International Journal of Informatics and Communication Technology (IJ-ICT)
Vol. 13, No. 1, April 2024, pp. 57∼66
ISSN: 2252-8776, DOI: 10.11591/ijict.v13i1.pp57-66

Remote practical instruction using web browsers

Nagaki Kentarou, Fujita Satoshi
Department of Information Science, School of Informatics and Data Science, Hiroshima University, Hiroshima, Japan

Article Info

Article history:

Received Apr 28, 2023
Revised Nov 6, 2023
Accepted Nov 11, 2023

Keywords:

Firebase
Interactive web application
Remote practical instruction
Three.js
Vue.js

ABSTRACT

This paper introduces a novel approach to remote coaching, specifically targeting the body movements of learners participating remotely. The proposed system employs a smartphone camera to capture the learner's body and represent it as a 3D avatar. The instructor can then offer guidance and instruction by manipulating the 3D avatar's shape, which is displayed on a web browser. The main challenge faced by the system is to enable the sharing and editing of 3D objects among users. Since the HTML5 drag-and-drop feature is inadequate for transforming virtual objects consisting of multiple interconnected rigid bodies, the system tracks the pivot point of the manipulated rigid body. It assigns attributes such as pivot points and action points to each object, extending beyond their 2D screen coordinates. To implement the system, an interactive web application framework following the model-view-view-model (MVVM) architecture is utilized, incorporating Vue.js, Three.js, and Google Firebase. The prototype system takes advantage of the data binding capability of the framework and successfully operates within the 3D space of a web browser. Experimental results demonstrate that it can effectively share transformation information with an average delay of 300 ms.

Corresponding Author:

Fujita Satoshi
Department of Information Science, School of Informatics and Data Science, Hiroshima University
Kagamiyama 1-4-1, Higashi-Hiroshima 739-8527, Japan
Email: fujita@hiroshima-u.ac.jp

1. INTRODUCTION

The onset of the pandemic in early 2020 has profoundly transformed our daily routines. With the realization that the virus primarily spreads through close personal contact, traditional face-to-face gatherings in confined spaces were deemed unsafe. As a result, the prevalence of remote work and online classes has persisted, even years later. When it came to transitioning classes to an online format, it became evident that the requirements differed between subjects like mathematics and language, and practical courses that involved physical movements. Many universities have already adopted learning management systems (LMS) for effective course management, and the combination of LMS and live video streaming has proven useful for delivering classroom lectures remotely [1]-[3]. However, for practical courses such as laboratory experiments and childcare practice, additional efforts were necessary to achieve the same educational outcomes as in traditional in-person teaching. Various solutions have been developed, including systems with multiple camera angles to teach surgical skills [4] and virtual reality environments to facilitate the acquisition of appropriate interpersonal behavior [5]-[9]. However, these systems often entail significant costs, making their implementation in elementary and junior high schools across the country unrealistic.

This paper investigates the development of a real-time, remote instruction system for body movements utilizing cutting-edge web technologies. The advent of HTML5 [10], [11], which received its recommendation

in 2014, has significantly enhanced the capabilities of web applications on widely-used browsers like Google Chrome and Mozilla Firefox. Notably, HTML5's video tag enables video playback without the need for plug-ins like Flash, while the recent introduction of the WebXR device application programming interface (API) [12] allows for bidirectional communication between browsers and extended reality (XR) devices. The primary focus of this paper is the model-view-view-model (MVVM) web architecture [13], employed by contemporary frameworks such as Vue.js, to construct interactive applications. Additionally, WebGL's capacity to render virtual 3D environments in a browser is explored. Motion information from a user can be shared with remote learners via the internet and the public cloud. In our prototype system, Firebase from the Google Cloud Platform is utilized for this purpose. However, in theory, it is feasible to establish regional local noSQL servers like GraphQL [14], [15] that support pub/sub capability, as it simply requires a JavaScript object notation (JSON) database. In summary, we examine the scenario of remote instruction for body movements and explore how web technologies can be leveraged to achieve real-time remote instruction. The authors have implemented this functionality as a single-page application and conducted performance evaluations, including response time, considering factors such as the complexity of shared 3D objects.

The remainder of this paper is organized as follows. Section 2 gives an overview of background technologies. After describing the setup process of the proposed system in section 3, section 4 describes how the modifications made by the teacher to the avatar's behavior are reflected on the remote learner's screen. Section 5 provides an overview of the experiments and the results. Section 6 reviews the latest related work, and section 7 concludes the paper with future issues.

2. BACKGROUND TECHLONOGIES

The objective of the proposed system is to facilitate remote instruction of practical skills through widely used web browsers like Google Chrome. The system enables a coach to manipulate the posture and movements of the learner's avatar, which is displayed in real-time on the learner's browser. To develop this system, Vue.js and Three.js technologies were employed. This section offers a high-level summary of these technologies.

