IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 12, 2020, accepted April 22, 2020, date of publication May 11, 2020, date of current version May 21, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2993613

Rendering Optimization for Mobile Web 3D Based on Animation Data Separation and On-Demand Loading

LIANG LI[1], XIUQUAN QIAO[2], QIONG LU[1], PEI REN[2], AND RUIBIN LIN[1]
[1]School of Electronics and Information, Communication University of Zhejiang, Hangzhou 310037, China
[2]State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China

Corresponding author: Xiuquan Qiao (qiaoxq@bupt.edu.cn)

This work was supported in part by the Basic Public Welfare Projects in Zhejiang, China, under Grant LGG19F020002, in part by the National Natural Science Foundation of China 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 Beijing Natural Science Foundation under Grant 4172042, and in part by the Fundamental Research Funds for the Central Universities under Grant 2018XKJC01.

ABSTRACT Based on advances in image processing technology and Web-enabling technologies for mobile devices, mobile Augmented Reality (AR) and Virtual Reality (VR) has developed rapidly. The rendering and interaction of 3D models is an important part of AR and VR applications and is closely related to user experience. However, since the existing WebGL 3D JavaScript libraries for Web-based mobile 3D (represented by three.js and babylon.js) load the entire model file at once, large-size 3D models with complex interactions cannot be rendered smoothly due to limited data transmission, the weak computation capabilities of mobile Web browsers, and the latency of 3D model rendering. In this paper, we first propose model-animation data separation and an on-demand loading mechanism to improve the data request and loading process of Web 3D models. The main mechanisms are the following: (1) The model data are segmented into topological data and animation data sequences, and only the necessary data of the model are loaded when the Web-based mobile 3D model is first rendered. (2) The 3D model animation data sequence is semantically decomposed, and a multigranular model animation data service is established to provide continuous animation data support. (3) An asynchronous request-response mechanism is used to optimize the loading method of the model data. The model rendering mechanism uses an on-demand request and rendering method to transform the centralized loading process of the 3D model into a decentralized process. According to the testing and verification results, this optimization method can reduce the latency of mobile Web 3D in model data transmission and rendering by 24.72% for the experiment models. The interaction experience of Web-based mobile AR and VR is substantially improved relative to existing Web 3D rendering engines and rendering mechanisms, especially in complex interactive service scenarios.

INDEX TERMS Mobile Web 3D, rendering interactive computing, on-demand loading, interfacing data services, augmented reality, virtual reality.

I. INTRODUCTION

3D model interaction, as a new type of information interaction mode, provides a more intuitive and efficient means of information presentation than the 2D plane method. With advances in network technology and improvements in the performance of user terminal equipment, this interactive technology is becoming increasingly popular. With the advent of mobile Augmented Reality (AR) and Virtual Reality (VR) technologies, this 3D model interaction technology has received widespread attention from academia and industry as a primary enabling technology [1]. However, app-based 3D application solutions require users to download and install applications, and because of the limitations of portability, cross-platform operation and information interoperability, this requirement significantly hinders large-scale cross-platform application and promotion. Therefore Web-based 3D applications offer new solutions. Web-based

The associate editor coordinating the review of this manuscript and approving it for publication was Guangjie Han.

3D applications in browsers are portable and versatile, providing the foundation for sharing 3D information across platforms [2].

3D model rendering is a key supporting technology for Web-based mobile AR adn VR applications. However, efficient rendering and dynamic interaction of 3D models for Web-based mobile AR and VR face challenges. On the one hand, due to the low efficiency of JavaScript interpretation, there are some shortcomings in computing power between the mobile Web platforms and native mobile applications. Because of insufficient computing power, applications cannot effectively process large amounts of data and complex calculations, which limits the carrying capacity of the mobile Web for large applications. On the other hand, 3D model files have more data, such as text, images, and animations, than do 2D rendered elements. With the increasing complexity of service requirements, such models also contain a large quantity of attribute data, especially for animation data, and complex interaction requirements further increase the amount of data used for mobile 3D rendering [3]. Moreover, because of the above two aspects, the computing power of the mobile Web platform and the data size of the 3D model affect the mobile Web AR and VR application user experience [4].

Since the advent of WebGL in 2011 [5], third-party JavaScript 3D rendering engines based on WebGL [6] (such as three.js and babylon.js) have been implemented for loading entire model files at one time. This mechanism has the following problems in mobile Web 3D applications. First, significant model loading rendering delay caused by one-time model data loading and rendering mechanism based on the synchronous communication mode: In the synchronous data communication mode, the client needs to wait for all model data transmission to complete before loading and rendering. On the one hand, a lack of bandwidth or instability of the mobile wireless network will cause delays in the loading of the model. Moreover, for mobile Web platforms with weak computing power, there will be a considerable delay in rendering a large 3D model at once, and the browser page may even appear to be stuck. Traditionally, the progressive mesh method used in computer 3D applications solves the problem of large amounts of data latency. However, because the model decompression process is complicated, the mobile Web platform will face increased computing pressure, which will lead to an increase in user response delays, thereby causing customers to abandon this method [7]. The end user cannot afford large-scale, complex interactive mobile 3D applications. Therefore, an asynchronous and decentralized 3D model transmission method is needed to solve the network congestion caused by one-time loading. Second, the problem of redundant computing caused by the one-time model data loading and rendering mechanism: In addition to the model topology data, a 3D model also carries a large amount of data related to user dynamic interaction requirements (such as animation data), which is initialized in the model. These data are not needed for rendering but are used in subsequent user interactions. However, the one-time model loading and rendering mechanism fully loads and renders the model structure data, animation data, etc., which brings additional loading and calculation pressure to the mobile Web platform with weak computational power, which makes the calculation response delay of the service longer. In the traditional cloud and mobile edge-side operation offloading method [8], due to the lack of computing power of mobile terminals, in the cellular network environment, the 3D model rendering calculation of complex interactions and large data volume result in increased communication, energy costs, and user service delays. Therefore, there is a need for an effective method for mitigating model initialization rendering and full rendering of data to alleviate the computational pressure of Web-based mobile terminals.

Based on the discussion of the above two aspects, using a one-time model loading rendering mechanism to build more complex mobile Web 3D applications may lead to severe problems in service delay or even service interruption. Therefore, it is vital to study a 3D model loading and rendering mechanism with sophisticated interactive capabilities for mobile Web platform environments.

To solve the problem that the existing mainstream WebGL-based Web 3D rendering method causes excessive time delay due to the one-time loading of model data and is thus unable to construct complex interactive services and high-quality 3D environments, this paper proposes a 3D model animation data on-demand loading and persistence service mechanism. Via the separation of model data and animation data, creation of a multigranularity animation data service and implementation of an on-demand loading method, this approach reduces the amount of model data initially loaded for Web 3D rendering to minimize the computational pressure of the mobile device and the delay of the application through optimization.

The reduction in the amount of 3D model data loaded and calculated by the mobile device in a unit time effectively improves the interactive response delay of the mobile Web 3D model, thereby improving the user experience. Specifically, first, the method optimizes the 3D model data service. This method converts the 3D mesh model into a JSON 3D (JD) data format. The JD model is segmented into topology data and animation data sequences of the model by the server-side application and provides independent data services for the client. Second, the method is optimized for 3D model data transmission. The loading method of the WebGL plugin is improved to realize asynchronous loading of 3D model data. At the same time, the method provides continuous animation data service based on asynchronous communication through the establishment of a multigranular model animation operation management mechanism. Third, the method is optimized for 3D model data rendering. The rendering method of the WebGL plug-in is improved, and on-demand loading of animation data according to the user's individual needs is implemented.

II. RELATED WORK

At present, in terms of browser-based 3D scene construction and optimization work, some researchers have proposed three-dimensional scene rendering and methods for mobile computing offload. In this section, we mainly overview and discuss the approaches for 3D scene construction and rendering acceleration on mobile browser platforms.

