Evacuation Drill Support Network System with Behavior History Recording/Reproduction Functions Utilizing Extended Reality

Riku Tono
Ritsumeikan University
Graduate School of Information
Science and Engineering
Osaka, Japan
is0568ke@ed.ritsumei.ac.jp

Hideaki MIYAJI   Hiroshi YAMAMOTO
Ritsumeikan University
College of Information
Science and Engineering
Osaka, Japan

***Abstract*—In Japan, evacuation drills are legally mandated for educational institutions and companies to promote disaster prevention and risk mitigation. However, conventional drills assume a condition where all participants are in the same place at the same time, making it difficult to share detailed behavioral information with those who cannot participate. Existing studies propose an extended reality (XR)-based system that can simulate realistic scenarios (e.g., flames, smoke, and sound), enhancing immersion and training effectiveness. However, these systems typically do not consider recording detailed behaviors of the participants or providing intuitive feedback to them. Moreover, the XR-based system rely on SLAM functionality on the XR headset for recognizing 3D structure of the field and localing the user, which restricts the number and size of environmental maps including the structure of a specific area that can be managed. As a result, their application to large-scale drills spanning entire facilities is difficult. To address these challenges, we propose a system that records behaviors of the participants in real time using motion-capture sensors and enables intuitive sharing of the recorded behaviors with others who conduct drills at the same location. In addition, rather than managing environmental maps solely on XR headsets, the proposed system centrally manages the maps on a server and dynamically distributes the appropriate map according to the current position of headset. This approach mitigates SLAM-related limitations on the number and coverage area of environmental maps to supports evacuation drills across large-scale facilities.**

***Index Terms*—Extended reality (XR), evacuation training, motion capture, SLAM, environmental map**

I. Introduction

Japan is geographically prone to natural disasters due to its location, topography, geology, and meteorological conditions. According to surveys by the Cabinet Office in Japan, natural disasters cause extensive damage every year, and in the case of large-scale disasters, the number can ranges from several thousand to tens of thousands [1].

In response to this background, evacuation drills for disaster prevention and mitigation are mandated in Japan, mainly in educational institutions, companies, and local governments. In particular, in facilities where large numbers of people gather, such as schools and office buildings, regular evacuation drills are indispensable for enabling prompt responses during emergencies. However, most conventional evacuation drills still rely on the condition where all trainees gather at the same time and location , and there is no intuitive method to share the situations of the drills with those who cannot participate.

To address such issues, existing studies propose a fire evacuation drill support system utilizing extended reality (XR) [2]. By using XR headsets, virtual elements such as flames, smoke, and sounds are superimposed onto the real world, thereby creating an immersive training environment as if trainees are in an actual fire scene. These systems also record behavioral data such as evacuation time, evacuation routes, number of contacts with flames or smoke, and frequency and duration of postural changes. After the drill, the data are delivered as textual feedback presented via XR headsets. Compared with traditional tabletop or paper-based drills, these systems are expected to provide higher immersion and learning effects. However, the feedback is provided only in textual information, and sufficient mechanisms are not developed to allow trainees to intuitively review their own behaviors or to share them among multiple trainees.

In addition, XR headsets are equipped with simultaneous localization and mapping (SLAM) functionality, which is essential for recognizing 3D structures of the field and localizing trainees to realize the stable superimpose of virtual content onto the real space. However, current XR devices impose restrictions on the coverage area of the real space that can be recongnized [3]. Therefore, it is difficult to realize evacuation drills that encompass an entire large-scale facility such as buildings with multiple floors as the training environment.

To address these challenges, this study proposes a new system that enables detailed recording behaviors of trainees and reproducing them by superimposing the virtual objects corresponding with the past trainees onto the real space utilizing XR technologies. In the proposed system, the movements and behaviors of the trainees are recorded using motion capture sensors in real time and are subsequently shared with others in an intuitive manner. Furthermore, to overcome the limitations of areas that can be managed by XR headsets, the proposed system manages environmental maps including 3D structures of the specific area of the field on a centralized server rather than on individual devices. By recognizing the location of the trainee, the server selects and distributes appropriate maps to the trainee. With this functionality, the proposed system enables effective evacuation drills that can flexibly support entire facilities.