2.1. Model-view-view-model and Vue.js

In the early days of the world wide web, a simple software architecture called model-view-controller (MVC) gained prominence in web application design. Ruby on Rails, a web application framework introduced in 2004, embraced this philosophy and enabled the rapid development of MVC-based web applications. Following the success of Ruby on Rails, similar frameworks emerged and became widely adopted, such as Django for Python, Spring framework for Java, and CakePHP and Laravel for PHP.

With the increasing demand for interactive web applications, the architecture of these applications has garnered significant attention in recent years. MVVM has emerged as a successor to MVC [13], originating from Microsoft's user interface (UI) subsystems such as Windows Presentation Foundation (WPF) and Silverlight. In the MVVM model, the presentation logic (input/output (I/O) functionality of UI applications) is separated from the model and divided into the declaratively described view and the ViewModel, which maps the state of the View to the Model. Unlike MVC, where the View maintains its own state as an object, MVVM stores the View's state in the ViewModel. Notably, MVVM encompasses the concept of data binding. In this architecture, the View directly sends user requests to the ViewModel, which then instructs the Model and automatically updates the View through data binding. Although initially employed in desktop applications like WPF, MVVM has been embraced by modern web application frameworks such as AngularJS (now renamed Angular) and Vue.js.

HTML5 [11] introduces a new feature called drag-and-drop, allowing virtual objects like rectangles to be easily moved and rotated within a web browser. By setting the object's draggable attribute to "true" in the HTML file and defining the drag event in JavaScript, basic object transformations can be achieved. However, for more intricate transformations, such as altering the shape of an object composed of multiple rigid bodies interconnected by joints, relying solely on the drag-and-drop API becomes insufficient. To accomplish these changes, it becomes necessary to track the pivot point of the manipulated rigid body. Additionally, the real-time calculation of the applied force direction in the 3D virtual space relies on a continuous stream of events from the view component, with the results being reflected in both the model and the view. In essence, each object necessitates a range of attributes, including the fulcrum and action points, in addition to its 2D coordinates.

2.2. Three.js

Three.js is a JavaScript library that facilitates the rendering of virtual 3D environments within a web browser. It serves as a wrapper for the WebGL API. One of the key features of Three.js is its grouping functionality, which enables the management of multiple virtual 3D objects together. This grouping capability allows objects A and B to be nested, resulting in rotation operations on A being reflected in B. Additionally, Three.js offers various animation systems, including the Keyframe animation-system utilized in the proposed system. Keyframe animation is an animation technique that involves defining keyframes at specific points in time and interpolating frames between them.

The Keyframe animation system consists of four fundamental components: AnimationClip, KeyframeTrack, AnimationMixer, and AnimationAction. The details of each component are explained as follows: i) AnimationClip: the AnimationClip serves as a container for storing motion data related to a virtual 3D object. For instance, if the object being animated is an avatar, separate AnimationClips can be created for actions like walking, jumping, or sidestepping; ii) KeyframeTrack: the KeyframeTrack acts as a track within an AnimationClip, holding specific animation property data. For example, in a walking animation clip, one KeyframeTrack could store position change data for the lower right arm, another could store rotation change data for the same part, and a third could store rotation change data for the upper left arm. Each track contains time-value pairs representing attributes such as position and angle; iii) AnimationMixer: the AnimationMixer controls the actual playback of animations and enables blending and merging of multiple animations. It facilitates simultaneous playback of different animations, allowing for seamless transitions and combinations; iv) AnimationAction: the AnimationAction complements the AnimationMixer and provides additional control over the playback of animations. It allows specifying when to play or stop a particular AnimationClip within one of the Mixers, determining whether the clip should be repeated, and specifying effects like fading or time scaling during playback. These components collectively contribute to the animation framework, with AnimationClip and KeyframeTrack providing animation data, AnimationMixer controlling the playback, and AnimationAction offering fine-grained control over individual clips within the Mixer.

3. INITIALIZATION OF THE PROPOSED APPLICATION

The configuration of the proposed system is depicted in Figure 1, providing an overview of its structure. It consists of two main components: the client-side and the server-side. On the server-side, Firebase from the Google cloud platform (GCP) is employed as the implementation platform. Meanwhile, the client-side comprises a program that runs within a web browser. To develop interactive web applications, Vue.js is utilized, while Three.js is employed to manage 3D graphics within the browser environment. Data storage and sharing among users are facilitated by Firebase. The system is designed as a single-page application using the MVVM architecture. User interactions with virtual objects are captured through document object model (DOM) elements, processed within the Vue instance, and subsequently displayed in the canvas element representing the 3D space. For a more comprehensive understanding of the implementation, further details are outlined in subsequent sections.