A. PROGRESSIVE MESH COMPRESSION AND STREAMING LOADING

The progressive transmission process is based on the data transmission method of streaming media. First, the original mesh model is compressed into a basic mesh model and a continuous stream file. In the process of receiving by the client, the simple basic mesh model is transmitted first, and the client performs a rendering calculation on the basic model of low data volume, starts the transmission at the same time, sends the stream file to the client in the form of the transport stream, and receives the client. After decoding, the details of the model are gradually refined until the model data are fully restored.

For the first time, hope propose a scheme for progressively transmitting 3D network model data based on grid simplification and obtained a continuous network model data via an edge folding operation. On the client side, the inverse operation of edge folding is performed via the method of vertex splitting to realize the multiresolution model data transmission process [9]. Scholars further optimized the method on this basis: for example, Pajarola and Rossiggnac proposed Compressive Progressive Mesh technology [10]. In recent years, with the development of technology, a more effective method for progressive compression of models has been introduced. Maglo *et al*. proposed a new random incremental lossless manifold triangle mesh compression algorithm. It is algorithmically allowed to input different parts of the model mesh data with different detail confinements during decompression to generate a smooth transition between adjacent regions, and the adjacent areas are decompressed with varying levels of detail [11]. Caillaud *et al*. proposed a progressive compression algorithm for texture surface meshes. The algorithm solves the discontinuities in polygon nonmanifold meshes and texture mappings. Iterative batch simplification is applied in the algorithm process to maintain the geometry and texture mapping, thus creating a high-quality level of detail for the model [12]. El-Leithy *et al*. introduced a three-dimensional semiregular mesh progressive compression technique based on self-organizing feature mapping. This method is based on multiresolution decomposition via the wavelet transform and quantizes coefficients using a Self-Organizing Feature Map (SOFM) as a vector. The quantizer improves the visual quality of the reconstructed mesh [4]. Krivokuca *et al*. proposed a redundant linear combination of eigenvectors of lattice Laplacian matrices and selected the atoms of the matrix eigenvectors via the matching pursuit algorithm. Compared with the traditional orthogonal basis method, this method has higher efficiency [13], [14].

For the mobile side, Yan proposed an effective progressive transmission model for the resolution of mobile devices and the characteristics of network instability [15]. Patney and Zakas offered a new incremental transmission algorithm for the attributes of mobile devices based on the screen characteristics of mobile devices [16], [17]. For browsers, Kai designed and implemented a 3D mesh model transmission technique using streaming compression on the Web through JavaScript and WebGL [18]. By using the similarity of 3D models, the data reduction method of component similarity in the model is performed, and all the processed data are organized using the structure of the lightweight scene graph to increase the efficiency of the model transmission and loading process [19]. For network transmission, a new container file format is proposed to address the shortcomings of HTTP request data. The method is optimized for progressive, Web-based 3D mesh data transmission with a minimal number of HTTP requests, highly configurable, and more efficient and flexible than traditional formats because it enables accurate progressive transmission of geometric data. Features such as partial geometry sharing between grids, direct graphics processing unit uploads, and interleaved transmission of geometry and texture data have been implemented [20], [21].

At the same time, with the development of technologies such as machine learning and neural networks, an updated method was proposed to realize the storage and transmission of 3D grid data using related technologies. Lalos proposed a Bayesian learning algorithm to reduce the coding complexity by using the multivariate Gaussian distribution and the expected value maximization method for the subsets without affecting the visual quality of the model [22].

Compared with computing environments such as PC Web and mobile applications, mobile Web has a greater delay in the progressive mesh splitting operation. This delay is due to the limitation of mobile computing capability and mobile browser security due to the rendering and computation of 3D scenes and 3D models on the browser. Additionally, the need to invoke the underlying computing and memory resources of the intelligent mobile terminal via the browser-related interfaces of the intelligent mobile terminal causes delay.

B. COMPRESSION METHOD BASED ON MOBILE DEVICES AND BROWSER

Because mobile devices or browsers and the traditional computer have many different aspects in terms of model rendering, interactive computing, and network transmission, the current research focuses mainly on the migration of mobile computing.

To overcome the shortage of computing resources of mobile devices, another major innovation in the past decade has been Mobile Cloud Computing (MCC), which allows users to offload compute-intensive tasks to many powerful cloud servers deployed on remote cloud platforms for processing [23]. In computing offload migration, the first step is accessing a cloud computing device, such as Amazon EC2, Azure, etc., through the network to help the invention [24]

to implement the computing process. The remote cloud system relies on code offloading to reduce the amount of data exchange in the communication. Because the code granularity introduces different levels of abstraction to manipulate tasks, multiple frameworks implement different levels of granularity. MAUI, ThinkAir [25] and COSMOS [26] perform offloading at the method level. Clonecloud [27] and COMET implement frameworks for offloading at the thread level. Other work employs different levels of granularity, such as classes [28] or jobs [29]. However, since the latency of communication can suddenly change, the opportunistic moments of offloading in a remote cloud system are different. Furthermore, distant cloud offloading is sensitive to multiple parameters of the system (the context of the device), which means that determining opportunistic moments for offloading is challenging [30], [31].

To mitigate the long processing delay and high power consumption of mobile AR applications, scholars recently proposed Mobile Edge Computing (MEC) [32]. The 3rd Generation Partnership Project (3GPP) also considered this technology in its future standards. By performing data processing at the edge of the network, MEC can effectively overcome network congestion and long delays in the cloud computing system [33]. By distributing traditional centralized cloud computing resources at the edge of the mobile network, MEC provides mobile users an adjacent computing environment and offers a variety of benefits, including ultra-low latency, real-time access, and location-aware services [34], [35]. On the other hand, by utilizing the proximity of mobile users in MEC and the abundant computing power available via MCC, effective collaboration between cloud and edge computing can further improve the system performance. Several layered edge-to-cloud architectures have been proposed to leverage the computing power of edge and cloud servers [36], [37].

III. METHOD ANALYSIS

In this section, we provide an overview of the proposed Web-based mobile 3D rendering optimization approach for the WebGL application framework and describe how to separate and load model data on demand.

A. JD MODEL FORMAT ANALYSIS

The main design goal of the Web-based mobile 3D rendering optimization method under the application framework of WebGL is to reduce the volume of data loaded by the mobile device, especially the amount of data loaded the first time. Through the integrated application of the optimization model and the existing model progressive compression, streaming and other technical solutions, the Web-based method relieves the pressure of data rendering and interaction of the client. At the same time, it improves the interactive response efficiency of the client. To ensure the validity of data reading and analysis, it is natural to consider using the JD format, which is exported via Blender software, as the research object in the optimization method. Unlike the standard 3D model formats on the market, such as fbx, obj, mtl, etc., the JD format data describe the object in the form of key-value pairs, which is more suitable for the JavaScript environment of the browser. Moreover, JD files take up less memory and have a more explicit data structure than other file formats that can be loaded by three.js, and the spilled server facilitates the storage management of model data. Based on the above reality, the Web-based mobile 3D rendering optimization method determines the internal data structure of the model through the internal structure of the model data and further optimizes the model transmission and rendering process.

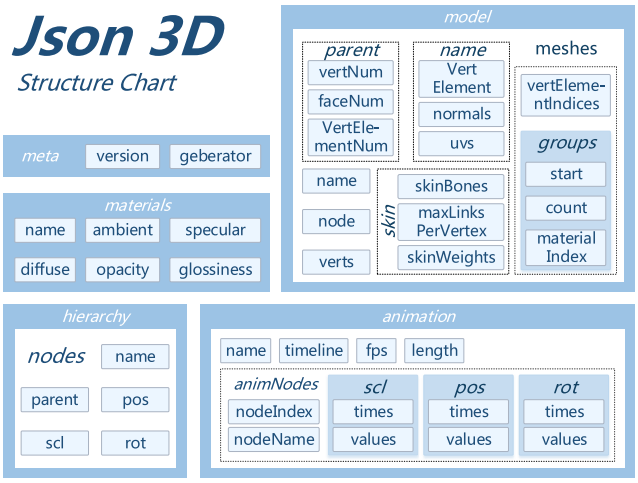


FIGURE 1. The structure of a JD file.

