Demystifying Mobile Extended Reality in Web Browsers: How Far Can We Go?

Weichen Bi
School of Computer Science, Peking University, Beijing, China
biweichen@pku.edu.cn

Yun Ma*
Institute for Artificial Intelligence, Peking University, Beijing, China
mayun@pku.edu.cn

Deyu Tian
School of Computer Science, Peking University, Beijing, China
tiandeyu@pku.edu.cn

Qi Yang
School of Computer Science and Engineering, University of New South Wales, Australia
qiyang@student.unsw.edu.au

Mingtao Zhang
School of Computer Science, Peking University, Beijing, China
1900013027@pku.edu.cn

Xiang Jing
School of Software & Microelectronics, Peking University, Beijing, China
jingxiang@pku.edu.cn

ABSTRACT

Mobile extended reality (XR) has developed rapidly in recent years. Compared with the app-based XR, XR in web browsers has the advantages of being lightweight and cross-platform, providing users with a pervasive experience. Therefore, many frameworks are emerging to support the development of XR in web browsers. However, little has been known about how well these frameworks perform and how complex XR apps modern web browsers can support on mobile devices. To fill the knowledge gap, in this paper, we conduct an empirical study of mobile XR in web browsers. We select seven most popular web-based XR frameworks and investigate their runtime performance, including 3D rendering, camera capturing, and real-world understanding. We find that current frameworks have the potential to further enhance their performance by increasing GPU utilization or improving computing parallelism. Besides, for 3D scenes with good rendering performance, developers can feel free to add camera capturing with little influence on performance to support augmented reality (AR) and mixed reality (MR) applications. Based on our findings, we draw several practical implications to provide better XR support in web browsers.

CCS CONCEPTS

• **General and reference** → **Measurement**; • **Software and its engineering** → **Software performance**.

KEYWORDS

Extended Reality; Web browser; Measurement

ACM Reference Format:
Weichen Bi, Yun Ma, Deyu Tian, Qi Yang, Mingtao Zhang, and Xiang Jing. 2023. Demystifying Mobile Extended Reality in Web Browsers: How Far Can We Go?. In *Proceedings of the ACM Web Conference 2023 (WWW '23), April 30–May 04, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3543507.3583329

*Corresponding author

WWW '23, April 30–May 04, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04…$15.00
https://doi.org/10.1145/3543507.3583329

1 INTRODUCTION

With the rapid development of network and hardware capacity, mobile extended reality (XR) is becoming increasingly popular in recent years [35, 45, 50, 56]. The market of mobile XR has reached $35.14 billion in 2022 and will increase to $345.9 billion in 2030 [8]. Numerous XR applications are emerging in various areas, such as games [29, 30, 52], education [36, 43], industrial engineering [28, 48], and medical treatment [25, 27, 62]. Hardware (Oculus [15], Pico [16]), software (ARCORE [3], ARKit [5]), and standards [24] have been developing rapidly and iteratively in recent years.

Currently, the typical way of accessing mobile XR is to download XR applications from app stores such as Google Play [12] and Apple Appstore [2]. However, app-based XR needs to be installed in advance, making it inconvenient for users to taste and also creating obstacles to the promotion of the application. Besides, the applications are developed by various languages and SDKs for different platforms, i.e., ARCore [3] for Android and ARKit [5] for iOS. As a result, developers need to make significant efforts for cross-platform applications.

Web-based XR, however, has the advantages of being lightweight and cross-platform compared to native applications, leading to the potential of providing a pervasive XR experience to users and making the development process for multiple platforms much easier [54]. Web-based XR has developed rapidly in recent years, and many frameworks have been proposed to support development of XR applications on the web browser. Javascript libraries such as three.js [21] and A-Frame [1] provide the basic functionality of supporting 3D exhibitions. They also use a standard called WebXR Device API [24] for camera capturing and real-world understanding. Other efforts, such as AR.js [4] and MindAR.js [13], are pure Javascript libraries designed for augmented reality.

Despite the strengths and fast growth of web-based XR, we know little about how well it actually performs on mobile devices, hindering the development of the XR experience and missing the direction on the enhancement of browsers' support on XR.

To fill the knowledge gap, in this paper, we conduct an empirical study of mobile XR in web browsers by evaluating its runtime performance. We select seven most popular open-source frameworks for web-based mobile XR and examine their performance during and after loading XR applications. The process of XR is divided into 3D rendering, camera capture, and real-world understanding. Each procedure is measured separately. We also excavate the possible reasons for performance decrease in a complex situation.

Based on our experiments, we make the following interesting and useful observations:

• **There exists no framework that dominates others in all aspects.** Frameworks that perform best in first loading and continuous rendering after loading are different.

• **Model complexity thresholds exist for performance.** As 3D model complexity increases, it has little negative impact on performance until model complexity reaches a certain threshold. But after the 3D model complexity overwhelms the threshold and continues to grow, the performance drops continuously.

• **Enhancing GPU utilization and computing parallelism is needed to improve frameworks' performance further.** GPU utilization can be further improved because the 3D rendering performance begins to drop when hardware resources are not fully utilized. In addition, there exist opportunities for parallel computation in 3D rendering but current frameworks do not fully utilize such parallelism.

• **Adding camera capturing has limited effect on well-performed 3D scenes.** When we add camera capturing for well-performed 3D rendering scenes to support augmented reality and mixed reality, there only exists limited influence on performance.

• **Accelerating web 3D rendering is urgent to save time for better real-world understanding.** Calculating the time left for real-world understanding, we find that 3D rendering needs to be further enhanced if we want a more realistic experience.

Based on the observations, we draw some implications for different stakeholders in the web-based mobile XR ecosystem, including XR app developers, XR framework vendors, and browser vendors.

The rest of this paper is organized as follows. Section 2 describes the methodology used in this empirical study. Section 3 presents the results of performance analysis. Section 4 discusses implications for different stakeholders. Section 5 surveys related work, and Section 6 concludes this paper.

2 METHODOLOGY

This section describes how we conduct the measurement study, including XR framework selection, 3D model generation, experiment setup, evaluation tool, and performance analysis sequence.

2.1 Selected XR Frameworks

In order to select the most popular frameworks, we searched on GitHub with the keywords related to web and XR, including web, Javascript, VR/AR/XR, augmented reality, virtual reality, and extended reality. Then we selected the frameworks whose stars exceed 1k and filtered out two non-active frameworks, i.e., tracking.js [23] and three.ar.js [20]. Seven frameworks finally remained:

• **Three.js** [21] is a Javascript 3D library that aims to create an easy-to-use, lightweight, cross-browser, general-purpose 3D library.

• **React-three-fiber** [18] is a React renderer for three.js.

• **Babylon.js** [6] tends to create a powerful, beautiful, simple, and open game and rendering engine.

• **A-Frame** [1] is a web framework for building virtual reality experiences, which makes 3D worlds with HTML and entity-component.

• **Playcanvas** [17] aims to provide a fast and lightweight Javascript game engine built on WebGL and glTF.

• **AR.js** [4] is a Javascript framework that supports augmented reality functions, including image tracking, location-based AR, and marker tracking.

• **MindAR.js** [13] is a framework for web-based augmented reality, which provides image tracking and face tracking functions.

In the following, we sometimes use abbreviations of the frameworks for better presentation. THR, RTF, BBL, AF, PC, and MA represent Three.js, React-Three-Fiber, Babylon.js, A-Frame, Playcanvas, and MindAR.js, respectively.