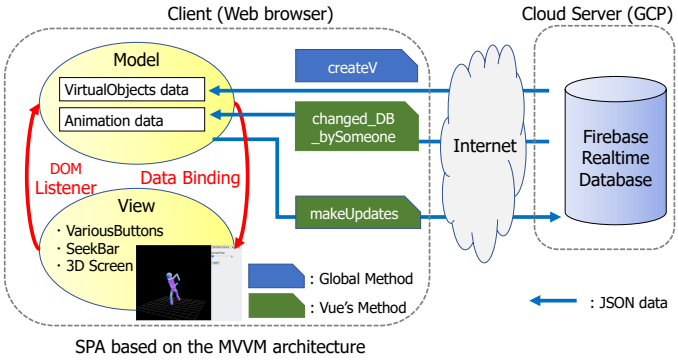


Figure 1. Configuration of the proposed system

3.1. Capturing body image into virtual space

Initially, the system captures a user's full-body image using their smartphone camera and displays it on the browser. This image is then used to create an avatar in a virtual space with the help of Three.js. To analyze the user's posture, TensorFlow.js and PoseNet, a posture estimation model, are employed. PoseNet requires a full-body image projected onto a DOM element, which is achieved using a free software called iriunWebCam. By accessing the smartphone camera and capturing the video output, the app displays the video on a DOM element. The streaming video information is obtained using the getUserMedia() function and assigned to the previously obtained DOM element using the srcObject() method.

The output from PoseNet consists of 17 key points that represent joint points in the human body. The relationship between these 17 key points and the corresponding joint points is illustrated in Figure 2. To create virtual 3D objects in Three.js, the system calculates the length of each body part based on the distance between the key points at the endpoints of that part. These lengths are then utilized to determine the width, height, and depth of the 3D objects, as indicated in Table 1. The size of each object is normalized, with the head size set to 1. Additionally, the positions of each group of body parts, such as the left arm consisting of the left forearm and left upper arm, are calculated after determining the size of each part. The resulting 3D object is stored in Firebase in JSON tree format, as shown in Figure 3. In this figure, the Object column stores the size of each avatar part and the position of each part group, while the Motion column contains information about the movement of each part.

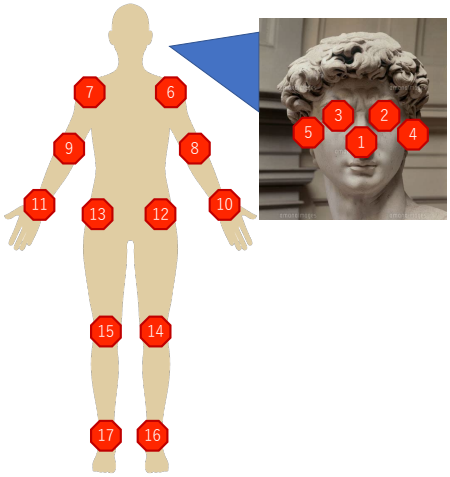


Figure 2. 17 key points contained in the output data of PoseNet and corresponding parts on the human body

Table 1. Definition of the length of each body part

Body part	Definition
Head	From the right ear (5) to the left ear (4)
Torso	From the midpoint of right shoulder (6) and left shoulder (7) to the midpoint of right temple (13) and left temple (12)
Right upper arm	From the right shoulder (7) to the right elbow (9)
Right forearm	From the right elbow (9) to the right hand (11)
Left upper arm	From the left shoulder (6) to the left elbow (8)
Left forearm	From the left elbow (8) to the left hand (10)
Right thigh	From the right temple (13) to the right knee (15)
Right lower leg	From the right knee (15) to the right ankle (17)
Left thigh	From the left temple (12) to the left knee (14)
Left lower leg	From the left knee (14) to the left ankle (16)

3.2. Creating Vue instance

When the editor app is launched, it follows a series of steps to display the initial screen in the browser. Firstly, the browser loads the HTML, CSS, and JavaScript files in that order. The loading of JS files is delayed until the DOM is created from the HTML because the DOM is required for creating Vue instances. During

the creation of the Vue instance, the data set located in the data property is initialized. This data set contains information related to the virtual space in the browser and the animation for the 3D objects. The specific details of this data set are summarized in Table 2.