Usually, the descriptive data of a JD model is divided into five parts: file source data (meta), structural data (model), material and texture data (materials), animation data (animation), and hierarchical structure data (hierarchy), as shown in Fig. 1. The file source data mainly describe the model file version, encoding and other basic information related to file parsing. The structural data mainly describe the vertex position information of the 3D model and the connections between the vertices. The material and texture data mostly represent the texture of the 3D model material and other related information. The animation data define the animation of relevant information data carried in the model. The hierarchical structure data are mainly model-level description information generated according to the structure of the model.

Part of the data in the model data composition is determined only by the demand, including the following:

- The file source data describe the model file and present relatively fixed attributes after the related file information is established.
- The structural data and hierarchical structure data are determined by the complexity of the service and the presentation requirements of the service and are presented differently according to the user's visual interaction requirements and the physical display characteristics of the client.

The topology data of the model formed by these data relate mainly to the physical structure and services content of the 3D

model. The amount of topology data of the initially loaded model and the amount of completion loaded in 3D rendering can be reduced mainly via model progressive transmission and model compression.

B. DISADVANTAGES OF ONE-TIME LOADING ANIMATIONS BASED ON FILES

In the current related application, a file-based one-time synchronous loading method is adopted for 3D model data. Significant model loading rendering delay is caused by the one-time model data loading and rendering mechanism based on synchronous communication mode. In synchronous data communication mode, the client must wait for all model data transmission to be completed before loading and rendering, which will lead to the following issues:

- For animation data, under different service environments, different users, and various individual requirements, it is critical to choose different interaction processes to retrieve different animation data. This diverse environment leads to a model file containing a large volume of redundant animation data.
- Further, for the interior of the animation data, there is a certain amount of repetitive animation frame data between the plurality of response animations. Under the request-response data loading mechanism, repetitive animation frame data are redundantly transmitted between the server and the client and repeatedly written into the client's computational cache.
- For complex multiframe sequence animation data, under the one-time request-response data loading mechanism, the animation data are affected by large-capacity data during network transmission and calculation data loading, and the data communication and calculation delay time is long.

The problems caused by such low data utilization include the following: (1) the model data transmission, loading, and rendering process wastes a considerable amount of the network resources and computational capability; (2) The model data that do not match the client computing requirements cause the client's computing pressure to be too significant, resulting in a long waiting time for the user or even a service crash. Therefore, the process of optimization of the relationship between model-animation data and the specific needs of users reduces the redundancy of model data and the pressure of model data transmission, rendering and interaction through strategies to respond to user requirements.

According to the storage structure characteristics of the JD model, the theoretical basis and feasibility of the model animation split loading can be derived. In the JD mode data structure, the animation data exhibit attributed characteristics; that is, each independent track interactive animation has no logical relationship in time and is independent in the internal structure of the file. From the perspective of service abstraction, the animation of the model responds to related interactions in the form of subtracks. From the logical point of view of the service, model rendering construction and model animation interaction are two relatively independent processes. Thus, the model's rendering construction process does not need to rely on animation interactions. The animation interaction operation can perform further superposition operations based on the basic model rendering construction and the data that must be rendered and processed. Each animation datum can be independently applied to the model's rendering data without affecting the overall Web-based mobile 3D service process. In the process of loading the model data, loose coupling between the animation data and the model data can be used to separate the topology rendering operation of the model from the animation rendering operation of the model and reduce the amount of calculations of the model in the initial loading. At the same time, during the model loading process, the client caches the model data service in cooperation with the network data model construction. A multigranular 3D model animation loading method is applied to reduce the animation loading delay caused by network transmission [38].

C. OPTIMIZATION MODEL

To realize the procedural loading of model data, it is necessary to solve the related technical issues, such as asynchronous loading of model data of the WebGL 3D JavaScript libraries, separation of animation data, and independent loading of animation data. Therefore, the flow of the method adopted in this paper is as follows:

1. During the processing of the original model data, the server stores different animations and other attribute data separately in a form independent of the model construction data file.
2. Multigranularity decomposition is performed on the independent animation file according to the semantic description, and a multigranularity animation library is established.
3. Describe the interactive animation data based on the separation semantics. Additionally, the storage model of the interactive animation data establishes a mapping relation with the user request.
4. In the interactive application process, the server analyzes the user's interaction request, acquires the corresponding interaction data on demand, and returns to the client to perform a similar interactive response by optimizing the loading process, thereby reducing the initial model data carried when rendering.

According to the design, the main components of the system model are as follows: (1) *Establishment of a multigranularity animation cell library.* As shown by the multigranularity animation cell library in Fig. 2, the animation data in the 3D model are separated from the model data by an automated or manual calculation method, and the animation data are further decomposed to establish a multigranularity animation unit on the server side. Then, logical analysis and semantic annotation are performed for multigranularity units, and a multigranularity animation unit database for user

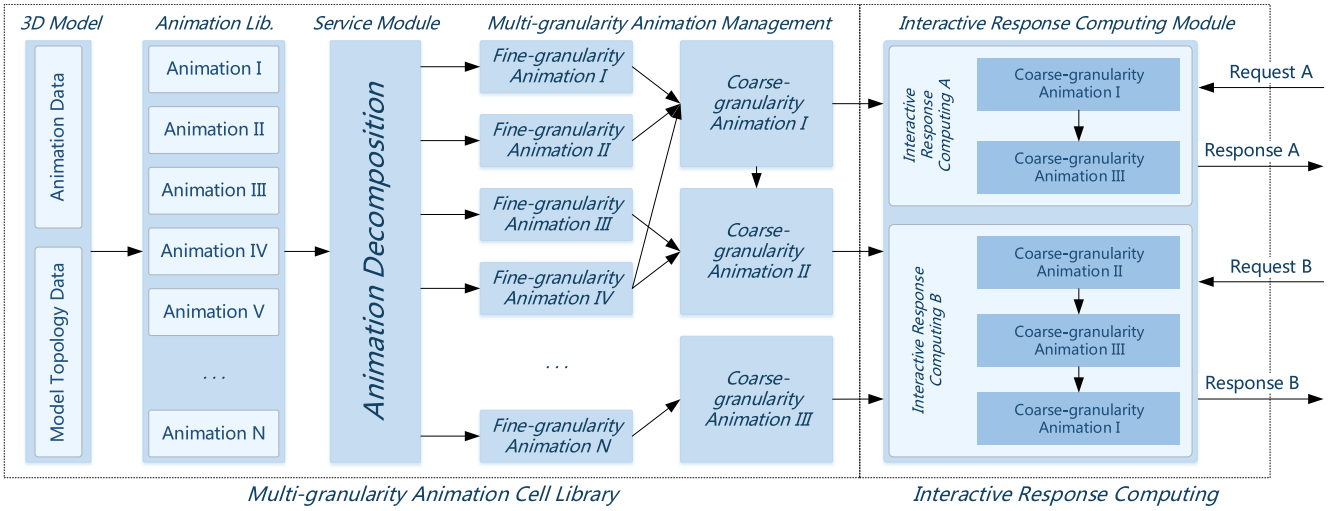


FIGURE 2. Server-side optimization model.

interaction requests is constructed to provide animated data service. (2) *Interactive response calculation.* As shown by the interactive response calculation in Fig. 2, an interactive response calculation module is constructed between the server-side multigranularity animation unit library and the network module. The interactive response calculation module converts the user's request data into a mapping and order relationship of the coarse-granularity animation unit module and provides streaming animation data service to the client according to the mapping and the order relationship. (3) *Collaborative animation data loading mechanism.* As shown in Fig. 3, when the client submits an interactive request, to reduce the delay caused by data communication between the client and the server, an animation data request and loading mechanism coordinated by the server and the client is adopted. When the interaction request is established, the coarse-granularity module number in the cache and the interaction request data form a request for data from the server. The server side calculates the coarse-granularity animation data necessary for the client by means of the interactive response calculation and provides the coordinated animation data service for the app with the coarse-granularity animation data in the cache. (4) *Model data asynchronous loading.* This service model increases the initial loading pressure of the Web-based mobile 3D rendering. Therefore, in the process of optimization, the data transmission process needs to be optimized at the application layer. The asynchronous transfer method is used for this situation: the model topology data and model animation data are asynchronously transferred to the mobile browser, and the model topology is rendered and animated via asynchronous loading and rendering operations.