II. Related Work and Objectives of This Study

A. *Fire Evacuation Training System Using Mixed Reality*

Sakaguchi et al. propose a fire evacuation training system that utilizes extended reality (XR) [2]. In this system, flames, smoke, and sound are superimposed onto the real world using an XR headset, thereby reproducing fire situations and enabling highly immersive evacuation training. Furthermore, the system records data related with the behaviors of the trainee such as evacuation time, evacuation routes, number of contacts with flames and smoke, and the frequency and duration of postural changes. These data are presented as feedback on the XR headset worn by trainees. However, in this study, the feedback is primarily presented in textual format, which lacks immersion when provided to a large number of trainees.

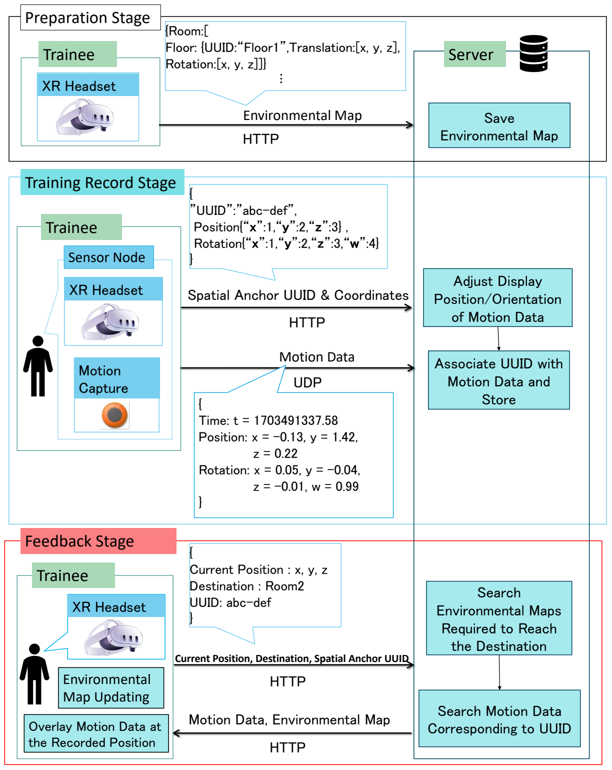
B. *VR Evacuation Training System with Gaze Visualization*

Ii et al. propose a VR-based evacuation training system equipped with functions for visualizing gaze of trainees [4]. In this system, the trainees wear a head-mounted display and perform evacuation actions within a virtual environment. During the training, their movement trajectories and gaze data are recorded, allowing the trainees to review their evacuation behavior afterward through third-person perspectives. This enables insights such as "the trainee kept focusing on the fire" or "the trainee failed to notice injured persons," thereby raising awareness of habits on evacuation and decision-making tendencies. However, this system does not faithfully reflect the structure or spatial characteristics of actual buildings, making it less suitable for practical training aligned with real-world environments.

C. *Objectives of This Study*

Existing studies propose XR-based evacuation training systems. However, the systems do not assume the implementation of functions that record detailed evacuation behaviors and provide intuitive feedback to multiple trainees.

Therefore, this study proposes a new system that records the behaviors of trainees using motion capture sensors and shares them with other trainees who later conduct training at the same location. Specifically, the system employs the SLAM functionality built on XR headsets to perform accurate indoor localization, linking the behaviors of trainees with their positions for recording. The SLAM (Simultaneous Localization and Mapping) is a technique that simultaneously performs self-localization and recognition of 3D structures in a field for map construction, using information obtained from sensors such as cameras, LiDAR, and IMUs (Inertial Measurement Units) [5]. The recorded behavioral data are then shared with



Preparation Stage
Trainee
XR Headset
{Room:[
Floor: {UUID:"Floor1",Translation:[x, y, z],
Rotation:[x, y, z]]}
⋮
Environmental Map
HTTP
Server
Save Environmental Map
Training Record Stage
Trainee
Sensor Node
XR Headset
Motion Capture
{
"UUID":"abc−def",
Position{"x":1,"y":2,"z":3} ,
Rotation{"x":1,"y":2,"z":3,"w":4}
}
Spatial Anchor UUID & Coordinates
HTTP
Motion Data
UDP
{
Time: t = 1703491337.58
Position: x = −0.13, y = 1.42, z = 0.22
Rotation: x = 0.05, y = −0.04, z = −0.01, w = 0.99
}
Adjust Display Position/Orientation of Motion Data
Associate UUID with Motion Data and Store
Feedback Stage
Trainee
XR Headset
Environmental Map Updating
Overlay Motion Data at the Recorded Position
{
Current Position : x, y, z
Destination : Room2
UUID: abc−def
}
Current Position, Destination, Spatial Anchor UUID
HTTP
Motion Data, Environmental Map
HTTP
Search Environmental Maps Required to Reach the Destination
Search Motion Data Corresponding to UUID

Fig. 1: Overview of the proposed system.

subsequent trainees at the same site and displayed on the XR headsets, enabling intuitive sharing of evacuation behaviors during training.

