Design and Implementation of Augmented Reality Browser for Mobile Terminal

Dan Li and Yanlei Shang*
Beijing University of Posts and Telecommunications, Beijing, China

*Corresponding author e-mail: shangyl@bupt.edu.cn

Abstract. The rapid advancement of mobile internet technology has promoted the vigorous development of Augmented Reality (AR) applications. However, the native AR application on the mobile terminal has the disadvantages of closedness and high cross-platform cost. The Web-based application has universality and good cross-platform characteristics. Using Web as the carrier of AR applications will solve the problems of the native application. However, there is a large gap between Web applications and the native applications in the performance and efficiency of the algorithm. To solve these problems, the mobile browser's support for AR applications has been expanded in this paper. The native computing ability of the mobile phone is encapsulated into a Web-callable interface, and the hybrid rendering method is used which uses both the native and the Web to render the AR applications.

1. Introduction
Augmented reality technology refers to direct or indirect visual techniques that enhance physical, real-world constituent elements of computer-generated inputs such as sound, image or GPS data [1]. Currently, augmented reality applications are based primarily on proprietary hardware devices or native applications on the mobile side. Among them, AR hardware equipment, such as AR helmet or AR glasses, is expensive and has high requirements on the site; and the native AR application on the mobile terminal has the disadvantages of closedness and high cross-platform cost.

Web-based applications have versatility and good cross-platform features, and as a carrier of AR applications will solve the problems of native applications. However, the existing mobile browsers lack support and expandability of augmented reality, which leads to the gap between the existing Web applications in terms of algorithm performance and efficiency.

Based on its versatility and cross-platform advantages, Web-based AR technology has attracted the attention of a large number of developers at home and abroad. In order to make full use of the powerful hardware capabilities of smart terminals, developers have also studied augmented reality browsers.

The Mozilla Foundation is a non-profit organization behind the Mozilla Firefox browser. They have launched an experimental WebXR Viewer application available for download on iOS devices to help developers quickly experiment with Web-based ARs built with Web technologies and ARKit.

TBS is a browsing service provided by Tencent. It is open to the AR, and the AR capability of the browser is open so that the front end can experience the same AR capability of the native and can use the TBS cloud image recognition capability at no cost.

In summary, the research purpose of this topic is to study and implement the design and implementation of augmented reality browser for mobile terminals
2. Demand analysis
The computing ability of the Web has always been limited by the browser framework, so that it can not achieve the same good AR effect as the native application. The key to implementing an augmented reality browser is to expose the phone's native computing ability to the Web application and complete the calculations required for the AR application.

2.1. Combine the native capabilities of the mobile phone with the capabilities of the Web to achieve the functionality of an AR system
A complete AR system consists of the following basic components, as shown in Figure 1.

![Figure 1. AR system architecture.](image)

1) Capturing the real world: The background of AR applications is the real world, done by cameras on mobile devices.
2) The construction of the virtual world: on the basis of the real world, many virtual object models can be added to form a complex virtual world.
3) Rendering: Renders the captured real world as the background of the AR application; renders the virtual world into the captured real world.
4) World Tracking: When the real world changes, such as moving the camera, it is necessary to track the position and angle change information of the current camera relative to the initial time in order to render the position and angle of the virtual world relative to the real world in real time.
5) Scene analysis: In order to make the virtual world and the real world more realistic, it needs a certain degree of understanding of the real world, such as plane recognition.
6) Interacting with the virtual world: interacting with the object model of the virtual world to increase the interest of the AR application.

In the Web-based mobile AR application, WebRTC is usually used to open the camera to realize real-world capture and rendering [2]; use Three.js to realize the construction of virtual world, the rendering of virtual world and the interaction with virtual world [3].

The WebRTC-based AR application uses WebTRC technology to open the camera, capture the real-world video stream, and render it to the screen. When a real-world image is needed, the image is taken from the video stream.

In doing so, the calculation delay is very large. After the image is obtained, the image is processed, and the processed result is reflected into the virtual world. There is a large delay in these steps, which may cause the virtual world to be out of sync with the real world.

In order to alleviate the pressure on the mobile side and ensure the unification of the virtual world and the real world, a hybrid rendering concept is proposed, which encapsulates the capture and rendering of the real world into the browser, using the native capabilities of the mobile terminal, and the virtual world-related parts. The rendering is still on the Web front end.
The world tracking and scene analysis part requires a lot of calculations, and the computing ability of the Web side limits the implementation of the Web-based mobile AR application in this respect. Therefore, only a few simple application modes can be implemented. This part is also the biggest limitation of current Web-based AR applications. Based on the native mobile application, there is no such trouble. The powerful computing power of the mobile terminal ensures the realization of the world tracking and scene analysis. Therefore, this part of the ability is packaged into the browser, using the mobile end native ability to achieve, to make up for the lack of Web-based AR applications in this regard.

