing Web Mash-ups for In-Situ Vision-Based Mobile AR Applications,” In Proc. ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication, ACM Press, 2013, pp. 271-274, doi: 10.1145/2494091.2494177.

[10] C. Oberhofer, J. Grubert, and G. Reitmayr, “Natural feature tracking in JavaScript,” In Proc. IEEE Virtual Reality Short Papers and Posters, IEEE Press, 2012, pp. 113-114.

[11] J. Jankowski, S. Ressler, K. Sons, Y. Jung, and J. Behr, “Declarative integration of interactive 3D graphics into the world-wide web: principles, current approaches, and research agenda,” In Proc. 18th International Conference on 3D Web Technology, ACM Press, 2013, pp. 39-45.

[12] S. Herhut, R.L. Hudson, T. Shpeisman, and J. Sreeram, “Parallel programming for the web,” In Proc. 4th USENIX Conference on Hot Topics in Parallelism, 2012, pp. 1-6.