The creation of a Vue instance involves a series of initialization processes, and the user can add their own code, called a lifecycle hook, to execute a function at a specific stage in this initialization process. In our system, we describe the process required for editing 3D objects in the browser by adding code to the mounted hook, which is executed immediately after the DOM element in the HTML file is mounted. Furthermore, a Vue instance can have data objects, and these data objects can have multiple properties, all of which can be added to the reactive system. When data added to the reactive system is updated, it triggers the re-rendering of the browser. Once the initialization is completed, the screen depicted in Figure 4 is displayed, and the setup of the 3D space and the construction of the animation system commence.

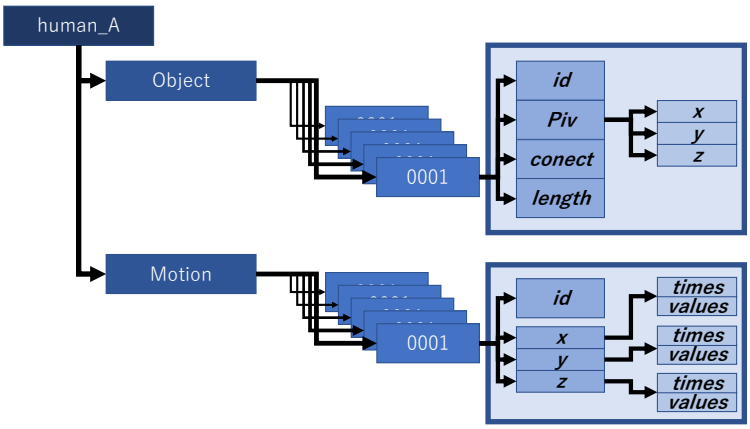


Figure 3. JSON tree structure stored in the database

Table 2. Description of data used in Vue instance

Data	Description
canvas, scene, renderer	Collection of data used to draw a 3D space
camera, controls	Collection of data used to control the viewpoint in the 3D space
light	Light source in the 3D space
bar_value	Holder of the current value of the frame seek bar
selected_parts_name, selected_parts	The former reads the name of the part being selected from the DOM, and the latter holds where the currently selected data is located in the object.
selected_parts_rotX(Y, Z)	When a time and a part are selected, rotation angles of the three axes for that part are retained based on the motion data stored in the DB.
rotationX_bar(Y, Z)	Holder of the current value of the angle bar.
human, human_clone	Holder of a complex object to be drawn in the 3D space. One is used to move the object during playback, and the other is used to move it in response to user's action.
keyframetracks, clips, mixers, actions	Collection of data used for animation based on the movement data.
reset_flag	Flag to determine if the animation playback status needs to be reset.
eventstart, eventmove, eventend	When setting up an event listener, it determines whether to assign mouse events or touch events, and keeps the event name.

3.3. Setting virtual 3D space

The virtual 3D space requires adjustments in terms of lighting, aspect ratio, and the creation of complex objects to facilitate touch screen manipulation. In our system, we make use of a grouping function to construct complex objects from simple ones. For instance, in the case of a humanoid model composed of multiple body parts connected by joints, each body part has its own shape and rotation center, while movement is applied to the entire body. By adding basic objects representing body parts, such as boxes and cylinders, to the body group using the add() method, we can effectively manage the rotation center of each part as a whole and accurately depict movement with human-like joints. Figure 5 provides a visual representation of the nested structure of a grouped object.

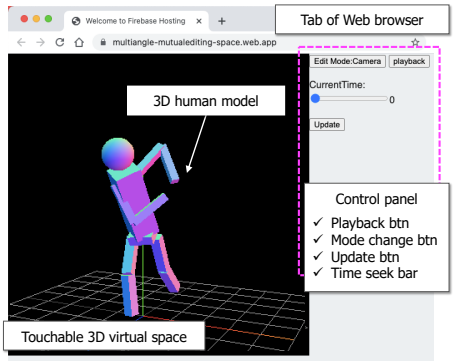


Figure 4. Screenshot after completing the initialization of the Vue instance, where a canvas is displayed on the left side of the window. On this canvas, we can edit the shape and the movement of an object with fingertips. An object has a structure consisting of multiple rigid bodies connected by joints

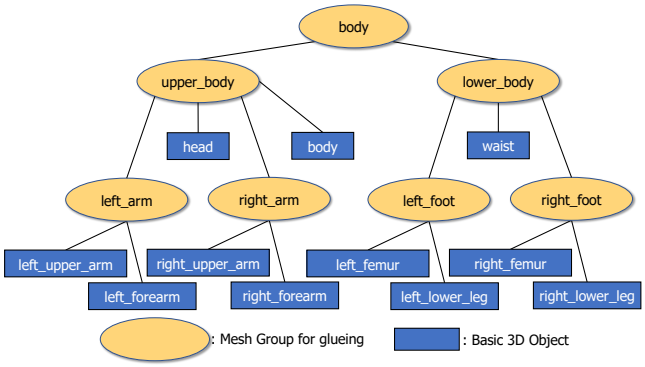


Figure 5. Nested structure of a complex object consisting of several basic 3D objects (blue rectangle represents 3D object and orange ellipse represents mesh group object used for the glueing of objects)