In addition, instead of storing and managing environmental maps solely within XR headsets, the proposed system introduces a mechanism to store the maps on a server and dynamically distribute them based on the current position of the trainee. This approach overcomes the limitations of SLAM with respect to the number of maps that can be stored on the headset, thereby enabling evacuation training across large-scale facilities.

III. Proposed Evacuation Training Support Network System

A. *Overview of the Proposed System*

An overview of the proposed system is shown in Fig. 1. The proposed system consists of sensor nodes worn by trainees and a server that manages behavioral records and environmental maps. The processes in the proposed system are divided into three stages: a Preparation Stage, a Training Record Stage, and a Feedback Stage.

In the Preparation Stage, the sensor node relies on the XR headset (Meta Quest 3) to scan the indoor environment through its built-in SLAM functionality for constructing the environmental maps. The generated map is then transmitted to the server.

In the Training Record Stage, the sensor node estimates its position using the SLAM function of the XR headset. To record the initial position just after starting the training, a data

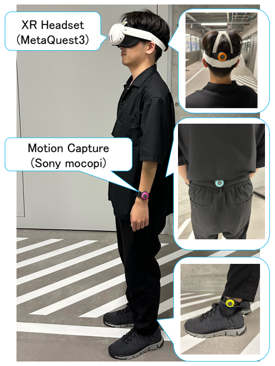


Fig. 2: Configuration of the sensor node.

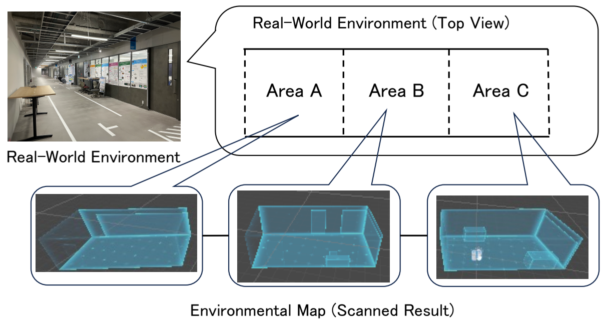

Real-World Environment (Top View)
Area A
Area B
Area C
Real-World Environment
Environmental Map (Scanned Result)

Fig. 3: Example of scanned results of real-world environments.

structure called a spatial anchor is used, including a specific position and orientation of the trainee on the coordiante system of the virtual space. Subsequently, motion data obtained from a mobile motion capture device (Sony Mocopi) are recorded with timestamps and transmitted to the server in real time, along with the results of self-localization and the UUID of the spatial anchor. The server associates the received motion data with the spatial anchor corresponding with the current position and stores them as behavioral records during training.

Furthermore, during both the Training Record Stage and the Feedback Stage, the sensor node continuously transmits the positional information of the trainee to the server. Based on this data, the server determines which environmental maps should be acquired or discarded and sends corresponding instructions to the sensor node. Following these instructions, the sensor node dynamically updates its stored maps, ensuring that only the necessary data are retained within the limited capacity of the sensor node.

In the Feedback Stage, the sensor node transmits the UUID of the spatial anchor corresponding to the recorded motion data. The server searches for the motion data associated with the received UUID and sends it back to the sensor node. The sensor node then superimposes the corresponding behavioral records onto the real space.

B. *Configuration of the Sensor Node*

The hardware configuration of the sensor node is shown in Fig. 2. For motion capture, the system employs a mobile motion capture device (Sony Mocopi). Mocopi estimates the full-body three-dimensional pose from time-series accelerometer and gyroscope data obtained from a small number of wearable sensors. Although IMU-based position estimation generally suffers from drift caused by the integration of acceleration signals, Mocopi mitigates this issue by incorporating a pre-trained AI model that directly estimates joint positions at each sensor-attached body part while compensating for accumulated errors. Furthermore, body joints without attached sensors (e.g., elbows and knees) cannot be uniquely determined only from geometric constraints due to the many degrees of freedom of the human body. Mocopi addresses this challenge by applying AI-driven pose interpolation to infer natural intermediate joint positions. Through these mechanisms, Mocopi achieves continuous and natural full-body motion estimation despite its minimal sensor configuration [6] [7].