We summarize some basic information and functionality of the frameworks in Table 1. *3D Rendering Engine* refers to the way a framework implements 3D rendering. Some frameworks integrate another to address 3D rendering. *Supported 3D Formats* shows the support of different frameworks for standard 3D formats, including glTF (Graphics Language Transmission Format) [11], OBJ/MTL (Wavefront Material Template Library) [14], and FBX (Filmbox) [9]. *AR Implementation* declares the ways the framework is used to support augmented reality. While some frameworks address the issue by Javascript library, others support augmented reality by calling WebXR Device API provided by web browsers.

2.2 3D Model Generation

We generate 3D models of different formats and complexity. To test the upper bound of current frameworks, we increase the model complexity until it damages the performance significantly.

3D Formats. We mainly discuss the glTF format that is supported by all frameworks. We also compare the performance of OBJ/MTL and FBX formats with glTF.

3D Model Complexity. 3D models consist of meshes and materials. Meshes contain points, lines, and triangles. We verify that the performance of a large model can be simulated by the performance of multiple small models as long as they have similar model complexity, and the verification is shown in APPENDIX A. Therefore, we use boxes with different numbers of textures as small models and render them multiple times to simulate different model complexity. For a single box, the number of textures ranges in [0,1,2,4], and the texture resolution is 2048×2048. The number of times that the box will be reused for simulation ranges in [8,64,512,4096,32768].

2.3 Experiment Setup

Hardware and Software. As for mobile devices, we use mobile phones with different capacities and operating systems, including Pixel 6 (high-end, Android 12), Mi 10 (low-end, Android 12), and iPhone 12 (high-end, iOS 14). In addition, we run our experiments on the browsers of Chrome (105.0.5195.17), Firefox (108.2.0), and Safari (14). We use the results of experiments run on Pixel 6 and Chrome with glTF models for general analysis and consider others for heterogeneity discussion. We set up a server on a macOS laptop

Table 1: Characteristics of frameworks for web-based XR.

		Three.js	React-Three-Fiber	Babylon.js	A-Frame	Playcanvas	AR.js	MindAR
Github Stars		85.3k	19.5k	18.3k	14.5k	7.7k	4.0k	1.2k
3D Rendering Engine		Y	Based on Three.js	Y	Based on Three.js	Y	Based on Three.js/A-Frame	Based on Three.js/A-Frame
Supported 3D Formats	**glTF**	Y	Y	Y	Y	Y	Y	Y
	OBJ/MTL	Y	Y	Y	Y	Only OBJ	Y	Y
	fbx	Y	Y	N	N	N	Y	Y
AR Implementation		WebXR Device API	WebXR Device API	WebXR Device API	WebXR Device API	WebXR Device API	Javascript	Javascript

and used the phone as a client to request the website. As for the frameworks, we use the latest released version.

Metrics. In order to answer how well the frameworks perform during loading, consider the loading time for the first visit to the page,i.e., first loading time (FLT), ignoring the influence of cache optimization. Since we want to filter out the influence of framework initialization, we add a button for each application, and the loading process will not start until the button is clicked. We also split the first rendering process into different parts and analyzed them separately for better understanding. As for the performance of the frameworks after loading, we mainly focus on frame rate per second (FPS). We measure the average FPS after loading for one minute. Moreover, to better understand the cause of the performance, we profile the applications by the web developer tools and get detailed information through our evaluation tool. We consider the resource usage at different times (CPU scripting usage/GPU usage), the time of different stages (network transferring/image decoding/scene calculation/scene rendering), and the time for each frame (FT).

2.4 The Evaluation Tool

To measure and analyze the performance of current frameworks, we develop an evaluation tool to extract quantitative information. The tool can get the time of different events during loading, including network transferring and image decoding. In addition, the tool can calculate CPU scripting usage and GPU usage for any period given the profiled file. Our evaluation tool is implemented based on the front end of the web developer tools [7, 10, 19, 22]. Besides, we add the functions that are needed to analyze the performance of current frameworks in detail.

2.5 Performance Analysis Sequence

From virtual to real, XR consists of virtual reality (VR), mixed reality (MR), and augmented reality (AR). Figure 1 shows the processes of XR. XR contains rendering 3D digital objects (3D rendering), capturing the physical environment by device camera (camera capturing), and obtaining information about the physical world for better interaction (real-world understanding). Real-world understanding involves environment understanding which aims to investigate the geometry information, and light estimation, which tends to get the light information for photorealistic rendering. While VR only cares about 3D rendering, MR and AR involve 3D rendering, camera capturing, and real-world understanding.

Therefore, in our performance analysis, we first discuss the performance of 3D rendering of current frameworks, which exists in almost all XR applications. Then we examine the influence of adding camera capturing and analyze the time left for real-world understanding, which is critical in MR and AR.

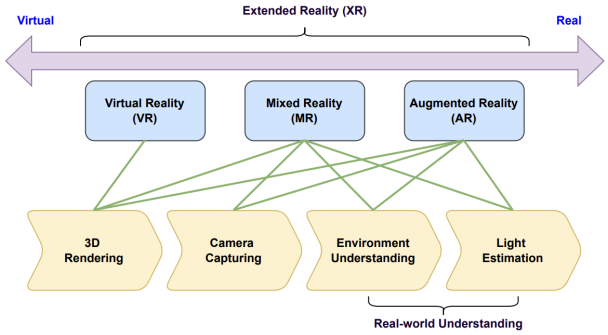

Figure 1: Processes of XR applications

Table 2: The number of top one and top two scenarios.

	THR	BBL	PC	AF	RTF
TOP1	3	0	6	3	8
TOP2	6	0	14	7	15

3 PERFORMANCE ANALYSIS

In this section, we conduct the measurement study and investigate how current frameworks perform during and after loading.

3.1 Performance of 3D Loading

Since web applications need to download all the required resources through the network when they are visited for the first time, the loading time is critical for user experience. Therefore, we first examine the 3D loading performance, which is the time spent on rendering the first frame. We conduct this experiment on five XR frameworks, i.e., Three.js, Babylon.js, Playcanvas, A-Frame, and React-three-fiber. The other two frameworks, i.e., AR.js and MindAR.js, are specifically designed for AR, of which the 3D loading process uses other frameworks.

3.1.1 Overall Performance. Figure 2 illustrates the FLT of different frameworks with different 3D model complexity. A missing bar indicates that a specific framework does not support the model. We have the following observations:

(1) **The FLT of React-three-fiber is relatively short.** Table 2 shows the number of best-performing and top-two-performing scenarios of different frameworks. A scenario is defined by the

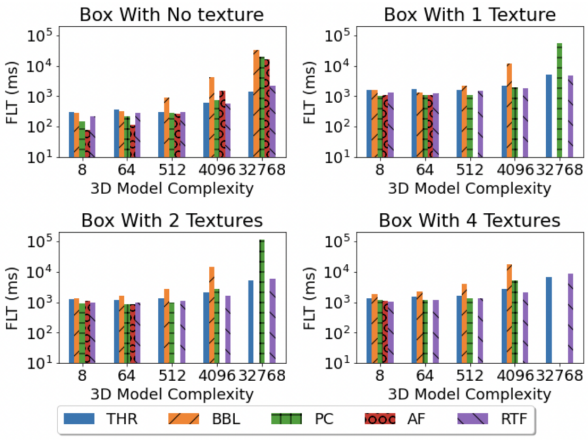

Figure 2: FLT of different frameworks and 3D model complexity (#box, #texture).