3.4. Setting Keyframe animation system

We employ the keyframe animation system of Three.js to animate 3D objects, which comprises four components. These components are stored as objects within the Vue instance and undergo the following initialization process: i) KeyframeTracks are created for each part of the object and for each axis in the 3D space; and ii) data from the database, which stores key-value pairs for each KeyframeTrack (with the value being an array type), is imported. This data is necessary for the animation. After creating all the KeyframeTracks, the remaining components are initialized to complete the setup. i) AnimationClip is generated by parsing an object with three properties: duration, name, and tracks. The duration represents the length of the animation, the name is used to distinguish between multiple AnimationClips, and tracks encompass all the KeyframeTracks associated with the animatio; ii) while it is customary to pass the original object to the AnimationMixer, in our implementation, we pass a clone of the object. This enables the clone to be used for playback in the browser, while the original object is retained for editing purposes; and iii) finally, the AnimationAction is created by invoking the clipAction() method in AnimationMixer, using the created clips as the argument. By following these steps, we establish the necessary components and configurations to animate 3D objects effectively.

4. MANUAL EDITING OF AVATAR MOVEMENTS

This section outlines the process of editing the pose and movements of the avatar displayed in the browser. The following steps are performed by the teacher to modify the avatar's pose and movements: i) identify the action that needs to be corrected in the animation. Set the seek bar to the start time of the action and touch the corresponding part of the avatar; ii) drag the touched part to adjust it to the desired pose or movement. Once the modification is made, drop the part in its new position; and iii) upon dropping the part, the modification is completed. The shared database is automatically updated to reflect the modification, and

the updated information is propagated to other clients. By following these steps, the teacher can effectively modify the pose and movements of the avatar in the browser, enabling precise adjustments to be made in the animation.

4.1. Select the action to bemodified

To modify an action, the first step is to adjust the value of the frame seek bar in the browser to match the start time of the action. This adjustment triggers a value change event, which is linked to the FrameSelect() function in the JavaScript file. The purpose of this function is to move the animation system to the specified time indicated by the bar's value (retrieved from the connected data property). Consequently, the avatar displayed on the screen will assume the pose corresponding to that specific time. When the user interacts with the virtual 3D space in the browser by touching it, the grapObject() function is executed. This function utilizes the raycaster() function from Three.js to determine which part of the avatar the mouse pointer is hovering over. It sets up two event listeners, one for dragging and one for dropping, and initiates the update() function. The drag event listener is activated only when the object detection successfully identifies a body part. Once created, it will be deleted after the object is dropped. The update() function is responsible for handling the dragging process. It runs continuously during the dragging phase and updates the database with the user's edits every 10 milliseconds. The result of the user's edits is stored in the updates variable. If the updates variable has a value, the function reflects its contents in the Motion column of the database. If the updates variable is empty, the function waits for 10 milliseconds before recursively calling the update() function again. By following this process, the user can effectively modify the actions of the avatar in the browser, with the changes being promptly stored in the database and reflected in the animation.

4.2. Modification of movements by dragging

During the dragging of a 3D object, the onDocumentMove() function is called to calculate the rotation value of the body part based on the mouse movement and record the operation's result. Let's consider the following editing operations: i) the displayed time on the frame bar is $t = 2$; ii) the right upper arm of the body is selected; and iii) the right upper arm is rotated by 1.57 radians in the x-axis direction. First, the system retrieves the motion data recorded for the x-axis of the right upper arm from the database and stores it in a local variable within the browser. It then checks the times array, which stores information about time transitions, and the values array, which stores information about rotation value transitions. If there is existing data for time $t = 2$ in the times array, the corresponding element in the values array is updated to reflect the rotation of 1.57 radians. If there is no data for $t = 2$, new data for $t = 2$ is added to both the times and values arrays. The updated values (and times) array is then added to the updates variable. The updates variable is utilized in the update function to update the columns related to the right upper arm in the database. It is important to note that this process is continuously performed during the dragging, resulting in constant updates to the updates variable.

4.3. Update of shared database

When the motion editing is completed, the dragged object is dropped, and the onDocumentUp() function is executed to reset the variables used in the editing process. This includes the following steps: i) removing the two event listeners that were set during object detection; ii) resetting the variables that hold the data of the selected body part and the variables used in the update process (such as updates); and iii) setting a flag to stop repeating updates. If the modification results in a database update, the updates from the Motion column are retrieved and distributed to all users through Firebase. Each client then updates its local data based on the received updates and displays the result on its own screen.