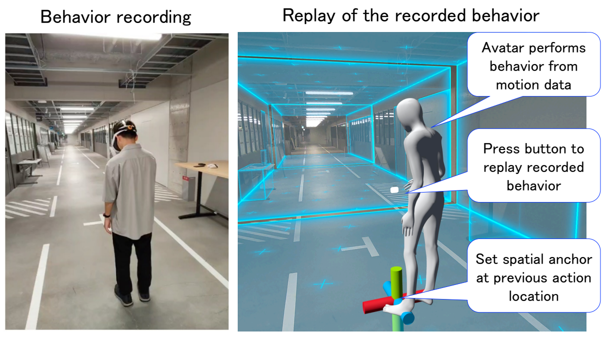
For spatial scanning and self-localization, the SLAM functionality of the XR headset (Meta Quest 3) is used. The Meta Quest 3 integrates SLAM by combining multiple external cameras with an internal IMU for the SLAM. This allows real-time understanding of the three-dimensional structure of the training space and the current positon of the headset, simultaneously [8]. During the Preparation Stage, the SLAM functionality is used to scan the training space and generate an environmental map in JSON format. In the Training Record Stage and the Feedback Stage, the headset transmits real-time positional information estimated by the SLAM to the server. The server, in turn, issues instructions to add or delete environmental maps on the headset. Additionally, in the Feedback Stage, the XR headset overlays recorded behavioral data onto the real environment.

C. *Process Flow in the Preparation Stage*

In the Preparation Stage, the SLAM functionality on the XR headset is used to create environmental maps of the training space. These maps are then transmitted and registered on the server, enabling subsequent processes to utilize shared spatial data. First, the trainee starts the sensor node and walks around the training space while scanning the environment using the SLAM. The generated environmental maps are stored in JSON format within the sensor node. An example of scanned spatial data is shown in Fig. 3. The map is generated in JSON format and is assigned UUIDs. The server manages these maps and records spatial relationships among them. In this way, the server maintains a graph structure in which nodes represent areas and edges represent connections. This structure forms the foundation for dynamically adding or deleting maps during training according to changes in the position of the trainee.

D. *Process Flow in the Training Record Stage*

In the Training Record Stage, the sensor node first performs self-localization using the SLAM functionality of the XR headset. A spatial anchor is set at the initial training position to record the starting point. At the same time, motion capture is initiated. The UUID and positional information of the spatial anchor and the motion data are transmitted from the XR headset to the server. The server adjusts the display


Behavior recording
Replay of the recorded behavior
Avatar performs behavior from motion data
Press button to replay recorded behavior
Set spatial anchor at previous action location
Fig. 4: Example of overlaid behavioral records.

position and orientation of the motion data based on the coordinates of the spatial anchor and stores them together with the UUID. This ensures that behaviors of the trainee are stored in association with their positions.

Additionally, during the Training Record Stage, the sensor node continuously transmits its current position to the server. The server identifies the current environmental map and adjacent areas corresponding to the location of the trainee and sends instructions for updating environmental maps. These instructions specify which maps to add and which to delete.

E. Process Flow in the Feedback Stage

In the Feedback Stage, the sensor node performs self-localization using the SLAM function of the XR headset. The sensor node then transmits the UUID of the spatial anchor corresponding to the recorded motion data to the server. The server retrieves the motion data corresponding with the UUID and transmits it to the sensor node. After that, the sensor node overlays the retrieved motion data onto the real-world environment displayed on the XR headset, as illustrated in Fig. 4. During the Feedback Stage, the sensor node also continues to transmit real-time positional information to the server. The server uses these data to provide instructions for dynamically updating environmental maps. This process is similar to that in the Training Record Stage.