2.2. **Build a bridge between the Web page program and the native application**

The smart terminal itself has strong development and application capabilities, but the browser between the Web application and the terminal does not expose the native capability of the hardware terminal to the Web application developer. Therefore, this topic is to complete the research and design of the intermediate link so that the Web application can easily call the native ability of the terminal itself.

2.3. **The function that can be called by the Web page is encapsulated into an interface plugin**

In order to make our research results more reusable, the native capabilities of the hardware terminal itself are provided to the Web application in the form of APIs, so as to implement the API exposed by the Web application through the JS browser, and then invoked by the browser. The process of hardware native capabilities. This will allow Web applications to use hardware capabilities directly like native applications. The easy-to-call API allows more developers to use and develop AR applications, while providing the possibility to refine and research future Web-based AR algorithms.

3. **System Architecture Design and Implementation of Augmented Reality Browser**

Through the demand analysis, the system architecture shown in the following Figure 2 is designed.

![Figure 2. System architecture](image)

The native browser of this project is based on the iOS system and uses the Swift language. The Web interface file is implemented in the JavaScript language.

3.1. **Native AR functional layer**

3.1.1. **Capture Real World module.** The capture of the real world is the basic function of the AR application. Since the traditional video stream is not conducive to subsequent processing, and the virtual world is easily out of sync with the real world, the real-world capture is realized by using a frame picture to simulate a video stream.

The main function of this module is to open the camera to get the frame picture. Use the official ARKit library to achieve [4]. Create an ARSession, configure it and run it. ARSession uses AVCaputreSession to get a sequence of images framed by the camera. ARSession processes the
acquired image sequence and finally outputs the ARFrame. The ARFrame contains all the information of the real world scene, including the acquired image, time stamp and camera’s information.

3.1.2. Real World Rending Module. The AR application is based on the real world and needs to render the real-world image acquired by the mobile camera as the background of the application. Since the human eye sees a static picture of more than 24 frames per second, it is considered to be a continuous dynamic video. Therefore, in the case of guaranteeing the number of frames, the frame picture acquired from the real world acquisition module from the real world rendering module will be placed to apply background video.

This module is implemented using the Metal framework. The Metal framework supports GPU hardware acceleration, advanced 3D graphics rendering, and big data parallelism. The real-world image sequence captured by the Capture Real World module is rendered into the device via Metal.

3.1.3. Scene Understanding Module. The main function of scene analysis is to understand the scenes of the real world. By processing the images acquired by the real world capture module, a certain understanding of the real world is obtained to increase the authenticity of the combination of the virtual world and the real world. Scene understanding mainly includes two aspects: plane detection and face detection.

Plane detection is implemented by ARKit. When ARSession is created, plane detection is enabled. When a plane is detected, ARKit creates a coordinate system for the plane and automatically adds an ARPlaneAnchor. ARPlaneAnchor mainly contains two parameters: coordinate system origin and plane size.

Face detection using the open source library Dlib, Dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real world problems [5]. Integrated Dlib into the iOS project, detection and obtain the face feature vector.

3.1.4. World Tracking Module. The world tracking module needs to track the position and angle change information of the current camera relative to the initial time in order to render the position and angle of the virtual world relative to the real world in real time.

This module is implemented using ARKit. By processing the image sequence acquired by the real world capture module, the feature points in each frame image are identified, and according to the positional change of the feature points between successive image frames, and then compared with the information provided by the motion sensor, Finally, a high-precision device position and deflection information is obtained [6]. ARKit uses AVCaptureSession to acquire image sequences; CMMotionManager is used to obtain motion information of the device, such as rotation angle and moving distance; finally, the tracking result is encapsulated into ARFrame through ARSession.

3.2. Interface layer

Information such as the position and angle of the camera calculated by the mobile terminal is transmitted to the Web application for the construction of the virtual world. At the same time, when the Web application needs to actively acquire information such as the plane position at some time. The interface layer is divided into two parts: the interface on the Web side and the interface on the native application.

3.2.1. Web Interface. It is divided into two parts, one part is used to receive the information actively transmitted by Native, and the other part is used to actively call the function of Native.