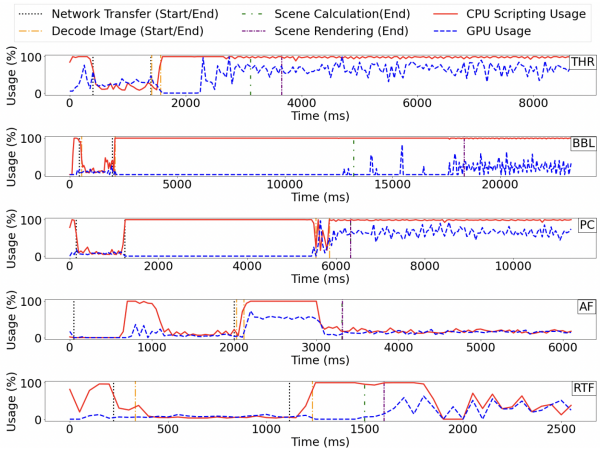

Figure 3: The CPU scripting and GPU usage of different frameworks during loading with stages marked.

Table 3: Performance gaps of FLT between the best and worst frameworks for different 3D model complexity.

#Models	Box_0	Box_1	Box_2	Box_4
8	4.1×	1.6×	1.5×	1.8×
64	3.2×	1.6×	1.9×	1.9×
512	3.3×	2.0×	2.9×	3.1×
4096	7.3×	6.7×	9.1×	8.3×
32768	24.0×	11.7×	21.7×	1.3×

Table 4: The average proportion of time of networking transferring (NT), image decoding (ID), and the sum of the two.

	THR	BBL	PC	AF	RTF
NT	37.1%	21.6%	46.8%	46.6%	63.7%
ID	2.5%	21.2%	3.1%	1.0%	4.8%
Total	39.6%	42.8%	50.0%	47.6%	68.5%

complexity of the 3D models. React-three-fiber has the highest count for both top1 and top2 performing.

(2) **A-Frame has poor support for complex models.** While A-Frame has a relatively short first-loading time for simple models, it cannot support the rendering of complex 3D models. For instance, it fails to render 512 boxes with one or two textures.

(3) **The FLT of different frameworks may vary significantly under the same situation.** Table 3 shows that when rendering the same models, Box_X indicates boxes with X textures. The FLT of the worst frameworks can be as high as 24 times the best. The gap becomes prominent when the complexity of models increases. For instance, the performance gaps are 1.5-4.1× when rendering eight boxes but rise to 6.7-9.1× when rendering 4096 boxes and even 1.3-24.0× when rendering 32768 boxes.

(4) **As the complexity of the model increase, the FLT of Babylon.js and Playcanvas increases significantly.** As we can see from Figure 2, when the model complexity increase, the average FLT of Three.js and React-three-fiber increase relatively gently. However, the average FLT of Babylon.js and Playcanvas multiply and can exceed 10s, which is very long for web loading.

†**Finding 1:** *There exists a threshold for 3D model loading. When model complexity increases, loading time first enhances slightly and then grows rapidly.*

3.1.2 Breakdown of 3D Loading. In order to better understand what happens during the 3D loading process, we divide the loading process into several stages:

• **Network transferring** is the transferring process of 3D models from server to client, including the data describing the geometric structure and textures of 3D models.

• **Image decoding** is the process of decoding texture images of 3D models. Image decoding can only begin after the end of the transfer of the first image but does not need to wait until received all images.

• **Scene calculation** is calculating 3D models and adding them to the scene. It can be partially parallel to previous stages but still need more time after the end of image decoding.

• **Scene rendering** is when the scene containing the 3D models is rendered and displayed on the screen.

Figure 4 quantitatively shows the breakdown of different stages of different frameworks for the first loading process. Each stage's time uses the model's average time with a different number of textures. Figure 3 presents the CPU scripting and GPU usage during loading 4096 boxes with four textures (8 boxes for A-Frame). Our findings are as follows:

(1) **The network transfer and image decoding process can take a relatively large part of the loading time.** Table 4 shows the average time of network transferring and image decoding. We can see that the time proportion of network transferring plus image decoding ranges from 39.6% to 68.5%. The parallel network transferring and image decoding with other stages can be crucial.

(2) **Babylon.js has a relatively high time proportion for image decoding.** During our experiments, we find that different frameworks use different ways to conduct image decoding for glTF models. While Babylon.js and Playcanvas use the main thread to decode images, React-three-fiber, A-Frame, and Three.js use web

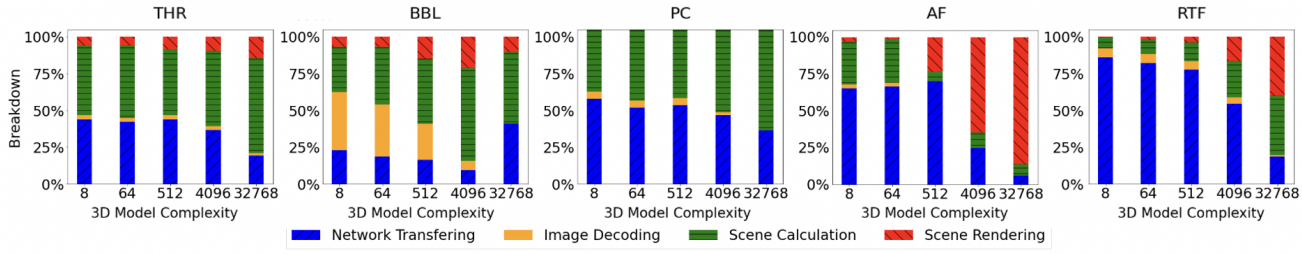

Figure 4: The breakdown of 3D loading process for different frameworks.

Table 5: Average FLT of different 3D model formats, frameworks, and numbers of models.

#Model	THR			BBL		AF		RTF		
	glTF (ms)	OBJ/MTL (ms)	FBX (ms)	glTF (ms)	OBJ/MTL (ms)	glTF (ms)	OBJ/MTL (ms)	glTF (ms)	OBJ/MTL (ms)	FBX (ms)
8	1111.5	471.3 (-57.4%)	466.3 (-58.2%)	1282.5	401.7 (-68.7%)	834.5	496.0 (-40.6%)	889.9	281.1 (-68.8%)	329.6 (-63.0%)
64	1210.0	518.1 (-57.2%)	442.1 (-63.5%)	1367.9	354.9 (-60.8%)	667.3	819.0 (+22.7%)	927.8	302.7 (-67.4%)	307.2 (-66.9%)
512	1218.4	567.0 (-52.7%)	532.2 (-56.3%)	2485.8	642.2 (-74.1%)	267.3	320.0 (+14.3%)	1064.0	328.4 (-69.1%)	344.6 (-67.6%)
4096	1876.0	584.4 (-68.8%)	742.7 (-60.4%)	12007.2	2541.2 (-78.8%)	1537.3	2090.0 (+36.0%)	1513.7	504.7 (-66.7%)	652.2 (-56.9%)
32768	4598.0	1111.7 (-75.8%)	1587.6 (-65.5%)	33135.7	16371.1 (-50.6%)	16153.0	15146.6 (-6.2%)	5383.7	1688.8 (-68.6%)	2826.1 (-47.5%)

workers to handle it without blocking the main thread and introducing the possible parallelism with other processes.

(3) **The resource usage is relatively low during network transferring and image decoding.** As shown in Figure 3, CPU scripting and GPU usage patterns remain the same in one stage and vary significantly between different stages. The resource usages descent rapidly when network transferring or image decoding start and increase again when the processes finish.

(4) **The GPU usage during loading is relatively low, except for Three.js.** Figure 3 shows that during the scene, calculation, and scene rendering process, the GPU usage of Three.js is relatively high, while other frameworks only use a small quantity of GPU.