IV. Environmental Map Updating Method

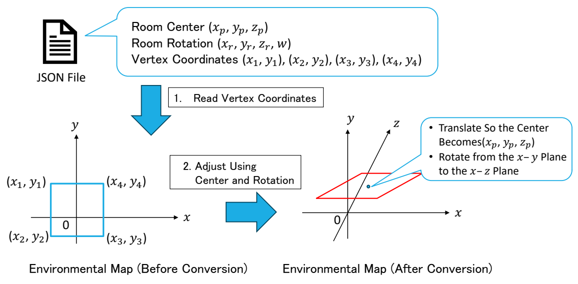
This section provides a detailed explanation of the process flow in the Preparation Stage, described in Section III-C, especially the method for updating environmental maps.

A. Details of Environmental Maps

The environmental maps used in the proposed system are generated by the SLAM functionality embedded in the XR headset. When a trainee wears the XR headset, the center of the play area (boundary) specified at startup is automatically set as the origin (0,0,0) of the coordinate system of the environmental map. As the user physically moves within the training space for scanning the surrounding environment, three-dimensional structures such as floors and walls are captured and recorded as an environmental map. The generated map reflects the structure of the real-world environment, allowing virtual objects to be placed in accurate positions. Up to 15

TABLE I: Data structure of a JSON file defining an environment map.

Item	Value	Description
UUID	41A6DAAFC43	Identifier of the map
SemanticClassifications	FLOOR	Classification label
Transform.Translation	[1.3897, −0.0377, −0.1488]	Center coordinates
Transform.Rotation	[270.0, 7.7712, 0.0]	Orientation
PlaneBounds.Min	[−4.1162, −2.2421]	Minimum boundary coordinates
PlaneBounds.Max	[4.1169, 2.2444]	Maximum boundary coordinates
PlaneBoundary2D	[4.1147, 2.2368], [−4.1162, 2.2444], [−4.1162, −2.2421], [4.1169, −2.2421]	Floor polygon vertices $[x, y]$


JSON File
Room Center (x_p, y_p, z_p)
Room Rotation (x_r, y_r, z_r, w)
Vertex Coordinates $(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)$
1. Read Vertex Coordinates
2. Adjust Using Center and Rotation
y
x
0
(x_1, y_1)
(x_4, y_4)
(x_2, y_2)
(x_3, y_3)
z
• Translate So the Center Becomes(x_p, y_p, z_p)
• Rotate from the $x-y$ Plane to the $x-z$ Plane
Environmental Map (Before Conversion)
Environmental Map (After Conversion)
Fig. 5: Overview of the conversion method.

environmental maps can be stored on the internal storage of the XR headset, each uniquely identified by a UUID. The format of the map is JSON, where geometric information such as three-dimensional structures and metadata of the map are stored in text-based format.

B. Management of Environmental Maps on the Server

In the proposed system, environmental maps scanned by the XR headset are centrally managed by the server. Each map corresponds to a single area in the training space, and the data are stored as JSON files. The data structure of a JSON file defining an environmental map is shown in Tab. I. This JSON file includes a UUID for identifying the area, semantic classifications (SemanticClassifications), transformation information (position and orientation), and geometric information such as floor shapes and ranges (PlaneBoundary2D).

The identification of environmental maps on the server is based on the UUID assigned to elements classified as `FLOOR` in the JSON file. The `FLOOR` element represents the floor surface of a area, and the server uses this information to determine the position and extent of the area. In particular, the `PlaneBoundary2D` stores multiple vertices representing the floor contour in $[x, y]$ format. Since the coordinate system is different between the headset and the server, a conversion processing of the coordinates of the map is required.

An overview of the conversion method is shown in Fig. 5. In this method, the values of the `Rotation` item in the `Transform` field are first used to rotate the floor geometry into the correct orientation. Next, the `Translation` values are used as the center coordinates of the floor, and all vertex coordinates are translated relative to this center. Through this

TABLE II: Data structure of the adjacency list.

Field	Example
from	Area A UUID
to	Area B UUID, Area C UUID, Area D UUID

Note: The `to` field stores UUIDs of multiple adjacent areas as a comma-separated string. When a one-way relation `from`→`to` is registered, the reverse relation `to`→`from` is also automatically registered on the server, effectively treating it as a bidirectional adjacency.

