2021 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW) | 978-1-6654-4057-8/20/$31.00 ©2021 IEEE | DOI: 10.1109/VRW52623.2021.00121

WebPoseEstimator: A Fundamental and Flexible Pose Estimation Framework for Mobile Web AR

Yakun Huang [1], Xiuquan Qiao *[1], Zhijie Tan[1], Jianwei Zhang[2], and Jiulin Li[3]

[1]State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China.
[2]Capinfo Company Limited, Beijing, China.
[3]Beijing National Speed Staking Oval Operation Company Limited, Beijing, China

ABSTRACT

Exploring immersive augmented reality (AR) on the cross-platform web has attracted a growing interest. We implement WebPoseEstimator, a fundamental and flexible pose estimation framework, running on the common web platform, and providing the core capability to enable true Web AR. WebPoseEstimator provides a first real-time pose estimation framework that optimizes the loosely coupled multi-sensor fusion framework to flexibly adapt heterogeneous mobile devices. We also introduce how to optimize and compile such computationally intensive pose estimation from C++ source code into JavaScript profiles of less than 2 MB, thus supporting the fundamental and underlying capability for implementing true Web AR.

Index Terms: Computing methodologies—Computer graphics—Graphics systems and interfaces—Mixed/augmented reality.

1 INTRODUCTION

With the continuous improvement of 5G commercial deployment and the landing of artificial intelligent (AI) technologies, mobile augmented reality (AR) applications have also ushered in vigorous development [1]. Currently, most mobile AR applications rely on expensive dedicated devices, such as HoloLens, Nreal glasses, etc., which require spending manpower and financial resources to develop different AR applications for adapting various devices. Besides, people today own one or more mobile devices such as smartphones and tablets, using the embedded SDKs such as ARCore and ARKit to develop an immersive experience, which has become a more popular way. However, it requires users to download and install specific applications for AR experience, such as AR games, AR navigations, etc. This goes against the willingness that most people do not want to download third-party applications, but just want to experience AR applications quickly. Therefore, it is difficult to spread mobile AR services among users on a large scale like short videos, small games, and MiNi programs. In this work, we focus on how to implement the fundamental and flexible pose estimation on the web, which is the core of implementing the AR engine, to provide more underlying capabilities for Web AR, such as plane detection, 6-DoF tracking, and point cloud reconstruction. Currently, there are mainly three methods to implement Web AR applications. The first approach is a pure front-end solution based on a general web browser, which uses WebRTC to obtain real-time video stream, and implements target recognition, tracking, and rendering based on the third-party JavaScript libraries. Although it is easy to provide web developers with user-friendly APIs for fast building Web AR applications, most of their implementations require designing specific markers and locations to bind the real world with the virtual world for 6-DoF tracking, lacking direct APIs for providing real-time pose estimation, plane detection, and tracking. The second approach includes extending the browser tag and JavaScript engine to implement the convenient AR model rendering on the web. As for extending the JavaScript engine at the browser function level, it embeds augmented reality computing-intensive algorithms into the web and provides a JavaScript interface on the upper layer. Another typical extension method is to use ARCore and ARKit as an AR engine and encapsulate web interface in the upper layer to develop web-oriented WebARonARKit and WebARonARCore. The more novel and promising methods are to explore WebXR device API to simulate various compatible devices, such as HTC Vive, Oculus Go Samsung Gear etc. In addition to the above two pure web-based implementation schemes, cloud-based Web AR, especially in the multi-user collaboration uses the cloud platform with rich computing resources to implement multi-user interaction. However, there is a serious problem of high communication delay and poor service experience under the traditional network service architecture. These methods explore the possibility of implementing AR on the web platform from different perspectives. However, none of them can be called true Web AR, mainly due to the lack of the fundamental capability of providing lightweight pose estimation on the web smoothly. This is also the core capability of implementing AR engine when compared with native AR applications with underlying SDK, and greatly limits the user's interest in current Web AR applications.