5. EVALUATION

In our system, users can perform a sequence of updates on a complex 3D object displayed in the browser, and these updates are sent to other users through a shared database (Firebase on GCP). The updated content is then displayed on the remote user's browser. To ensure a smooth user experience and minimize network stress, we conducted an experiment to measure synchronization time and update time between browsers and the database. The experiment proceeded as follows: i) user a clicks the update button on their browser, initiating an update of their edited content at time t_1; ii) the browser calls the update() method to update Firebase at time t_2, and the update process is completed at time t_3; and iii) user B's browser detects the update in the

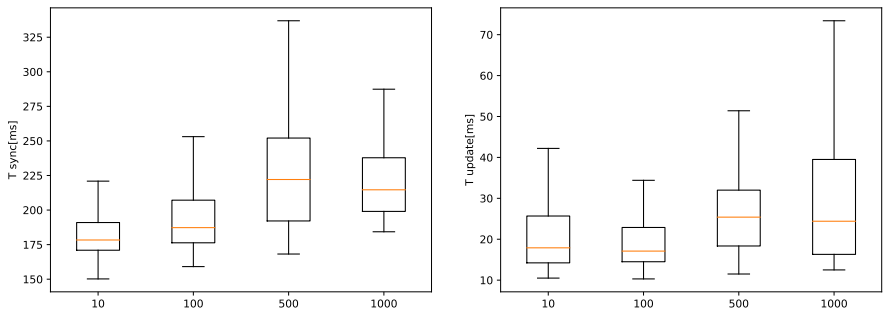
database at time t_4 and receives the updated information, completing the reconstruction of the virtual object animation at time t_5.

The time between t_5 and t_1 is referred to as the synchronization time between browsers, while the time between t_3 and t_2 is referred to as the update time of the database. Collectively, these times are referred to as the processing time. Regarding the size of data exchanged between the database and the browsers, we observed that it is approximately 3KB when there are 10 virtual objects.

5.1. Impact of the complexity of manipulated object on the processing time

In the proposed system, the 3D object representing an avatar is composed of multiple basic objects. To assess the impact of the number of objects on the processing time, we conducted experiments where we varied the number of objects from 10 to 1000. The processing time was measured using the Performance API, which allows us to measure the execution time of JavaScript programs in milliseconds. The experiment was repeated 300 times, and the results are presented in Figure 6.

From the Figure 6, it can be observed that as the number of objects increases from 10 to 1000, both the synchronization time (Figure 6(a))and the update time show a slight increase (Figure 6(b)). However, even with a larger number of objects, the third quartile of the synchronization time and the third quartile of the update time remain below 250 ms and 40 ms, respectively. This indicates that the stress caused by synchronization latency is limited, even for complex objects, although it may vary depending on the specific use case. It should be noted that for real-time collaborative work synchronized with background music, where an acceptable latency is around 30 ms for remote session systems used by musicians, our system may not be suitable. However, for other applications that are less interactive, such as terrestrial digital broadcasting systems with a latency of approximately 2 seconds, a synchronization time of 250 ms should be sufficient.



(a) (b)

Figure 6. Impact of the complexity of manipulated object on the processing time, where (a) depicts the synchronization time and (b) depicts the update time. The horizontal axis represents the number of basic objects that make up the shared 3D object

5.2. Implementation-dependent processing time

We measured the performance of Firebase RealtimeDatabase using the database profiler built into FirebaseCLI, which logs all database activities and generates a detailed report. In the experiment, we checked the update process logs with various numbers of objects and found the average processing time, as summarized in Table 3. We also used gcping [16] to measure the ping time in the experimental environment and found that the median of upload and download latency is 170 ms. Combining these supplementary results with the previous experimental results, we estimate the overhead of our implementation to be approximately 9.92 to 32.18 ms on average, with 10 to 100 objects.

Table 3. Average update time measured on Firebase

Number of objects	Average update time (ms)
10	3.67
100	8.83
500	12.25
1000	19.86

6. RELATED WORK

In the field of remote instruction, numerous efforts have been made to teach people correct movements using various technologies such as cave automatic virtual environments (CAVEs), head-mounted displays (HMDs), and inertia measurement units (IMUs). One notable system in this regard is ImmerTai [17], an immersive virtual reality (VR) system designed for training in Tai Chi movements. ImmerTai captures the movements of a Tai Chi expert and transmits them to remote students wearing HMDs. The system also captures the movements of the students and evaluates their performance, creating collaborative learning experiences in virtual groups. Motion capture in ImmerTai is performed using Microsoft Kinect. Studies have shown that using ImmerTai can result in a learning process that is up to 17% faster compared to traditional PC-based learning. Another study conducted by Liu *et al*. [18] utilized an inertial measurement unit (IMU) called perception neuron for motion capture instead of Kinect. In a different study Li *et al*. [19], investigated a method for recognizing the motions of Baduanjin using data sequences acquired from IMUs. The study collected data from 54 participants and employed various techniques such as dynamic time warping (DTW), hidden Markov models (HMM), and recurrent neural networks (RNN) to successfully recognize the movements of Baduanjin. However, their focus did not extend to displaying the motion on a browser or converting the learner's motion into an editable animation of a virtual object, which are key features in the proposed method.