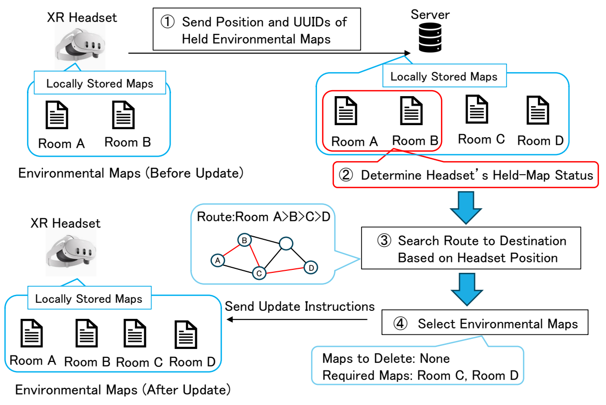

XR Headset
① Send Position and UUIDs of Held Environmental Maps
Server
Locally Stored Maps
Room A
Room B
Environmental Maps (Before Update)
Locally Stored Maps
Room A
Room B
Room C
Room D
② Determine Headset' s Held-Map Status
XR Headset
Route:Room A>B>C>D
③ Search Route to Destination Based on Headset Position
Send Update Instructions
④ Select Environmental Maps
Locally Stored Maps
Room A
Room B
Room C
Room D
Environmental Maps (After Update)
Maps to Delete: None
Required Maps: Room C, Room D

Fig. 6: Flow of environmental map updating method.

procedure, the environmental map stored in the JSON file can be converted into the coordinate system on the server.

Additionally, the server constructs adjacency lists that describe the physical spatial relationship among all environmental maps. The data structure of the adjacency list is shown in Tab. II. The adjacency refers to the condition in which areas are directly connected and accessible. For example, if Area A is adjacent to Area B, the UUID of Area B is registered in the adjacency list of Area A, and vice versa. This adjacency information is maintained internally on the server as a graph structure, where nodes represent areas and edges represent connections between the areas. This structure is utilized for route searching during training and for determining priorities when updating the environmental map on the sensor node.

C. *Environmental Map Updating Method*

The storage capacity of XR headsets is limited, and the maximum number of environmental maps that can be stored simultaneously is 15. To address this, the proposed system introduces a mechanism in which only the maps necessary for the current position of trainee are retained, while unnecessary maps are deleted. The process is applied in both the Training Record Stage and the Feedback Stage. The flow of the environmental map updating method is shown in Fig. 6.

At the start of training, the sensor node sends its estimated current position obtained through SLAM, along with a list of UUIDs of currently stored maps, to the server. Based on this information, the server identifies the area where the trainee is located and its adjacent areas, and extracts the corresponding maps. Furthermore, the server uses a Dijkstra algorithm to

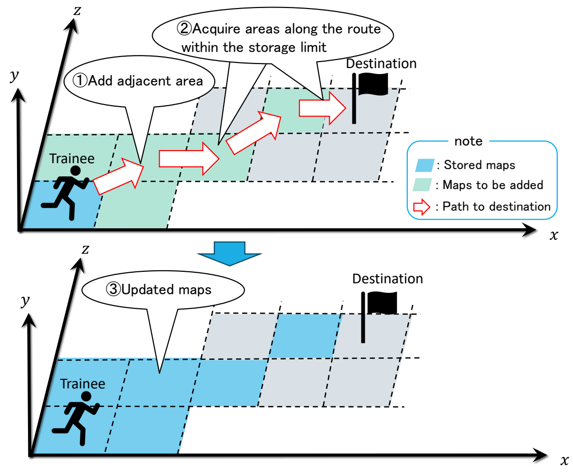


Fig. 7: Selection and update procedure for maps.

search for a pre-specified destination in the adjacency graph of areas, and extracts the maps corresponding to the areas along the path.

Here, the Dijkstra algorithm is a method for determining the shortest distance from a starting point to each vertex in a weighted graph with non-negative edge weights [9]. The shortest path and its cost to the destination can be obtained by iteratively selecting the vertex with the smallest tentative distance and updating the distances of its neighboring vertices. In this setting, vertices correspond to areas, and edge weights correspond to the distances between the centers of adjacent areas. Based on the results, the server selects which environmental maps should be added and which should be deleted, and transmits this information to the sensor node via HTTP communication. If the number of maps to be acquired exceeds the storage limit, the server specifies UUIDs of maps that can be deleted according to priority. The server then instructs their removal and subsequently directs the sequential downloading of the required maps.