†**Finding 2**: *FLT of frameworks can be improved through increasing parallelism and resource utilization. Increasing the parallelism of different stages of loading can reduce the FLT, like using a web worker to deal with image decoding or parallel the network transferring with other processes. Besides, the utilization of GPU has a large room for improvement to accelerate the loading process.*

3.1.3 Impact of 3D Model Formats. As mentioned earlier in Table 1, glTF is the only 3D format that has full support from all of the five frameworks. Some frameworks also support other 3D formats like OBJ/MTL and FBX. Three.js, Babylon.js, A-Frame, and React-three-fiber can load OBJ/MTL models, and Three.js and React-three-fiber can load FBX models.

Table 5 shows the average loading time of different 3D formats and frameworks. The percentages in the parentheses indicate the difference in the FLT between OBJ/MLT and FBX models compared with glTF models. As we can see, for Three.js, Babylon.js, and React-three-fiber, OBJ/MTL performs better than glTF. Moreover, for Three.js and React-three-fiber, FBX fulfills worse than glTF. We note that the FLT of different formats of 3D models is still in the same order of magnitude for the same 3D model complexity.

†**Finding 3**: *There exists no dominant 3D model format. The format with the best performance differs for different 3D model complexity and frameworks.*

3.2 Performance of 3D Rendering

We then examine the performance of 3D rendering after loading for five frameworks. Note that MindAR and AR.js conduct 3D rendering by other frameworks.

3.2.1 Overall Performance. Figure 5 illustrates the FPS of different frameworks for different 3D model complexity. Based on the results, our findings are as follows:

(1) **When the number of models increases, the FPS will remain the same and descend significantly.** As we can see from Figure 5, when the number of models does not exceed a certain threshold, the FPS will remain around 60, which is the ideal situation. However, when the model number exceeds the point, the FPS will drop continuously as the model complexity increases. We note that the threshold is different for different frameworks. Given the number of models, some frameworks may experience an FPS decrease. In contrast, others still have the best performance.

(2) **The FPS of Playcanvas is relatively high.** As Figure 5 shows, the FPS of Playcanvas is higher than other frameworks in most situations.

†**Finding 4**: *There exists a threshold for 3D model rendering. When model complexity increases, FPS remains the same until the complexity reaches a point.*

†**Finding 5**: *There exists no dominant framework. The framework that performs best after loading differs from the one during loading.*

3.2.2 Resource Utilization. To better understand the most critical factors that influence the FPS of 3D rendering in web browsers, we examine the resource usage of different model complexity for different frameworks. Figure 6 presents the average CPU scripting usage and GPU usage. In addition, we also want to know the impact of texture, and Figure 7 illustrates the results. Our observations are as follows:

(1) **The GPU usage is relatively low except for Three.js.** Figure 6 shows that the average CPU scripting usage is relatively high for different frameworks. However, GPU usage is still relatively

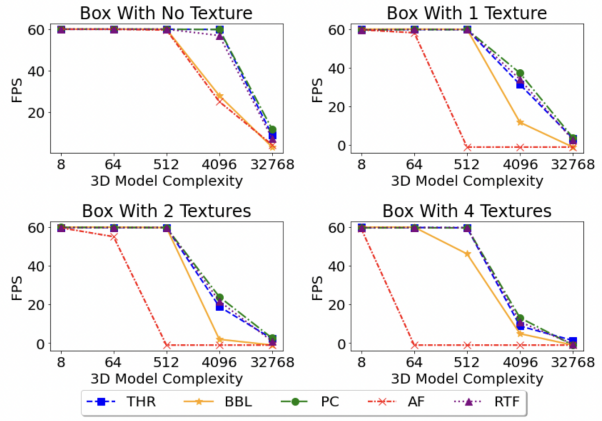

Figure 5: FPS for different frameworks and 3D model complexity (#box, #texture).

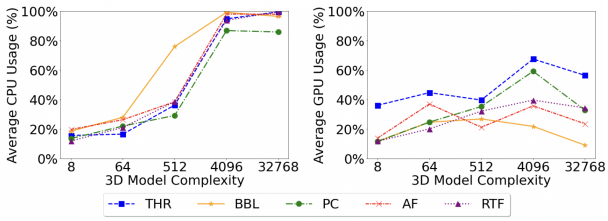

Figure 6: The average CPU scripting usage and GPU usage of different frameworks.

low for the models with increased complexity. Furthermore, the GPU usage of three.js is higher than others in most situations.

(2) **CPU scripting becomes the bottleneck for most situations.** Figure 7 shows the trend of CPU scripting usage, GPU usage, and FPS of different 3D model complexity. While CPU scripting usages of different frameworks increase at almost the same rate as 3D model complexity increases, FPS decreases at different complexity.

†Finding 6: *FPS of the frameworks can be optimized by using resources more efficiently. For instance, frameworks can use GPU to accelerate more calculations for rendering 3D. Moreover, enhancing parallelism and scheduling calculations better can also be helpful.*

3.3 Impact of Heterogeneity

We investigate the impact of heterogeneity of equipment and infrastructure on FLT and FPS, including device capacity, browsers, and operating systems.

3.3.1 Impact of Device Capacity. To investigate the impact of device capacity, we compare the performance of 3D loading and rendering on low-end mobile phones (Mi 10) and high-end mobile phones (Pixel 6). Figure 8 illustrates the results. Compared to the high-end device, the FLT of the low-end device increased by over 100% for Playcanvas and 15%-75% for other frameworks. In addition, the FPS of the low-end device is similar to the high-end device for simple 3D models. But when the model complexity increase, the low-end mobile phone will have a much lower FPS than the high-end one.

Table 6: Performance of Firefox and Safari compared to Chrome on high-end devices.

	THR	BBL	PC	AF	RTF
Firefox FPS (%)	-22.7	-6.8	-11.3	-4.4	-0.7
Firefox FLT (%)	-21.1	-25.3	-15.8	-13.5	-20.4
Safari FPS (%)	+64.9	+51.5	+34.6	+120.9	+107.0
Safari FLT (%)	-15.3	-9.4	-10.0	-7.2	-15.1

Table 7: Performance of high-end iOS device (iPhone 12) compared to high-end Android device (Pixel 6).

	THR	BBL	PC	AF	RTF
iPhone FLT (%)	-44.7	+29.6	-19.0	+11.3	-23.7
iPhone FPS (%)	-0.8	-29.9	+4.1	-21.4	-2.6

†Finding 7: *The process of 3D rendering needs further improvement so that the cross-platform advantage of the web can be brought into full play, especially for low-end devices.*

3.3.2 Impact of Browsers. We compare the performance of 3D loading and rendering in Firefox and Safari with Chrome. Table 6 illustrates the results. On Pixel 6, compared with Chrome, the FLT in Firefox decreases by 9%, but FPS drops by 19.2% on average. Firefox performs better in the aspect of loading, while Chrome wins in the aspect of rendering for all frameworks. Besides, on iPhone 12, compared with Chrome, the FLT on Safari increases by 74.8%, and FPS drops by 11.7% on average. Chrome has better performance than Safari for all frameworks.

†Finding 8: *The browser can have a non-negligible effect on performance. In particular, Safari has a much lower FPS than Chrome or Firefox.*

3.3.3 Impact of Operating Systems. We compare the performance of 3D loading and rendering on different operating systems, i.e., Android and iOS, and the results are shown in Table 7. On average, compared with Chrome on a high-end Android device (Pixel 6), the FLT of Chrome on a high-end iOS device (iPhone 12) decreases by 9.3%, but FPS drops by 10.1% on average. Moreover, the results are different for different frameworks. During loading, iOS performs better than Android, but Android performs better after loading.