IV. METHOD

The technical methods for the separation of animation data and multigranularity collaborative loading of animation data are adopted in the method design, and the technical solution is provided by the data service providing the mechanism, the asynchronous loading of the model data, and the on-demand rendering of the animation data to build a system model.

A. DESIGN OF THE PROVISIONING MECHANISM OF DATA DEVICES

The data service delivery mechanism is designed to transform the traditional file data service form into an interfaced data service form. For the preprocessed interaction model, the model data and the internal structure are serialized by the algorithm of tree structure traversal, and the model is transformed into a data object. By traversing the attributes of the object, acquiring the attribute set of the model data object and performing the calculation, the other attribute data of the model are integrated, and a new topology model is constructed and stored. The serialization operation is performed on the model animation data, and the animation data sequence object is constructed by using the independent animation data as elements [39].

After the animation data are separated from the model data, the independent animation data are decomposed into fine-granularity animation units with simple semantic meanings (such as words) by means of manual or intelligent algorithms. The fine-granularity animation units are further reorganized and arranged according to the semantic relationship of the animation, and a coarse-granularity animation cell oriented to the user request with complex semantic attributes (such as phrases) is constructed, as shown in the multigranularity animation cell library section of Fig. 2.

The individual objects in the multigranularity animation cell are traversed, the interactive semantic description is established, and a multigranularity animation library matching the service is constructed to provide a data interface for the interactive service matching process by means of the traversal process.

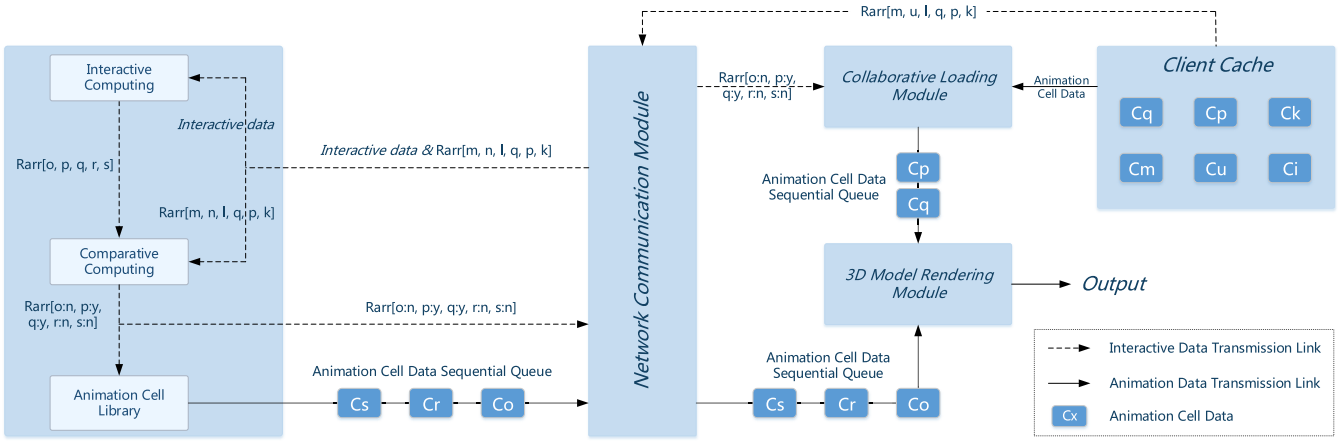


FIGURE 3. Animation collaborative loading process.

In the response of the multiservice response mechanism of the client, the interactive response analysis process is transferred to the server according to the service requirements. The interactive data are transformed into a mapping relationship related to the coarse-granularity animation unit. On the basis of the relationship between the application and the animation cell sequence, the server can generate an accurate response to the user's handover operation and establish an appropriate response mechanism, as shown in the interactive response calculation section of Fig. 2. The external data service provides services to clients with a unified data interface for client requests for the same function. When requesting data, the server-side interactive response calculation obtains the coarse-grained animation unit number of the mapping according to the interaction and compares the animation unit data in the cache to determine the return animation unit data state, as shown in Fig. 3. Then, the client takes the user's interaction description information as a parameter and converts the user's request data into an animation-related semantic description via the interaction request analysis module. The obtained semantic description query, calculation and matching corresponding animation data are returned to the browser in the form of JSON data for rendering and interactive operations.

B. MODEL DATA ASYNCHRONOUS COMMUNICATION

The browser's JavaScript engine is single-threaded, and synchronous loading of model data causes program blocking, which seriously affects the client user experience in the case of large data volumes.

Therefore, in the optimization of the on-demand loading process of model animation data, the WebGL 3D JavaScript library must be addressed via a synchronous-to-asynchronous conversion process applied to browser data requests, as shown in the model data asynchronous loading section of Fig. 3. First, based on the file loading interface corresponding to the model of three.js, the model loading method and rendering method are rewritten and encapsulated, and the FileLoader object is established. The FileLoader object mainly encapsulates the ajax asynchronous request method of JavaScript, and it monitors and processes related network request information to construct an asynchronous data communication channel between the client browser and the server. At the same time, the received model data stream is passed to the conversion module for corresponding instantiation operations and finally rendered by the three.js program at the bottom of the WebGL to display the model.

After the current model is delivered, a new data connection can be initiated via asynchronous processing. The client requests new model topology data or animation data from the server and asynchronously processes the topology data rendering and the animation data interaction calculation of different models or the same model, thereby reducing the short-term calculation pressure of the client.

C. ANIMATION DATA COLLABORATIVE LOADING

To reduce the time delay caused by the data communication and rendering process, based on the multigranularity model animation data service and asynchronous data loading method, a collaborative communication loading method for animation unit data is constructed between the client cache and the server. The method is implemented as follows.

- When the client starts the interaction request, the program driver network communication module traverses the client cache to obtain the queue number array of the coarse-grained animation module temporarily stored in the current cache. When the request is sent to the server, the interactive request data and cached animation unit data are used as request data and passed to the server, as shown in Fig. 3.
- The server-side interactive response calculation obtains the mapped coarse-grained animation unit number, compares the cached animation unit array in the interactive request, and determines the response data status of the animation unit in the application-related response data queue, as shown in Fig. 2.

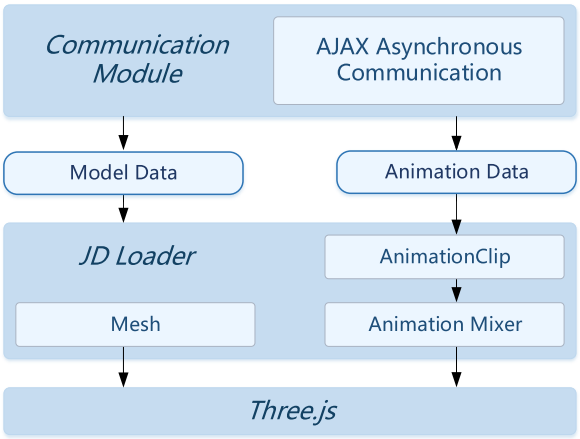


FIGURE 4. Optimized structural of animation loading.