Numerous studies have explored the application of extended reality (xR) technology for remote collaboration [20]. One notable example is the web-based VR system called CLEV-R [21], which creates virtual simulations of various physical environments such as lecture rooms, classrooms, libraries, and meeting rooms. Each simulated space offers unique functionalities and content, aiming to facilitate interaction and communication between students and teachers primarily through text chat. CLEV-R also includes features for targeted audio broadcasting, enabling in-school broadcasts or live streaming using a webcam. Although it shares similarities with our proposed system in terms of being browser-based, CLEV-R does not provide adequate support for physical instruction.

Wu *et al*. [22] investigated the effectiveness of using HMDs for immersive VR (IVR). A systematic literature review conducted from 2013 to 2019 [23] demonstrated that IVR with HMDs outperforms non-experiential (PC-based) learning methods. The study found that IVR had the greatest impact on K-12 students, science education, and skill development, particularly when incorporating simulations or virtual world representations. The meta-analysis further suggested that HMDs enhance both knowledge acquisition and skill development, with the learning effects persisting over time. These findings support the efficacy of the experiential approach through the manipulation of 3D avatars, as implemented in our proposed system, and encourage further exploration of IVR in future studies.

7. CONCLUSION

This paper presents a system that aims to improve the efficiency of online education by enabling remote instruction of body movements. The system allows for the deformation of virtual 3D objects and provides direct editing of animations within a web browser. To ensure accessibility and user-friendliness, the system is developed as a single-page application that facilitates real-time sharing of 3D object data through Firebase. In our future work, we intend to enhance the system by integrating an efficient method to incorporate motion data acquired from PoseNet into animations. Additionally, we plan to evaluate the effectiveness of the system by implementing it in elementary and junior high schools, with a focus on assessing its impact on remote instruction and learning outcomes.

REFERENCES

[1] N. T. Fitter, N. Raghunath, E. Cha, C. A. Sanchez, L. Takayama, and M. J. Mataric, "Are we there yet? comparing remote learning technologies in the university classroom," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2706–2713, Apr. 2020, doi: 10.1109/LRA.2020.2970939.

[2] Z. Chen *et al*., "Learning from home: a mixed-methods analysis of live streaming based remote education experience in Chinese colleges during the COVID-19 pandemic," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, May 2021, pp. 1–16, doi: 10.1145/3411764.3445428.

[3] B. L. Moorhouse and A. M. Beaumont, "Utilizing video conferencing software to teach young language learners in Hong Kong during the COVID-19 class suspensions," *TESOL Journal*, vol. 11, no. 3, Sep. 2020, doi: 10.1002/tesj.545.

[4] E. Z. Barsom, M. Graafland, and M. P. Schijven, "Systematic review on the effectiveness of augmented reality applications in medical training," *Surgical Endoscopy*, vol. 30, no. 10, pp. 4174–4183, Oct. 2016, doi: 10.1007/s00464-016-4800-6.