In this work, we focus on the pervasive web and propose a fundamental and flexible pose estimation framework, named WebPoseEstimator, based on the existing web-supported APIs for the first time to provide the corresponding low-level ability to implement pose estimation for various devices. We first optimize the loosely coupled multi-sensor fusion pose framework MSF [2] to flexibly adapt the heterogeneous mobile devices according to whether the current device supports IMU or camera. We mainly simplify the initial MSF in C++ by removing the redundant parts and optimizing the core functions to support the web platform. Then, we leverage Emscripten [3] to compile our simplified MSF from C++ source code into JavaScript profiles that can run on the common web platform smoothly. Also, we rewrite and optimize unsupported instructions and system libraries on the web platform, especially for the acceleration and optimization of Streaming SIMD Extensions (SSE) and NEON instruction sets for Inter and ARM platform, respectively.

2 DESIGN OF WEBPOSEESTIMATOR

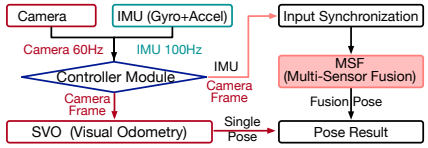


Figure 1: Overview of the WebPoseEstimator framework.

Based on the above analysis, we describe our web-oriented pose estimation framework based on MSF that is a lightweight loosely coupled scheme [2]. As shown in Figure. 1, WebPoseEstimator mainly includes data streaming management module, pose estimation module from SVO [4], synchronization module, and multi-sensor fusion pose estimation module. We also introduce the whole process of SVO that mainly supports the camera pose estimation, and analyze the implementation of multi-thread management and

*E-mail:qiaoxq@bupt.edu.cn

978-1-6654-4057-8/21/$31.00 ©2021 IEEE
DOI 10.1109/VRW52623.2021.00121

optimization of SVO on the web in Figure. 2. There are two parallel threads in SVO, one is to estimate the camera pose and another is to extend the map of the unknown environment. We do not introduce more details about SVO and you can acquire more details from [4]. Thus, to implement these two threads on the web, we only use another two web workers to handle the estimation and mapping, in which these two workers can be easily managed by the main thread of the whole JavaScript.

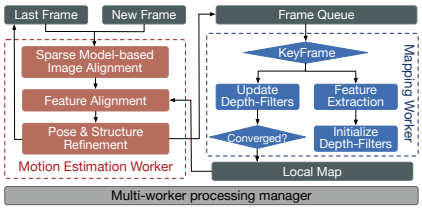


Figure 2: Web-based SVO estimation with mobile web worker.

3 Implement WebPoseEstimator in JavaScript

In this section, we focus on how to compile and generate the WebPoseEstimator with available JavaScript library by using Enscripten, an LLVM-based source-to-source compiler that can convert C++ source code into JavaScript and efficient WASM in Figure. 3. First, to reduce the size of compiled profiles of WebPoseEstimator, it is necessary to remove, simplify and rewrite the components and dependency libraries that are originally suitable for specific platforms such as ROS and Linux in the source code of MSF. Also, we optimize the lack of unsupported instructions and system libraries on the web, especially for SIMD. For instance, we mainly optimize the SIMD instructions to support SSE and NEON on the cross-platform web for vectorization acceleration. Next, we modify the core binding in C++ for the binding generator to export interfaces of classes and functions to JavaScript profiles. Similar to OpenCV.js [5], we can generate a glue code that maps JavaScript symbols to C++ symbols, and compile the rest of the functions and classes into available JavaScript with these binding information and function whitelists. For this first version of WebPoseEstimator, we mainly support OpenCV data structure (CV: mat) extracted from OpenCV, and main pose estimation interface with the video stream input and IMU input. Besides, another important step is how to complete the configuration from CMake instruction to emccmake. Before that, we need to recompile some function libraries that are only used in EMCC, such as math.h. We need to compile from the C++ source code in advance to the ".a" format supporting in emccmake. Finally, based on the above configuration and optimization, we can get WASM and main JavaScript profile from the output of the toolchain, as well as the implementation of multi-thread management worker.js.