- In the response time sequence and data construction of animation units, the orderly queue of application-related animation units obtained via interactive computing is traversed, and the data status of each animation unit in the state of animation queue is calculated by comparing the cache animation unit array. The priority is then returned to the requesting client.
- According to the sequence of the animation unit queue, the animation units that are not in the client cache are returned to the client's network communication module continuously.
- The 3D model rendering module of the client obtains the corresponding animation unit data from the network communication module or the collaborative loading module according to the returned application-related animation unit queue and the status of the animation unit. Then, the rendering calculation is performed step by step to complete the loading process of the collaborative animation unit data.

D. ANIMATION RENDERING ON DEMAND

The on-demand animation rendering is based on the method of asynchronous loading and rendering calculation after the client browser receives the animation data of the asynchronous server response. The user submits the interactive request through the mobile browser operation; the server responds to the animation data via the interface, including the information of each keyframe track and the name and duration of the animation; and the browser writes the data into the cache of the browser according to the corresponding service requirements. Asynchronous rendering calculations are then performed. The animation loading rendering process, which is independent of the model topology data, mainly includes three aspects, animation data combination, animation data fusion, and playback control, as shown in Fig. 4.

First, animation data combination:After receiving the animation data, the method in the loader parses the obtained animation data to generate an AnimationClip object. In the specific implementation method, the animation data acquired in the memory stack of the browser are traversed, the keyframe orbit data, including the position, the zoom, and the angle are acquired, and the obtained animation, name, time, etc., are combined. A reusable keyframe trackset AnimationClip object that matches the current animation data is generated as an action object that matches the animation to the model. The related code is shown in Algorithm 1.

Algorithm 1 Action Objects Matching

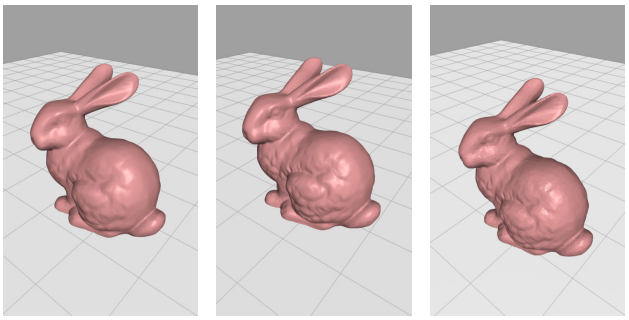
Require:
Animation nodes: $A = \{A_1, A_2, \ldots, A_n\}$
Ensure:
AnimationClip
for each animation node $A_i \in A$ **do**
/* Store the coordinate position of the animation */
$T_{i_1} \leftarrow VectorKeyFrame(A_i,\ pos)$
/* Store the scale of the animation */
$T_{i_2} \leftarrow VectorKeyFrame(A_i,\ scl)$
/* Store the quaternion of the animation */
$T_{i_3} \leftarrow QuaternionVectorKeyFrame(A_i,\ rot)$
end for
/* Mix all tracks to creat AnimationClip */
return $AnimationClip(name,\ length,\ T)$

Second, animation and model data fusion: In the combination of the animation data and the model topology data, an animation mixer object is constructed for each independent mesh object in the model topology data by rewriting the animation rendering method in the loader to implement a specific object in a specific scene. At the same time, the storage of model data is optimized from the bottom of WebGL, which reduces the cache resources consumed by the animation mixer and reduces the buffer pressure of the browser. The underlying WebGL uses a special data type Float32Array instead of a traditional array. A Float32Array is used to store binary data in JavaScript. It can be read more quickly and uses only half of the original memory. It is used in high-frame-rate model animation rendering scenarios. Storing model data will greatly improve the performance of the browser.

Third, animation playback control: After the animation data are parsed, the AnimationClip object is layered by the animation action's clip-action method. The browser generates an AnimationAction object and stores it in the animation blend of the corresponding mesh object. The AnimationAction object is used to dispatch AnimationClip animations, which can perform animations, pause animations, set animation execution times, execution modes, etc., to complete the corresponding service process.

V. OPTIMIZATION EFFICIENCY EVALUATION

The performance evaluation aims to consider the performance improvement achieved by optimized loading and traditional loading in different computing environments and different network environments in the process of the model loading and rendering calculation in ThreeJS. Meanwhile, the



(a) 10000 Faces (b) 50000 Faces (c) 100000 Faces

FIGURE 5. The loading results of different models.

TABLE 1. The optimized efficiency in model loading.

Model	File Size (KB)	Animation Size (%)	others (%)
Model I	1480	39%	61%
Model II	3948	19%	81%
Model III	11865	0.45%	99.55%

TABLE 2. The meaning of parameters in this section.

Parameter	Meaning
τ_{or}	The loading and rendering delay of the original method
τ_{op}	The loading and rendering delay of the optimized method
Δ_{op}	The delay with optimization methods saved ($\Delta_{op} = \tau_{or} - \tau_{op}$)
δ_{op}	The optimized efficiency of delay ($\delta_{op} = \Delta_{op}/\tau_{op}$)

evaluation verified the impact of animation loading on the user experience during the on-demand loading process and the effect of cache collaborative loading on user interaction. Finally determine the optimization efficiency of the scheme.

The information of the experimental model objects used in the experiment is shown in Fig. 5 and Table 1.

In this article, the application of Google Chrome DevTools is used to simulate the user's various network environment bandwidths and computing environment CPU computing capabilities under actual scenarios. The test environment including: computer architecture—ACPI x64; processor—Intel (R) Core (TM) i7-8750H, 2.21Ghz; operating system—Windows 10 Home, 64-bit; graphics processing unit—GeForce GTX 1060; browser—Google Chrome version 79.0.3945.130, 64-bit; test tool—Google Chrome DevTools. The parameters used are shown in Table2.

A. OPTIMIZATION EFFICIENCY EVALUATION WITH DIFFERENT MODEL

The effect of model loading optimization is assessed in the experimental environment. During the test, the time efficiency of the three models in carrying the animation and in the initial loading after optimization is determined. The model composition is presented in Fig. 5.

For service construction, a model containing a single animation is used as a reference for testing. In the experimental

TABLE 3. The optimized efficiency in model loading.

File	τ_{or} (ms)	τ_{op} (ms)	Δ_{op} (ms)	δ_{op}
Model I	510.3	294.9	215.4	42.21%
Model II	651	494.5	156.5	24.04%
Model III	917.4	844.7	72.7	7.92%

TABLE 4. The delay with different quantities of animation.

# Animation	τ_{or} (ms)	τ_{op} (ms)	Δ_{op} (ms)	δ_{op}
0	1278	–	–	–
1	1380	1278	102	7.41%
2	1451	1278	173	11.95%
3	1536	1278	258	16.80%
4	1617	1278	339	20.98%
5	1697	1278	419	24.69%
6	1783	1278	505	28.34%

environment,The network bandwidth is set to 10 Mbps, and the CPU frequency is set to 2.21 Ghz. By means of the test data, we obtain the phenomena reported in Table 3 in the process of building the service environment for the first time when the terminal situation is fixed.

According to the data in Table 3, in the process of separating model construction data from model animation data for on-demand loading method, the initial loading time is decreased to a certain extent. The average savings for the three models is 24.72% in experiment environment. The process of downloading and rendering the model animation data is moved back to the interaction process between the user and the application. At the same time, as the proportion of animation data in the overall model data increases, the rendering delay efficiency of the model is significantly improved under the optimization method.