†Finding 9: *The impact of the operating system on performance is not apparent.*

3.4 Impact of Camera Capturing

In order to investigate the impact of adding camera capturing, we compare the FLT and FPS of the same 3D scenes with and without camera capturing.

For FLT, adding camera capturing has an acceptable impact. Figure 9 (left) shows the relationship between FLT with and without the camera. As we can see, adding camera capturing increases the FLT by 20% on average, which is not significant for loading.

The impact of adding camera capturing to well-performed 3D scenes is limited for FPS. For pure 3D rendering, we regard a scene as well-performed when FPS equals or exceeds 59. For frameworks whose AR is based on WebXR device API, the max FPS is 30 FPS due to the implementation of the kernel of the Chrome browser. Thus, we regard scenarios with FPS equal to or larger

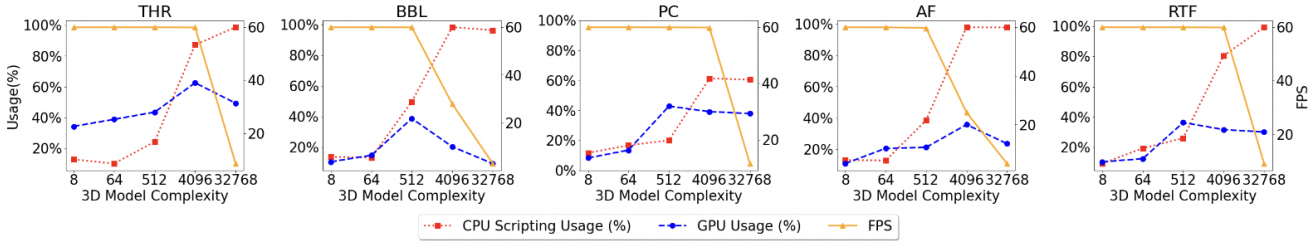

Figure 7: CPU scripting usage, GPU usage, and FPS of different 3D model complexity.

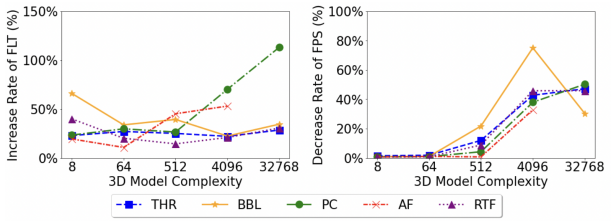

Figure 8: Comparison of performance between low-end mobile phone (Mi 10) and high-end mobile phone (Pixel 6).

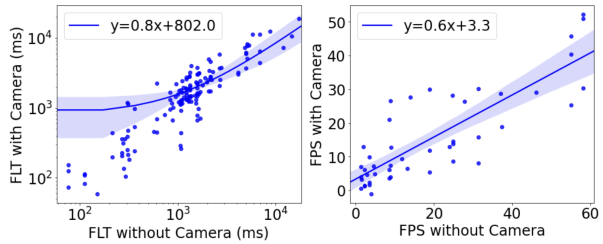

Figure 9: Relationship between FLT and FPS (not well-performed) with and without camera capturing.

than 29 as well-performed after adding camera capturing. Table 8 shows the number of well-performed 3D scenarios with and without camera capturing. For X/Y, Y is the number of scenarios well-performed without camera capturing, and X is the number of scenarios still well-performed after adding camera capturing. As we can see, for most situations, a well-performed 3D scene still has ideal FPS after adding camera capturing, except for MindAR.js based on Three.js. Figure 9 (right) presents the relationship of FPS with and without camera capturing for scenarios that are not well-performed. Adding camera capturing decreases the FPS by 40%.

†Finding 10: *Adding camera capturing to a well-performed 3D scene has limited effect on performance, which leads to a most straightforward AR application with the least effort.*

3.5 Time Left for Real-World Understanding

Real-world understanding takes data from the camera and other sensors as input and gets the result used for 3D rendering for the next frame as output. For instance, light estimation captures the physical light information and uses it to render 3D models to make them look real. Environment understanding, like tracking, gets several position vectors that will decide the position of 3D models

Table 8: The number of well-performed scenarios for different frameworks.

	THR	BBL	PC	AF	RTF
#Well-performed	13/13	10/11	13/13	6/6	13/13
	AR.js-THR	**AR.js-AF**	**MA-THR**	**MA-AF**	
#Well-performed	12/13	8/8	0/13	8/8	

Table 9: Average frame time (ms) of different frameworks and different 3D model complexity.

Frameworks	#8	#64	#512	#4096	#32768
THT	0.6	1.7	6.4	22.2	133.9
BBL	1.2	2.7	12.9	102.6	—
PC	1.4	3.2	11.4	68.8	—
AF	1.9	6.4	11.8	14.0	—
RTF	1.2	2.5	10.2	23.1	177.2
AR.js-THR	10.8	13.0	21.8	77.3	349.6
AR.js-AF	2.5	8.5	14.9	65.0	—
MA-THR	1.1	3.1	15.5	88.6	748.2
MA-AF	2.9	9.0	15.2	106.2	—

for the next frame. One functionality may have different algorithms of different performance and latency.

To investigate how much time is left for real-world understanding, we measure the average time for each frame for 3D rendering with the camera capturing added. We note that if we want to achieve the performance of 30 FPS, each frame must be finished in 33.3 ms. Table 9 shows the average frame time of different frameworks and different 3D model complexity. As we can see, the time for rendering and camera capturing can rise to over 10 ms, leaving only 20 ms for real-world understanding, which is not much for state-of-the-art methods. For instance, the method of light estimation called Xihe[67] takes 24.04ms, and the full-body tracking method called HybridTrak[70] takes 82.7ms to process RGB images to 3D pose. Although the result of one processing of light estimation or tracking may be used for several consecutive frames, one application may include several functionalities. Thus, 3D rendering and real-world understanding must be further optimized for well-performed AR and MR. Moreover, developers need to leverage model complexity, the accuracy of real-world understanding, and overall performance.

†Finding 11: *Time left for real-world understanding is insufficient if we want to have rich and real experiences.*

4 IMPLICATIONS

From our analysis and observations, we conclude several implications for different stakeholders.

For XR app developers:

(1) Findings 1 and 4 tell us that as 3D model complexity increases, the FLT increases smoothly, and FPS remains the same until model complexity reaches a certain threshold. Thus, it is free to add 3D models in web browsers without worries of negative impact on performance as long as the complexity is not overwhelmed.

(2) From Findings 3 and 5, the best framework or 3D model formats differ in different situations. So XR app developers need to choose frameworks and 3D model formats due to their condition for better performance.

(3) Findings 7, 8, and 9 suggest that when considering different equipment and infrastructure, device capacity and browser have an apparent impact on performance. Therefore, XR app developers must consider browsers and device capacity carefully for cross-platform development.

(4) By Finding 10, we can see that adding camera capturing for well-performed 3D scenes have limited influence on FLT and has almost no impact on FPS. Hence, it is easy to change well-performed 3D scenes to simple AR/MR applications with good performance by adding camera capturing.

(5) Finding 11 suggests that when 3D models are complex, the time left for real-time understanding will be insufficient. Thus, when designing XR applications that deserve real-world understanding, a trade-off between 3D model complexity and the accuracy of the real-world understanding algorithm needs to be considered.

For XR framework vendors:

(1) Findings 3 and 5 imply that different framework has different advantages when dealing with model loading for various formats and model rendering. So framework vendors can consider combining the benefits of different frameworks to achieve further improvement for current frameworks.

(2) Findings 2 and 6 show that the performance begins to drop when hardware resource is not fully utilized. Therefore, there is room to improve current frameworks by increasing resource utilization and parallelism.