[5] M. R. Desselle, R. A. Brown, A. R. James, M. J. Midwinter, S. K. Powell, and M. A. Woodruff, "Augmented and virtual reality in surgery," *Computing in Science & Engineering*, vol. 22, no. 3, pp. 18–26, May 2020, doi: 10.1109/MCSE.2020.2972822.
[6] C.-S. Chan, J. Bogdanovic, and V. Kalivarapu, "Applying immersive virtual reality for remote teaching architectural history," *Education and Information Technologies*, vol. 27, no. 3, pp. 4365–4397, Apr. 2022, doi: 10.1007/s10639-021-10786-8.
[7] M. Hernández-de-Menéndez, A. V. Guevara, and R. Morales-Menendez, "Virtual reality laboratories: a review of experiences," *International Journal on Interactive Design and Manufacturing (IJIDeM)*, vol. 13, no. 3, pp. 947–966, Sep. 2019, doi: 10.1007/s12008-019-00558-7.
[8] K. Nesenbergs, V. Abolins, J. Ormanis, and A. Mednis, "Use of augmented and virtual reality in remote higher education: a systematic umbrella review," *Education Sciences*, vol. 11, no. 1, p. 8, Dec. 2020, doi: 10.3390/educsci11010008.
[9] A. Yoshimura and C. W. Borst, "Remote instruction in virtual reality: A study of students attending class remotely from home with vr headsets," in *Hansen, C., Nürnberger, A. & Preim, B. (Hrsg.), Mensch und Computer 2020 - Workshopband. Bonn: Gesellschaft für Informatik e.V.*, 2020. doi:10.18420/muc2020-ws122-355.
[10] B. Lawson and R. Sharp, *Introducing HTML5*. New Riders, 2011.
[11] G. Anthes, "HTML5 leads a web revolution," *Communications of the ACM*, vol. 55, no. 7, pp. 16–17, Jul. 2012, doi: 10.1145/2209249.2209256.
[12] B. Maclntyre and T. F. Smith, "Thoughts on the future of WebXR and the immersive Web," in 2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct), Oct. 2018, pp. 338–342, doi: 10.1109/ISMAR-Adjunct.2018.00099.
[13] C. Anderson, "The model-view-viewmodel (MVVM) design pattern," in Pro Business Applications with Silverlight 5, Berkeley, CA: Apress, 2012, pp. 461–499, doi: 10.1007/978-1-4302-3501-9_13.
[14] S. Buna, *GraphQL in Action*, Simon and Schuster, 2021.
[15] A. Quiña-Mera, P. Fernandez, J. M. García, and A. Ruiz-Cortés, "GraphQL: A Systematic Mapping Study," *ACM Computing Surveys*, vol. 55, no. 10, pp. 1–35, Oct. 2023, doi: 10.1145/3561818.
[16] gcping. (2020). [Online]. Available: https://gcping.com/.
[17] X. Chen *et al*., "ImmerTai: immersive motion learning in VR environments," *Journal of Visual Communication and Image Representation*, vol. 58, pp. 416–427, Jan. 2019, doi: 10.1016/j.jvcir.2018.11.039.
[18] J. Liu, Y. Zheng, K. Wang, Y. Bian, W. Gai, and D. Gao, "A real-time interactive tai chi learning system based on VR and motion capture technology," *Procedia Computer Science*, vol. 174, pp. 712–719, 2020, doi: 10.1016/j.procs.2020.06.147.
[19] H. Li, S. Khoo, and H. J. Yap, "Implementation of sequence-based classification methods for motion assessment and recognition in a traditional Chinese sport (Baduanjin)," *International Journal of Environmental Research and Public Health*, vol. 19, no. 3, p. 1744, Feb. 2022, doi: 10.3390/ijerph19031744.
[20] A. Schäfer, G. Reis, and D. Stricker, "A survey on synchronous augmented, virtual, andmixed reality remote collaboration systems," *ACM Computing Surveys*, vol. 55, no. 6, pp. 1–27, Jul. 2023, doi: 10.1145/3533376.
[21] T. Monahan, G. McArdle, and M. Bertolotto, "Virtual reality for collaborative e-learning," *Computers & Education*, vol. 50, no. 4, pp. 1339–1353, May 2008, doi: 10.1016/j.compedu.2006.12.008.
[22] B. Wu, X. Yu, and X. Gu, "Effectiveness of immersive virtual reality using head-mounted displays on learning performance: a meta-analysis," *British Journal of Educational Technology*, vol. 51, no. 6, pp. 1991–2005, Nov. 2020, doi: 10.1111/bjet.13023.
[23] A. F. D. Natale, C. Repetto, G. Riva, and D. Villani, "Immersive virtual reality in K-12 and higher education: A 10-year systematic review of empirical research," *British Journal of Educational Technology*, vol. 51, no. 6, pp. 2006–2033, Nov. 2020, doi: 10.1111/bjet.13030.

BIOGRAPHIES OF AUTHORS

Nagaki Kentarou received the B.E. degree in information engineering from Hiroshima University in 2021, and is master's student at Hiroshima University. His research interests include distributed systems and network applications. He can be contacted at email: m215356@hiroshima-u.ac.jp.

Fujita Satoshi received the B.E. degree in electrical engineering, M.E. degree in systems engineering, and Dr.E. degree in information engineering from Hiroshima University in 1985, 1987, and 1990, respectively. He is a professor at Graduate School of Advanced Science and Engineering, Hiroshima University. His research interests include communication algorithms, parallel algorithms, graph algorithms, and parallel computer systems. He is a member of the Institute of Electronics, Information and Communication Engineers (IEICE), the Information Processing Society of Japan, the Japan Society for Industrial and Applied Mathematics (JSIAM) and IEEE Computer Society. He can be contacted at email: fujita@hiroshima-u.ac.jp.