B. MODEL WITH MULTIPLE ANIMATIONS

This subsection studies the impact on the initial loading delayof on-demand loading method for a model with multiple animations. Taking model I as a reference for testing, this subsection analyzes the corresponding effects on the initial loading of the service and interactive operations under the premise of increasing animation data. For service construction, the model loads 0 to 6 animations of the same size and test the loading delay. We compares the relationship between the original loading method and the on-demand loading method in the initial loading time and the optimization efficiency of the on-demand loading method according to the data in Table 4.

It shows that the original loading method increases the delay as the number of carried animations increases according to the data in Table 4. The on-demand loading method maintains a constant loading delay because it does not load animation. The animation data will be loaded and rendered while users interact (The effect of animation loading delay on user interaction experience is analyzed in SubSection V-D). Therefore, as the number of animation tracks increases,

TABLE 5. The delay with different CPU slowdowns.

CPU slowDown	τ_{or} (ms)	τ_{op} (ms)	Δ_{op} (ms)	δ_{op}
1×	532.8	286.1	246.7	46.30%
2×	1617.2	552.9	1064.3	65.81%
3×	2576.5	755.6	1820.9	70.67%
4×	3606.6	1076.4	2530.2	70.15%
5×	4410.9	1386.7	3024.2	68.56%
6×	5286.1	1688.1	3598	68.07%
7×	5997.5	2161.7	3835.8	63.96%
8×	6897	2524.9	4372.1	63.39%
9×	8040.1	3087.5	4952.6	61.60%
10×	9062.1	3484.9	5577.2	61.54%

TABLE 6. The delay with different network environments.

Download Spped	τ_{or} (ms)	τ_{op} (ms)	Δ_{op} (ms)	δ_{op}
50000 kbps	784.4	462	322.9	41.17%
10000 kbps	2220.3	1335	885.7	39.89%
5000 kbps	4108.2	2479	1629	39.65%
1000 kbps	19510.4	11928	7582.9	38.87%

the on-demand loading method optimizes the initial loading delay decreases.

C. INFLUENCE OF CLIENT COMPUTATION CAPABILITY AND NETWORK CAPABILITY ON OPTIMIZATION EFFICIENCY

The computation and network capability of the client are the main factors affecting the latency of mobile web 3D application loading. This subsection tests the optimize efficiency of on-demand loading method in different computing and network evironment. We use Google Chrome DevTools to simulate different computation and network environments.

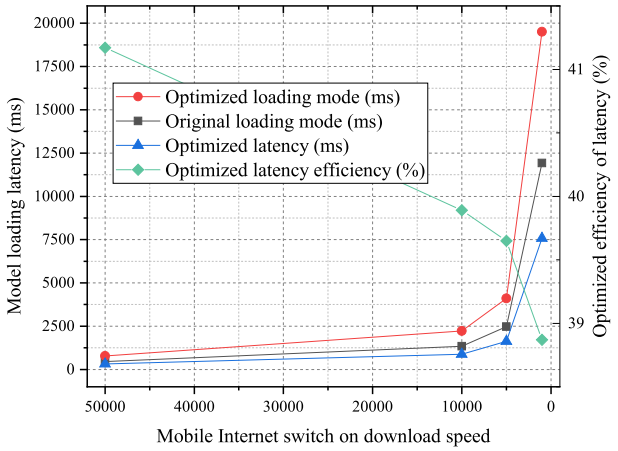
Based on the test of the CPU computing simulation deceleration of browser development tools, the impact on the loading efficiency improvement is analyzed under various circumstances, as reported in Table 5.

It indicates that the loading and rendering delay of mobile Web 3D applications increases with the decrease in CPU computing frenquecy according the data in Table 5. We can see that the on-demand loading method saves more loading and rendering time (Δ_{op}) with the CPU's computing frenquecy decreases. The on-demand loading method also shows an increasing trend in optimization efficiency (δ_{op}) with the CPU's computing frenquecy decreases.

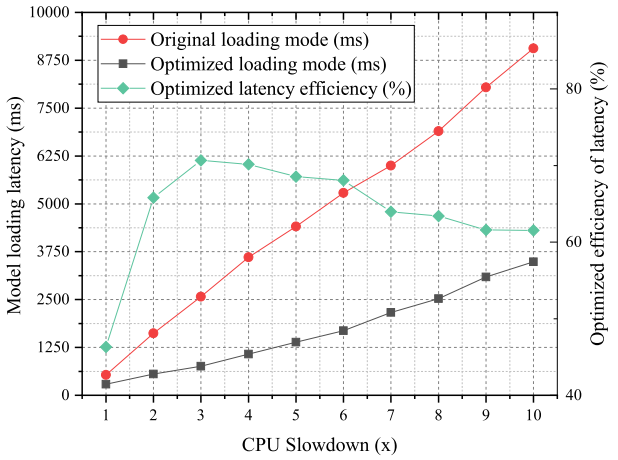
On the basis of the browser simulation in different network resource environments, the relationship between the loading delay and the network bandwidth is analyzed (see Table 6).

According to the data of Table 6, it indicates that the on-demand loading method saves more loading and rendering time and it shows an increasing trend with the bandwidth of mobile Access networkdeceases.

The expriments in this subsection indicates that the worse the environment (network and computing environment) of the mobile Web 3D application, the more loading time can be saved and the more optimization efficiency can be improved by the on-demand loading method as shown in Fig. 6. Therefore, the on-demand loading method is more obvious for mobile web 3D applications in non-ideal environments.



(a) Optimized efficiency of delay with different computing environment.



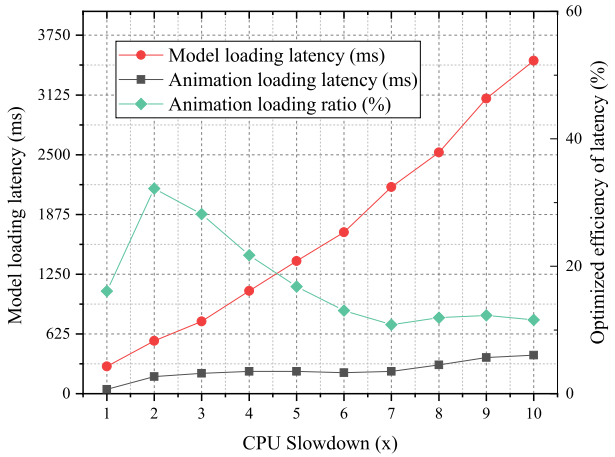
(b) Optimized efficiency of delay with different network bandwidth.

FIGURE 6. Influence of client computation capability and network capability.

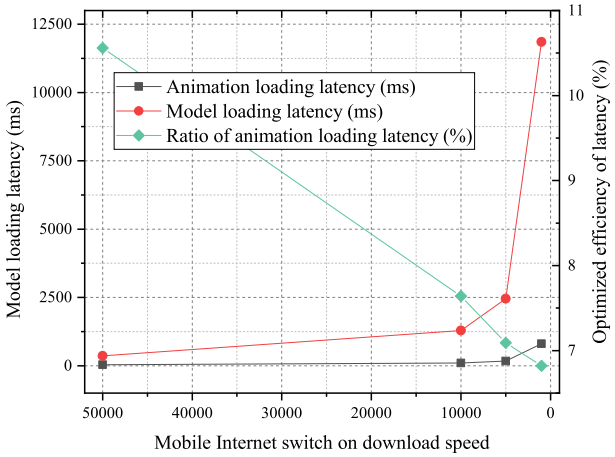
D. INFLUENCE OF ANIMATION LOADINNG DELAY

This subsection verificates the animation loading delay and its impact in the on-demand loading method. In the experimental design, we test separately the delay of loading with same animation in different computation and network environments and different file size animations in the same computation and network environment. The experiment uses *Model I* as a reference for testing. The delay of animation loading is shown in Fig. 7.