For browser vendors:

(1) From Finding 11, the time left for real-world understanding is limited. Considering that there are different real-world understanding algorithms with different latency and accuracy, it is promising to provide and improve functionalities for XR by providing various abilities and algorithms for choice.

(2) Findings 2, 6, and 8 imply that the browser is vital to performance, and full use of model hardware capacity can further improve the performance. Hence, browser vendors can consider exploiting more potential capabilities of modern GPU hardware for the web, such as supporting the multi-threading of GPU.

5 RELATED WORK

Mobile XR Due to the challenge of limited resources of mobile devices, many efforts were made on the performance optimization of mobile XR, including graphics rendering, network traffic, and power conservation [32, 33, 37, 55, 58, 60, 69]. Besides, enhancing the capacity of XR is also an important research direction, such as improving the realness of light estimation, enabling follow-up effects, and supporting multi-user applications [44, 47, 53, 57, 59, 61, 66, 68, 70]. However, most prior work focuses on implementing and optimizing app-based mobile XR, and web-based model XR deserves more attention.

Web XR A couple of work conducts usability studies of web-based XR. Ma et al. [40] collected web-based VR experiments powered by the crowd. Oduor et al. [51] analyzed participants' perceptions of a web-based AR application. Besides, Qiao et al. [54] conducted a comprehensive web AR survey. Moreover, multiple studies focus on designing and implementing a novel system for new capabilities [34, 41, 64, 67]. For instance, Lam et al. [46] presented an AR web browser with an integrated context-aware AR-to-Web content recommendation service. Hobson et al. [39] developed AR graphics extensions for web applications. With all these developing systems and emerging approaches, there is still a limited understanding of the performance of current web XR frameworks.

Mobile XR benchmarks Some previous work provides benchmarks for mobile XR. Wu et al. [65] proposed a benchmark for indoor planar object tracking. Chetoui et al. [31] designed an AR benchmark for mobile devices. Tran et al. [63] focused on measuring 3D modeling in VR. Marneanu et al. [49] evaluated AR frameworks for android development. Although these studies have analyzed XR, they mainly study app-based XR and focus on a specific task.

Measurement of Web XR Hemstrom et al. [38] compared webVR and native VR, but they mainly focused on a particular application called Valtech Store, which is different from our work. Bakri et al. [26] considered QoS and QoE of 3D web content to develop web-based virtual museums, which also focus on a specific application. Karanjai et al. [42] measured web VR and implemented optimization by the findings, but they only considered two frameworks, and optimization for Web VR is their main contribution. Compared to previous work, our work selects the most popular frameworks for mobile XR in web browsers and analyzes their performance. We also consider all XR scenarios, including VR, AR, and MR.

6 CONCLUSION

In this paper, we conducted the first empirical study investigating the performance of current mobile extended reality in web browsers. After selecting seven most popular open-source active frameworks, we develop a tool to evaluate and measure their 3D rendering, camera capturing, and real-world understanding performance [1]. We quantitatively addressed extensive measurement experiments on different frameworks to understand their performance during and after loading XR applications. The results lead to interesting and valuable observations, which reveal the current landscape of mobile XR in web browsers and lead to practical implications for different stakeholders. We believe that our work can be helpful for the future development of mobile XR in web browsers.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China under the grant number 62102009.