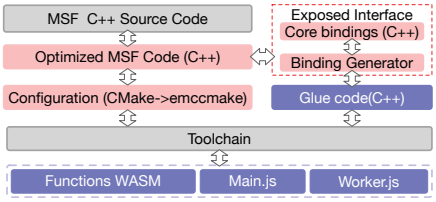
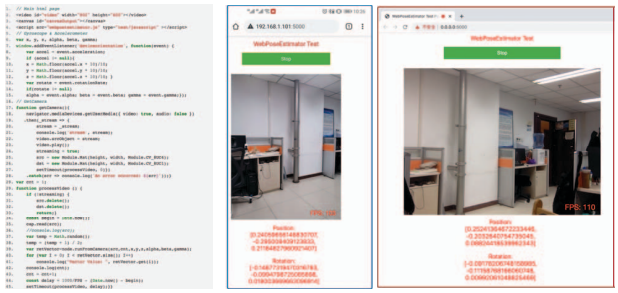


Figure 3: Optimizing and translating MSF from C++ source code to WASM and JavaScript.The gray block represents the existing code and tools, red represents our contribution code, and blue represents the final generated profiles.

4 A case study for using WebPoseEstimator

We give a case study in Figure. 4(a) to show how to use WebPoseEstimator for obtaining and outputing real-time pose estimation results. This example runs directly on the HTML page, and the first line defines the video tag to obtain a video stream from the camera. The second line uses a canvas element to output real-time results and processing frame rate, and the third line invoking our WebPoseEstimator library like the ordinary JavaScript. We add an event listener to the IMU sensor using DeviceOrientationEvent from the window object, so as to obtain real-time angular velocity and acceleration information from line 6 to line 16. For sampling video stream, we use *MediaDevices.getUserMedia()* in line 18 from the camera of the mobile device. Before running the example, we have used the existing popular calibration method to calibrate the camera-IMU. Thus, when open the test page, it calls 27 line to execute the main processor can output the results. We illustrate two snapshots of running the test page with Chrome on the mobile device and the PC in Figure. 4(b) and Figure. 4(c) respectively.



(a) A case study of using WebPoseEstimator in JavaScript (b) Running test page on the mobile device (c) Running test page on the PC

Figure 4: A case study and the WebPoseEstimator testing.

5 Conclusion

We implement a lightweight and flexible pose estimation framework on the web platform to provide fundamental capabilities for true Web AR services. For the purpose of smoothly running pose estimation with JavaScript like ordinary libraries, we simplify and optimize the efficient MSF framework, and employing Enscripten to compile it for the web. More importantly, we are first to reveal how to implement real-time pose estimation with JavaScript from the initial C++ source code. This is also useful to extend more fundamental capabilities from the traditional ROS environment to the web platform. Third, we also provide a general interface of pose estimation to easily provide many immersive services.

Acknowledgments

This research was supported in part by the National Key R&D Program of China under Grant 2019YFF0301500, in part by the National Natural Science Foundation of China (NSFC) under Grant 61671081, in part by the Funds for International Cooperation and Exchange of NSFC under Grant 61720106007, in part by the 111 Project under Grant B18008, in part by the Fundamental Research Funds for the Central Universities under Grant 2018XKJC01.

References

[1] Xiuquan Qiao, Pei Ren, Schahram Dustdar, Ling Liu, Huadong Ma, and Junliang Chen. Web ar: A promising future for mobile augmented reality—state of the art, challenges, and insights. *Proceedings of the IEEE*, 107(4):651–666, 2019.

[2] Stephan Weiss and Roland Siegwart. Real-time metric state estimation for modular vision-inertial systems. In *2011 IEEE international conference on robotics and automation*, pages 4531–4537. IEEE, 2011.

[3] Alon Zakai. Emscripten: an llvm-to-javascript compiler. In *ACM international conference companion on Object oriented programming systems languages and applications companion*, pages 301–312, 2011.

[4] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *IEEE international conference on robotics and automation*, pages 15–22. IEEE, 2014.

[5] Sajjad Taheri, Alexander Vedienbaum, Alexandru Nicolau, Ningxin Hu, and Mohammad R Haghighat. Opencv. js: Computer vision processing for the open web platform. In *Proceedings of the 9th ACM Multimedia Systems Conference*, pages 478–483, 2018.