Fig. 7(a) and Fig. 7(b) show that the delay of animation loading is related to the computing and network environment of mobile devices. It indicates that the delay time of the animation loading is the same trend as that of the model loading delay comparing with Fig. 6. The ratio of animation loading delay to model loading delay varies. In Fig. 7(a), with the network bandwidth decreases, the ratio of animation



(a) Delay of animation loading in different bandwidth.



(b) Delay of animation loading in different CPU cycle.

FIGURE 7. The delay of animation loading.

loading delay to model loading delay decreases. Meanwhile, in Fig. 7(b), with computing capability decreases, the ratio of animation loading delay to model loading delay is also decreasing. This indicates that in the case of network transmission and computing capability lack, animation loading in the on-demand loading method has little impact on the user's interaction experience.

E. INFLUENCE OF THE MULTIGRANULAR ANIMATION LOADING MECHANISM

For service construction, an animation is decomposed into six coarse-granular parts which size is equal (the coarse-grained part has a larger file size to avoid delay test errors). Coarse-grained data in the cache is written using asynchronous preloading. This subsection tests the loading latency of different numbers of coarse-granular animations that have been loaded into the cache. By means of the test data, we obtain the phenomena reported in Table 7 which the average rendering latency (ms) of the loaded coarse-granular parts at different positions in the sequence are measured respectively.

According the data in Table 7, it shows that the the animation rendering and loading time become shorter with the

TABLE 7. Rendering time with different animation granularuty.

Cache Loading [a]	0	1	2	3	4	5	6
Network Loading [b]	6	5	4	3	2	1	0
Rendering Time	520.9	434	357	283	200	111	67

[a] animation granularity units are loaded in the cache.
[b] animation granularity units are loaded over the network.

granularity of animation data in the cache increases. This indicate that when mobile web applications contain multiple animations and there are certain repetitive key frames between animations, the latency of network communication and loading data can be reduced through the cache loading mechanism. Moreover, the more keyframe data is repeated between multi-track animations, the higher the efficiency of animation loading. Therefore, the joint application of the on-demand loading method and the multi-granular collaborative animation loading method is suitable for a complex, multi-animation interactive mobile web 3D application.

VI. DISCUSSION

The performance evaluation demonstrates that the method for on-demand loading of animation data improves the quality of the user interaction experience by reducing the amount of data loaded initially. However, the number of animation tracks and the size of the independent animation data affect the optimization of this method. In the condition of this experimental network environment and computing configuration, this method is not particularly efficient for rendering optimization in some model cases: (1) For mobile Web 3D loading services with a large single animation capacity, the efficiency of multigranular collaborative on-demand loading is affected by the granular coupling relationship between animations. In the case where the capacity of independent track animation is large (such as greater than 5000 kb) and the granularity coupling coefficient of the animation between tracks is low, the multigranular collaborative on-demand loading mode has a poor optimization effect and may even cause more interactive response delays. (2) The optimization efficiency is not significant for a 3D model that contains only one animation track, and the animation data volume is small (e.g., less than 50 kb). However, the separation of model topology data and animation data causes new delays. Therefore, this optimization method is suitable for 3D models with multiple animation tracks and a single animation data volume between 50 and 5000 kb. At the same time, non-open source 3D model formats (such as FBX) cannot directly adopt this method, and the file format needs to be converted into JD format.

VII. CONCLUSION

This paper proposes a method to separate animation data from 3D models and load them on demand for rendering based on multigranularity collaboration between a server and browser. In the process of implementation, the mesh model is first converted into a JD data format, and the animation data are

separated from the model topology data and stored according to the animation track. Further, the 3D model animation data sequence is semantically decomposed, and a multigranular model animation data service is established to provide continuous animation data support. On the basis of the above work, this paper further optimizes the loading and rendering methods of Three.js and aims to achieve asynchronous data loading and 3D model rendering on demand. The method realizes the rendering of a large and complex mobile Web 3D model by reducing the amount of data that are initially loaded and consummation loaded. Through the experimental data, we obtain the following conclusions: (1) The method of collaborative asynchronous data loading and on-demand rendering of 3D models can effectively shorten the initial loading time of the model. The improvement for model loading efficiency is particularly evident, especially in scenarios with complex interactions, in which the model carries multiple independent animations. (2) In a multigranular collaborative loading environment, the higher the semantic correlation between different trajectory animations is, the shorter the loading time of the 3D model animation and the greater the optimization efficiency. (3) The optimization has different effectiveness in different computing and network environments. Specifically, when the computation capabilities and network bandwidth are gradually reduced, the optimization efficiency of rendering is more prominent.

Given the conclusions obtained above, in subsequent research, we will focus mainly on preloading large-data-volume animations using context-aware methods and a collaborative rendering method based on mobile edge servers and cloud servers.

REFERENCES

[1] A. Syberfeldt, O. Danielsson, and P. Gustavsson, "Augmented reality smart glasses in the smart factory: Product evaluation guidelines and review of available products," *IEEE Access*, vol. 5, pp. 9118–9130, 2017.

[2] X. Qiao, P. Ren, S. Dustdar, and J. Chen, "A new era for Web AR with mobile edge computing," *IEEE Internet Comput.*, vol. 22, no. 4, pp. 46–55, Jul. 2018.

[3] M. Makar, V. Chandrasekhar, S. S. Tsai, D. Chen, and B. Girod, "Interframe coding of feature descriptors for mobile augmented reality," *IEEE Trans. Image Process.*, vol. 23, no. 8, pp. 3352–3367, Aug. 2014.

[4] T. Engelke, M. Becker, H. Wuest, J. Keil, and A. Kuijper, "MobileAR browser—A generic architecture for rapid AR-multi-level development," *Expert Syst. Appl.*, vol. 40, no. 7, pp. 2704–2714, Jun. 2013.

[5] D. Chatzopoulos, C. Bermejo, Z. Huang, and P. Hui, "Mobile augmented reality survey: From where we are to where we go," *IEEE Access*, vol. 5, pp. 6917–6950, 2017.

[6] C. Marrin. *Webgl Specification [EB/OL]*. Accessed: Apr. 4, 2020. [Online]. Available: http://www.khronis.org/registry/webgl/

[7] M. Baker and J. Manweiler, "Views of current and future technology," *IEEE Pervas. Comput.*, vol. 16, no. 2, pp. 9–13, Apr. 2017.

[8] E. G. Larsson, O. Edfors, F. Tufvesson, and T. L. Marzetta, "Massive MIMO for next generation wireless systems," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 186–195, Feb. 2014.

[9] H. Hoppe, "Progressive meshes," in *Proc. 23rd Annu. Conf. Comput. Graph. Interact. Techn.*, 1996, pp. 99–108.

[10] R. Pajarola and J. Rossignac, "Compressed progressive meshes," *IEEE Trans. Vis. Comput. Graphics*, vol. 6, no. 1, pp. 79–93, Jan./Mar. 2002.

[11] A. Maglo, I. Grimstead, and C. Hudelot, "POMAR: Compression of progressive oriented meshes accessible randomly," *Comput. Graph.*, vol. 37, no. 6, pp. 743–752, Oct. 2013.

[12] C. Kangying, J. Wenfei, L. Tao, and T. Jiang, "Pattern-based 3D model compression," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2016, pp. 2872–2876.

[13] P. Fraga-Lamas, T. M. Fernández-Caramés, O. Blanco-Novoa, and M. A. Vilar-Montesinos, "A review on industrial augmented reality systems for the industry 4.0 shipyard," *IEEE Access*, vol. 6, pp. 13358–13375, 2018.

[14] M. Krivokuca, W. H. Abdulla, and B. C. Wunsche, "Sparse approximations of 3D mesh geometry using frames as overcomplete dictionaries," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops*, Dec. 2013, pp. 660–667.