[1] Code Release: https://doi.org/10.5281/zenodo.7627141

REFERENCES

[1] 2022. *A-Frame.* https://github.com/aframevr/aframe.
[2] 2022. *Apple App Store.* https://www.apple.com/app-store/.
[3] 2022. *ARCore.* https://developers.google.com/ar.
[4] 2022. *AR.js.* https://github.com/AR-js-org/AR.js.
[5] 2022. *ARKit.* https://developer.apple.com/augmented-reality/.
[6] 2022. *babylon.js.* https://github.com/BabylonJS/Babylon.js.
[7] 2022. *Chrome Dev Tool.* https://developer.chrome.com/docs/devtools/.
[8] 2022. *Extended Reality Market.* https://www.precedenceresearch.com/extended-reality-market.
[9] 2022. *FBX: Adaptable file format for 3D animation software.* https://www.autodesk.com/products/fbx/overview.
[10] 2022. *Firefox Dev Tool.* https://firefox-dev.tools/.
[11] 2022. *glTF RUNTIME 3D ASSET DELIVERY.* https://www.khronos.org/gltf/.
[12] 2022. *Google Play.* https://play.google.com/store/apps/?hl=en&gl=US.
[13] 2022. *MindAR.js.* https://github.com/hiukim/mind-ar-js.
[14] 2022. *OBJ File Format.* https://docs.fileformat.com/3d/obj/.
[15] 2022. *Oculus Quest.* https://www.oculus.com/experiences/quest/.
[16] 2022. *Pico VR Headset.* https://www.picoxr.com/us/G2_4K.html.
[17] 2022. *Playcanvas.* https://github.com/playcanvas/engine.
[18] 2022. *React-three-fiber.* https://github.com/pmndrs/react-three-fiber.
[19] 2022. *Safari Dev Tool.* https://support.apple.com/en-hk/guide/safari/sfri20948/mac.
[20] 2022. *three.ar.js.* https://github.com/google-ar/three.ar.js.
[21] 2022. *three.js.* https://github.com/mrdoob/three.js/.
[22] 2022. *TraceLib: A Node.js library that provides Chrome DevTools trace models to parse arbitrary trace logs.* https://github.com/saucelabs/tracelib.
[23] 2022. *tracking.js.* https://github.com/eduardolundgren/tracking.js/.
[24] 2022. *WebXR Device API.* https://www.w3.org/TR/webxr/.
[25] Christopher Andrews, Michael K Southworth, Jennifer NA Silva, and Jonathan R Silva. 2019. Extended reality in medical practice. *Current treatment options in cardiovascular medicine* 21, 4 (2019), 1–12.
[26] Hussein Bakri. 2019. Adaptivity of 3D web content in web-based virtual museums: a quality of service and quality of experience perspective.
[27] Tayebeh Baniasadi, Seyed Mohammad Ayyoubzadeh, and Niloofar Mohammadzadeh. 2020. Challenges and practical considerations in applying virtual reality in medical education and treatment. *Oman medical journal* 35, 3 (2020), e125.
[28] Eleonora Bottani and Giuseppe Vignali. 2019. Augmented reality technology in the manufacturing industry: A review of the last decade. *Iise Transactions* 51, 3 (2019), 284–310.
[29] Eric Bubar, Susan Agolini, Deana Jaber, and Amanda Wright. 2021. Three Methods for Adapting Physical Games to Virtual Formats in STEM Courses – Easy (Google Suite), Medium (Web GL Games in Unity) and Hard (Virtual Reality). In *HCI International 2021 - Posters*, Constantine Stephanidis, Margherita Antona, and Stavroula Ntoa (Eds.). Springer International Publishing, Cham, 141–147.
[30] Hsin-Yuan Chen, Ruey-Tzer Hsu, Ying-Chiao Chen, Wei-Chen Hsu, and Polly Huang. 2021. AR Game Traffic Characterization: A Case of Pokémon Go in a Flash Crowd Event. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services* (Virtual Event, Wisconsin) *(MobiSys '21)*. Association for Computing Machinery, New York, NY, USA, 493–494. https://doi.org/10.1145/3458864.3466914
[31] Sofiane Chetoui, Rahul Shahi, Seif Abdelaziz, Abhinav Golas, Farrukh Hijaz, and Sherief Reda. 2022. ARBench: Augmented Reality Benchmark For Mobile Devices. In *2022 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 242–244. https://doi.org/10.1109/ISPASS55109.2022.00035
[32] Jaewon Choi, HyeonJung Park, Jeongyeup Paek, Rajesh Krishna Balan, and JeongGil Ko. 2019. LpGL: Low-Power Graphics Library for Mobile AR Headsets. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services* (Seoul, Republic of Korea) *(MobiSys '19)*. Association for Computing Machinery, New York, NY, USA, 155–167. https://doi.org/10.1145/3307334.3326097
[33] Jaewon Choi, Hyeonjung Park, Jeongyeup Paek, and JeongGil Ko. 2018. Reactive Mesh Simplification for Augmented Reality Head Mounted Displays. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services* (Munich, Germany) *(MobiSys '18)*. Association for Computing Machinery, New York, NY, USA, 527. https://doi.org/10.1145/3210240.3210820
[34] Mariana Danielová, Pavel Janečka, Jakub Grosz, and Aleš Holý. 2019. Interactive 3D Visualizations of Laser Plasma Experiments on the Web and in VR. In *EuroVis 2019 - Posters*, João Madeiras Pereira and Renata Georgia Raidou (Eds.). The Eurographics Association. https://doi.org/10.2312/eurp.20191145
[35] János Dóka, Bálint György Nagy, Muhammad Atif Ur Rehman, Dong-Hak Kim, Byung-Seo Kim, László Toka, and Balázs Sonkoly. 2020. AR over NDN: Augmented Reality Applications and the Rise of Information Centric Networking. In *Proceedings of the SIGCOMM '20 Poster and Demo Sessions* (Virtual event) *(SIGCOMM '20)*. Association for Computing Machinery, New York, NY, USA, 44–45. https://doi.org/10.1145/3405837.3411386
[36] Laura Freina and Michela Ott. 2015. A literature review on immersive virtual reality in education: state of the art and perspectives. In *The international scientific conference elearning and software for education*, Vol. 1. 10–1007.
[37] Yunha Han, Chunggi Lee, Sanghoon Kim, and Sungahn Ko. 2019. System Architecture for Progressive Augmented Reality (Poster). In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services* (Seoul, Republic of Korea) *(MobiSys '19)*. Association for Computing Machinery, New York, NY, USA, 522–523. https://doi.org/10.1145/3307334.3328605
[38] Matteus Hemström and Anton Forsberg. 2020. A Comparison of WebVR and Native VR: Impacts on Performance and User Experience.
[39] Tanner Hobson, Jeremiah Duncan, Mohammad Raji, Aidong Lu, and Jian Huang. 2020. Alpaca: AR Graphics Extensions for Web Applications. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. 174–183. https://doi.org/10.1109/VR46266.2020.00036
[40] Chidinma U. Kalu, Stephen B. Gilbert, Jonathan W. Kelly, and Melynda Hoover. 2021. Translating Virtual Reality Research into Practice as a Way to Combat Misinformation: The DOVE Website. In *HCI International 2021 - Late Breaking Posters*, Constantine Stephanidis, Margherita Antona, and Stavroula Ntoa (Eds.). Springer International Publishing, Cham, 341–348.
[41] Chidinma U. Kalu, Stephen B. Gilbert, Jonathan W. Kelly, and Melynda Hoover. 2021. Translating Virtual Reality Research into Practice as a Way to Combat Misinformation: The DOVE Website. In *HCI International 2021 - Late Breaking Posters*, Constantine Stephanidis, Margherita Antona, and Stavroula Ntoa (Eds.). Springer International Publishing, Cham, 341–348.
[42] Rabimba Karanjai. 2018. *Optimizing Web Virtual Reality.* Ph. D. Dissertation. Rice University.
[43] Sam Kavanagh, Andrew Luxton-Reilly, Burkhard Wuensche, and Beryl Plimmer. 2017. A systematic review of virtual reality in education. *Themes in Science and Technology Education* 10, 2 (2017), 85–119.
[44] Antonio La Salandra, Piero Fraternali, and Darian Frajberg. 2018. A Location-Based Virtual Reality Application for Mountain Peak Detection. In *Companion Proceedings of the The Web Conference 2018* (Lyon, France) *(WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1206–1214. https://doi.org/10.1145/3184558.3191559
[45] Zeqi Lai, Y. Charlie Hu, Yong Cui, Linhui Sun, Ningwei Dai, and Hung-Sheng Lee. 2020. Furion: Engineering High-Quality Immersive Virtual Reality on Today's Mobile Devices. *IEEE Transactions on Mobile Computing* 19, 7 (2020), 1586–1602. https://doi.org/10.1109/TMC.2019.2913364
[46] Kit Yung Lam, Lik Hang Lee, and Pan Hui. 2021. A2W: Context-Aware Recommendation System for Mobile Augmented Reality Web Browser. In *Proceedings of the 29th ACM International Conference on Multimedia* (Virtual Event, China) *(MM '21)*. Association for Computing Machinery, New York, NY, USA, 2447–2455. https://doi.org/10.1145/3474085.3475413
[47] Tengpeng Li, Nam Son Nguyen, Xiaoqian Zhang, Teng Wang, and Bo Sheng. 2019. PROMAR: Practical Reference Object-Based Multi-User Augmented Reality (Poster). In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services* (Seoul, Republic of Korea) *(MobiSys '19)*. Association for Computing Machinery, New York, NY, USA, 531–532. https://doi.org/10.1145/3307334.3328610
[48] Vasiliki Liagkou, Dimitrios Salmas, and Chrysostomos Stylios. 2019. Realizing virtual reality learning environment for industry 4.0. *Procedia Cirp* 79 (2019), 712–717.
[49] Iulia Marneanu, Martin Ebner, and Thomas Rößler. 2014. Evaluation of Augmented Reality Frameworks for Android Development. *International Journal of Interactive Mobile Technologies (iJIM)* 8 (10 2014), 37–44. https://doi.org/10.3991/ijim.v8i4.3974
[50] Rohit Mehra, Vibhu Saujanya Sharma, Vikrant Kaulgud, Sanjay Podder, and Adam P. Burden. 2020. Towards Immersive Comprehension of Software Systems Using Augmented Reality - An Empirical Evaluation. In *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 1267–1269.
[51] Michael Oduor and Timo Perälä. 2021. Interactive Urban Play to Encourage Active Mobility: Usability Study of a Web-Based Augmented Reality Application. *Frontiers in Computer Science* 3 (2021). https://doi.org/10.3389/fcomp.2021.706162
[52] Wayne Piekarski and Bruce Thomas. 2002. ARQuake: the outdoor augmented reality gaming system. *Commun. ACM* 45, 1 (2002), 36–38.
[53] Siddhant Prakash, Alireza Bahremand, Linda D. Nguyen, and Robert LiKamWa. 2019. GLEAM – An Illumination Estimation Framework for Real-Time Photorealistic Augmented Reality on Mobile Devices (Demo). In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services* (Seoul, Republic of Korea) *(MobiSys '19)*. Association for Computing Machinery, New York, NY, USA, 659–660. https://doi.org/10.1145/3307334.3328570
[54] Xiuquan Qiao, Pei Ren, Schahram Dustdar, Ling Liu, Huadong Ma, and Junliang Chen. 2019. Web AR: A Promising Future for Mobile Augmented Reality—State of the Art, Challenges, and Insights. *Proc. IEEE* 107, 4 (2019), 651–666. https://doi.org/10.1109/JPROC.2019.2895105
[55] Arunkumar Ravichandran, Ish Kumar Jain, Rana Hegazy, Teng Wei, and Dinesh Bharadia. 2018. Facilitating Low Latency and Reliable VR over Heterogeneous Wireless Networks. In *Proceedings of the 24th Annual International Conference on*