1) Receiving end. Receive and trigger events by adding event listeners. The list of events is shown in Table 1.
Table 1. Event list.

| Event name     | Description                                           |
|----------------|-------------------------------------------------------|
| anchorsupdated | Fires when the trace is updated.                     |
| planesadded    | Discover a new plane, triggered when creating a plane anchor |
| planesupdated  | Triggered when the plane trace is updated            |
| planesremoved  | Triggered when the plane anchor is removed            |
| facesadded     | New face detected, triggered when creating face anchor|
| facesupdated   | Triggered when the face anchor is updated             |
| facesremoved   | Triggered when the face anchor is removed             |

2) Calling end. Native functions are called using the postMessage method provided by the WKScriptMessageHandler framework. The list of callable functions is shown in Table 2.

Table 2. Function list.

| Function name     | Description                                           |
|-------------------|-------------------------------------------------------|
| hideCameraFeed    | Turn off real-world video streaming through the camera |
| showCameraFeed    | Turn on real-world video streaming through the camera  |
| resetPose         | Reset camera pose                                     |
| setDepthNear      | Set the near camera value of the perspective camera    |
| setDepthFar       | Set the perspective camera far-plane value            |
| addAnchor         | Increase the point                                    |
| removeAnchor      | Remove anchor point                                   |

3.2.2. Native Interface. It is divided into two parts, one is used to actively send information to the Web, and the other is used to receive function calls on the Web side.

1) Receiving end. It is used to receive the function call information transmitted by the Web terminal through postMessage. Received via the didReceiveScriptMessage method provided by the WKScriptMessageHandler framework. The information list is the same as Table 2.

2) Calling end. When a trigger event is needed, a statement that triggers the event response is constructed and the JS statement is executed via the evaluateJavaScript function. The list of triggerable events is the same as Table 1.

3.3. User Interface Layer
As a browser, the project has a simple user interface for receiving the URL entered by the user and opening the corresponding Web page. The UI layer mainly includes four functional modules:

1) URL input box: Users can enter the URL they want to access.
2) Refresh button: Click the button to refresh the opened Web page.
3) Back button: Click the button to return to the last page you put back.
4) Web page display area: The Web page corresponding to the Web address input by the user is displayed in the area.

This layer of functionality is implemented using the UIKit framework. The UIKit framework provides a set of Classes to build and manage user interface interfaces, application objects, event controls, drawing models, windows, views, and interfaces for controlling touch screens, etc. for iOS applications.

4. Experiment
Write experimental example using the encapsulated Web-side interface. The effect is expected to be: Turn on the camera to capture the scenes in the world and identify the planes in the scene. When clicking on the plane, place a simple 3D model on the plane. The effect is shown in Figure 3. There is a chair on the screen. And after click the plane of the chair, a virtual chess will be placed on the chair. Different angles of the pieces can be observed by turning the camera.
5. Conclusion
Web-based applications have versatility and good cross-platform features. However, the computing ability of the Web is limited by the browser framework, and it is not possible to achieve better AR effects like native mobile applications. This paper uses the native ability of the mobile phone to complete the part of the AR application that requires a lot of calculation and realizes the mutual invocation of the Web and the original, and finally encapsulates the native ability that the Web can call into a convenient API. At the same time, using the hybrid rendering method, the rendering of the AR scene is completed by using the native and the Web together. Expanded support for AR applications in mobile browsers. By writing a test case, the functionality of the augmented reality browser has been tested and proved the usability of the project.

Acknowledgments
This work was supported by the National Natural Science Foundation of China under Grants No. 61671081 and No. 61720106007, the Beijing Natural Science Foundation under Grant No. 4172042, and the 111 project (B18008), the Fundamental Research Funds for the Central Universities under Grant No. 2018XKJC01.

References
[1] Augmented reality. Wikipedia[EB/OL].[2012-07-19].http://en.wikipedia.org/wiki/Augmented_reality.
[2] S. Gunkel, M. Prins, H. Stokking and O. Niamut, "WebVR meets WebRTC: Towards 360-degree social VR experiences," 2017 IEEE Virtual Reality (VR), Los Angeles, CA, 2017, pp. 457-458.
[3] A. I. Subratty Mohammad, J. T. Tonta Yannick and R. K. Moloo, "Increasing efficiency in visualizing large sets of 3D models on the Web," 2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), Coimbatore, 2015, pp. 1-4.
[4] E. Ito, T. Ozono and T. Shintani, "Extracting Paper Sticky Notes with Visual-Inertial Odometry of ARKit," 2018 5th International Conference on Computational Science/ Intelligence and Applied Informatics (CSII), Yonago, 2018, pp. 37-42.
[5] Davis E. King, "Dlib-ml: A Machine Learning Toolkit", Journal of Machine Learning Research, vol. 10, pp. 1755-1758, 2009.
[6] Apple. Understanding World Tracking in ARKit[EB/OL]. https://developer.apple.com/documentation/arkit/understanding_world_tracking_in_arkit.