[15] Y. Yan and L. Kunhui, "3D visual design for mobile search result on 3G mobile phone," in *Proc. Int. Conf. Intell. Comput. Technol. Automat.*, vol. 1, May 2010, pp. 12–16.

[16] A. Patney, M. Salvi, J. Kim, A. Kaplanyan, C. Wyman, N. Benty, D. Luebke, and A. Lefohn, "Towards foveated rendering for gaze-tracked virtual reality," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 1–12, Nov. 2016.

[17] N. C. Zakas, "The evolution of Web development for mobile devices," *Commun. ACM*, vol. 56, no. 4, pp. 42–48, Apr. 2013.

[18] K. Wang, G. Lavoué, F. Denis, and A. Baskurt, "A comprehensive survey on three-dimensional mesh watermarking," *IEEE Trans. Multimedia*, vol. 10, no. 8, pp. 1513–1527, Dec. 2008.

[19] L. Wen, N. Xie, and J. Jia, "Fast accessing Web3D contents using lightweight progressive meshes," *Comput. Animation Virtual Worlds*, vol. 27, no. 5, pp. 466–483, Sep. 2016.

[20] A. Ynnerman, T. Rydell, D. Antoine, D. Hughes, A. Persson, and P. Ljung, "Interactive visualization of 3D scanned mummies at public venues," *Commun. ACM*, vol. 59, no. 12, pp. 72–81, Dec. 2016.

[21] M. Limper, M. Thöner, J. Behr, and D. W. Fellner, "SRC—A streamable format for generalized web-based 3D data transmission," in *Proc. 19th Int. ACM Conf. 3D Web Technol.*, 2014, pp. 35–43.

[22] A. S. Lalos, I. Nikolas, E. Vlachos, and K. Moustakas, "Compressed sensing for efficient encoding of dense 3D meshes using model-based Bayesian learning," *IEEE Trans. Multimedia*, vol. 19, no. 1, pp. 41–53, Jan. 2017.

[23] F. Liu, P. Shu, H. Jin, L. Ding, J. Yu, D. Niu, and B. Li, "Gearing resource-poor mobile devices with powerful clouds: Architectures, challenges, and applications," *IEEE Wireless Commun.*, vol. 20, no. 3, pp. 14–22, Jun. 2013.

[24] H. Flores and S. N. Srirama, "Mobile cloud middleware," *J. Syst. Softw.*, vol. 92, pp. 82–94, Jun. 2014.

[25] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proc. IEEE Int. Conf. Comput. Commun.*, Mar. 2012, pp. 945–953.

[26] C. Shi, K. Habak, P. Pandurangan, M. Ammar, M. Naik, and E. Zegura, "COSMOS: Computation offloading as a service for mobile devices," in *Proc. 15th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2014, pp. 287–296.

[27] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: Elastic execution between mobile device and cloud," in *Proc. 6th Conf. Comput. Syst.*, 2011, pp. 301–314.

[28] F. Xia, F. Ding, J. Li, X. Kong, L. T. Yang, and J. Ma, "Phone2Cloud: Exploiting computation offloading for energy saving on smartphones in mobile cloud computing," *Inf. Syst. Frontiers*, vol. 16, no. 1, pp. 95–111, Mar. 2014.

[29] M.-R. Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, and R. Govindan, "Odessa: Enabling interactive perception applications on mobile devices," in *Proc. 9th Int. Conf. Mobile Syst., Appl., Services*, 2011, pp. 43–56.

[30] H. R. Maamar, A. Boukerche, and E. Petriu, "Streaming 3D meshes over thin mobile devices," *IEEE Wireless Commun.*, vol. 20, no. 3, pp. 136–142, Jun. 2013.

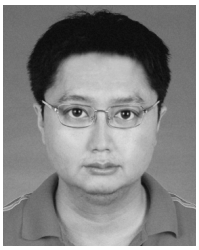
[31] A. Shatte, J. Holdsworth, and I. Lee, "Mobile augmented reality based context-aware library management system," *Expert Syst. Appl.*, vol. 41, no. 5, pp. 2174–2185, Apr. 2014.

[32] J. Ren, Y. He, G. Huang, G. Yu, Y. Cai, and Z. Zhang, "An edge-computing based architecture for mobile augmented reality," *IEEE Netw.*, vol. 33, no. 4, pp. 162–169, Jul. 2019.

[33] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 5031–5044, May 2019.

[34] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges," *IEEE Commun. Mag.*, vol. 55, no. 4, pp. 54–61, Apr. 2017.

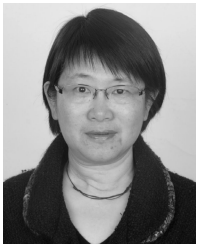
[35] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.
[36] X. Masip-Bruin, E. Marín-Tordera, G. Tashakor, A. Jukan, and G.-J. Ren, "Foggy clouds and cloudy fogs: A real need for coordinated management of fog-to-cloud computing systems," *IEEE Wireless Commun.*, vol. 23, no. 5, pp. 120–128, Oct. 2016.
[37] H.-J. Jeong, C. H. Shin, K. Y. Shin, H.-J. Lee, and S.-M. Moon, "Seamless offloading of Web app computations from mobile device to edge clouds via HTML5 Web worker migration," in *Proc. ACM Symp. Cloud Comput.*, Nov. 2019, pp. 38–49.
[38] Y. Huang, X. Qiao, P. Ren, L. Liu, C. Pu, and J. Chen, "A lightweight collaborative recognition system with binary convolutional neural network for mobile Web augmented reality," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019, pp. 1497–1506.
[39] K. Y. Lam, L. H. Lee, T. Braud, and P. Hui, "M2A: A framework for visualizing information from mobile Web to mobile augmented reality," in *Proc. IEEE Int. Conf. Pervas. Comput. Commun. (PerCom)*, Mar. 2019, pp. 1–10.



LIANG LI is currently a Senior Experimentalist with the School of Electronics and Information, Communication University of Zhejiang (CUZ), Hangzhou, China.

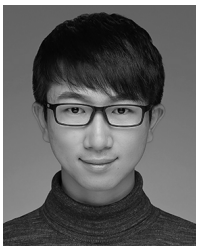
His research interests lie in augmented reality, virtual reality, services computing, computer vision, and 5G networks.

XIUQUAN QIAO is currently a Full Professor with the Beijing University of Posts and Telecommunications and the Deputy Director of the Network Service Foundation Research Center, State Key Laboratory of Networking and Switching Technology. He has authored or coauthored more than 60 technical articles in international journals and conferences, including the *IEEE Communications Magazine*, *Computer Networks*, the IEEE Internet Computing, the IEEE Transactions on Automation Science and Engineering, and *ACM SIGCOMM Computer Communication Review*. His main research interests lie in future Internet, services computing, computer vision, augmented reality, virtual reality, and 5G networks. He is a winner of the 2008 Beijing Nova Program and a recipient of the First Prize of the 13th Beijing Youth Outstanding Science and Technology Paper Award, in 2016.



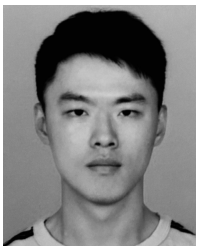
QIONG LU is currently an Associate Professor with the School of Electronic and Information, Communication University of Zhejiang (CUZ), Hangzhou, China.

Her research interests lie in media content management and computer control.



PEI REN is currently pursuing the Ph.D. degree with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China.

He is also a Visiting Scholar with the School of Computer Science, Georgia Institute of Technology, funded by the China Scholarship Council. His current research interests include the services computing, augmented reality, edge computing, future Internet architecture, and 5G networks.



RUIBIN LIN is currently pursuing the bachelor's degree with the Zhejiang University of Media and Communications. His research interests are Web front-end technology, WebGL, augmented reality, and virtual reality.