Mobile Computing and Networking (New Delhi, India) *(MobiCom '18)*. Association for Computing Machinery, New York, NY, USA, 723–725. https://doi.org/10.1145/3241539.3267781

[56] Khyrina Airin Fariza Abu Samah, Mohd Nor Hajar Hasrol Jono, Al-Afiq Che Mohamad Zulkepli, Noor Afni Deraman, Shahadan Saad, Alya Geogiana Buja, and Lala Septem Riza. 2021. Immersive Virtual Reality in Preserving Historical Tourism. In *2021 IEEE 11th International Conference on System Engineering and Technology (ICSET)*. 234–239. https://doi.org/10.1109/ICSET53708.2021.9612577

[57] Jiacheng Shang, Si Chen, Jie Wu, and Shu Yin. 2022. ARSpy: Breaking Location-Based Multi-Player Augmented Reality Application for User Location Tracking. *IEEE Transactions on Mobile Computing* 21, 2 (feb 2022), 433–447. https://doi.org/10.1109/TMC.2020.3007740

[58] Shu Shi, Michael Hwang, Varun Gupta, and Rittwik Jana. 2019. Latency Adaptive Streaming of 8K 360 Degree Video to Mobile VR Headsets (Demo). In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services* (Seoul, Republic of Korea) *(MobiSys '19)*. Association for Computing Machinery, New York, NY, USA, 651–652. https://doi.org/10.1145/3307334.3328566

[59] Hong-Han Shuai, Yueh-Hsue Li, Chun-Chieh Feng, and Wen-Chih Peng. 2018. Four-Dimensional Shopping Mall: Sequential Group Willingness Optimization under VR Environments. In *Companion Proceedings of the The Web Conference 2018* (Lyon, France) *(WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 131–134. https://doi.org/10.1145/3184558.3186961

[60] Xiang Su, Jacky Cao, and Pan Hui. 2020. 5G Edge Enhanced Mobile Augmented Reality. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking* (London, United Kingdom) *(MobiCom '20)*. Association for Computing Machinery, New York, NY, USA, Article 64, 3 pages. https://doi.org/10.1145/3372224.3417315

[61] Shih-Wei Sun and Yi-Shan Lan. 2019. Augmented Reality Displaying Scheme in a Smart Glass Based on Relative Object Positions and Orientation Sensors. *World Wide Web* 22, 3 (may 2019), 1221–1239. https://doi.org/10.1007/s11280-018-0592-z

[62] Kevin S Tang, Derrick L Cheng, Eric Mi, and Paul B Greenberg. 2020. Augmented reality in medical education: a systematic review. *Canadian medical education journal* 11, 1 (2020), e81.

[63] Thi Tran, Gilles Foucault, and Romain Pinquié. 2022. Benchmarking of 3D Modelling in Virtual Reality. *Computer-Aided Design and Applications* 19 (03 2022), 1184–1190. https://doi.org/10.14733/cadaps.2022.1184-1190

[64] Chao Wang, Shuanq Lianq, and Jinyuan Jia. 2018. Immersing Web3D Furniture into Real Interior Images. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. 721–722. https://doi.org/10.1109/VR.2018.8446341

[65] Ziming Wu, Jiabin Guo, Shuangli Zhang, Chen Zhao, and Xiaojuan Ma. 2019. An AR Benchmark System for Indoor Planar Object Tracking. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*. 302–307. https://doi.org/10.1109/ICME.2019.00060

[66] Jingao Xu, Guoxuan Chi, Zheng Yang, Danyang Li, Qian Zhang, Qiang Ma, and Xin Miao. 2021. FollowUpAR: Enabling Follow-up Effects in Mobile AR Applications. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services* (Virtual Event, Wisconsin) *(MobiSys '21)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3458864.3467675

[67] Jackie (Junrui) Yang, Tuochao Chen, Fang Qin, Monica S. Lam, and James A. Landay. 2022. HybridTrak: Adding Full-Body Tracking to VR Using an Off-the-Shelf Webcam. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 348, 13 pages. https://doi.org/10.1145/3491102.3502045

[68] Zhijian Yang, Yu-Lin Wei, Sheng Shen, and Romit Roy Choudhury. 2020. Ear-AR: Indoor Acoustic Augmented Reality on Earphones. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking* (London, United Kingdom) *(MobiCom '20)*. Association for Computing Machinery, New York, NY, USA, Article 56, 14 pages. https://doi.org/10.1145/3372224.3419213

[69] Juheon Yi and Youngki Lee. 2020. Heimdall: Mobile GPU Coordination Platform for Augmented Reality Applications. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking* (London, United Kingdom) *(MobiCom '20)*. Association for Computing Machinery, New York, NY, USA, Article 35, 14 pages. https://doi.org/10.1145/3372224.3419192

[70] Yiqin Zhao and Tian Guo. 2021. Xihe: A 3D Vision-Based Lighting Estimation Framework for Mobile Augmented Reality. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services* (Virtual Event, Wisconsin) *(MobiSys '21)*. Association for Computing Machinery, New York, NY, USA, 28–40. https://doi.org/10.1145/3458864.3467886

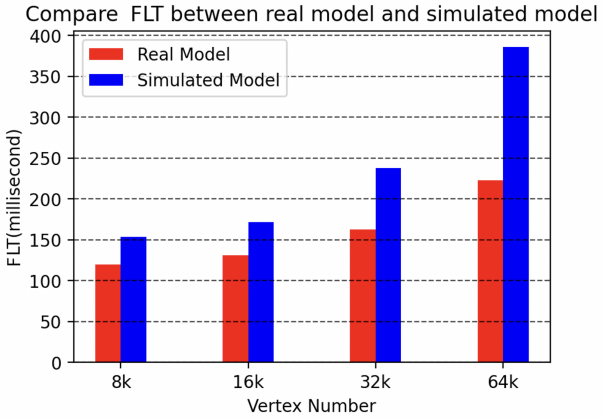

Figure 10: FLT comparison between real models and simulated models.

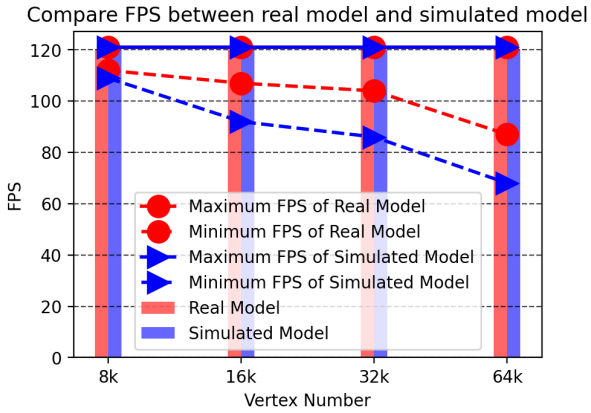

Figure 11: FPS comparison between real models and simulated models.

A VERIFICATION OF SIMULATION

Simulation by small boxes This paper uses many small boxes to simulate a complex model. We use a model object with N vertexes as a real-world model (called real model) and a scene with N/8 small cubes as a simulated model (called simulated model). We compare FLT and FPS of real models and simulated models. Figure 10 presents the FLT comparison results and Figure 11 presents the FPS comparison results.

According to Figure 10, we can see that as 3D model complexity increases, the FLT of simulated models and real models increase in the same pattern. The difference is caused by the model size, which influences the network transfer time. According to Figure 11, the maximum FPS of real and simulated models are approximately the same for all model complexity, and the minimum FPS of the actual model decrease in the same pattern. Thus, the trend and pattern of performance of real models can be estimated by the simulated models.