Figure 7 illustrates the selection and update procedure for environmental maps. High-priority maps include those corresponding to the current area of trainee and adjacent areas. This ensures accurate self-localization and proper placement of virtual objects in the current location. Additionally, maps along the path to the destination are sequentially selected as acquisition targets, with the number limited to 15 according to their necessity. Following the received instructions, the XR headset deletes unnecessary maps, downloads the required maps in JSON format from the server and stores them locally. This enables smooth updating of maps during training while retaining only the maps necessary for the current and upcoming locations.

V. Evaluation of the Effectiveness of the Proposed System

A. *Experimental Setup*

Because the proposed system should promptly optimize the environmental maps on the sensor node based on the current position of trainee, we mainly evaluate the real-time performance of updating the maps. Focusing on the in-training map-update pipeline described in Sections 3 and 4, we measure the end-to-end processing time from selecting the environmental maps required for a route to a destination to completing their acquisition.

The target pipeline consists of four components: (i) transmission of the current position and the list of held maps, (ii) server-side selection of required maps and dispatch of addition/deletion instructions, (iii) deletion of unneeded maps and downloads of necessary maps on the sensor node, and (iv) loading the downloaded JSON files as environmental maps. Here, the server-side selection denotes the procedure that uses the trainee position and the UUIDs of the stored maps. This process searches the areas at the current location and along the route to the destination, extracts the relevant maps, and determines which should be added to or deleted from the sensor node. In this experiment, we evaluate the processing time of each component and derive the average of the total processing time.

Experiments are conducted under two network conditions that differ in upstream (backhaul) characteristics. Environment α is a relatively stable network, whereas Environment β is a congested network with many users. Under both conditions, we repeat the same processing ten times and calculate the average processing time.

At the beginning of each trial, the XR headset on the sensor node holds only one environmental map (the map of the area where the sensor node is currently located). The sensor node then acquires 14 additional maps consecutively to reach the maximum of 15 maps. This scenario emulates a use case in which the route to the destination is long and traverses many areas, placing the highest load on the map-update pipeline. All target maps are stored in JSON format. Each map has an average file size of approximately 10 KB, resulting in a total of about 140 KB for 14 files.

B. Results

Table III summarizes the results. In the stable network (Environment α), the average end-to-end time is approximately 0.053 seconds, and even in the congested network (Environment β) it is approximately 0.117 seconds. Since both are well below one second, the system can update environmental maps in real time relative to trainee movement.

The dominant portion of the total time is consumed by the server-side process of selection and instruction dispatch, accounting for approximately 92% in Environment α and 97% in Environment β. Since this step directly reflects network latency and server load, differences in network conditions appear prominently in the results. By contrast, “new download of environmental maps” and “JSON file loading” each take less than a few milliseconds in both environments, which can be attributed to the small data size and light-weight I/O on the XR headset. Even in Environment β, the total time remains under 0.2 seconds, indicating that sufficient real-time performance can be achieved in typical indoor networks such as office or commercial buildings. Overall, the results demonstrate that the proposed system can maintain responsiveness in realistic indoor training scenarios.

TABLE III: Average processing time

Process	Time (ms), α	Time (ms), β
Position & held-map list transmission	1.0	1.0
Map selection & instruction dispatch (server)	49	114
New download of environmental maps	0.002	0.002
JSON file loading	3.0	2.0
Total	53	117

VI. Conclusion and Future Work

This study proposed a system that records the movements of trainee in real time using motion-capture sensors and enables intuitive presentation and sharing of detailed behaviors with subsequent trainees. To overcome constraints on the creation and management of environmental maps on XR headsets, the system centrally manages maps on a server and dynamically updates them according to the current position of the trainee. Through evaluation experiments, we demonstrated that environmental maps can be updated in real time in response to trainee movement during training. As future work, we will apply AI-based analysis to the recorded motion data to automatically recognize key trainee behaviors and identify inefficient or unsafe movement patterns. By leveraging machine-learning models on the time-series sensor data, we aim to derive objective performance indicators and provide more effective feedback during evacuation training.

Acknowledgement

This work is supported by Japan Society for the Promotion of Science (JSPS) KAKENHI Grant Number JP24K20774, 24K03045.

References

[1] Cabinet Office, Government of Japan, “White Paper on Disaster Management 2025,” [Online]. Available: https://www.bousai.go.jp/kaigirep/hakusho/r7.html. Accessed: Aug. 2025. [in Japanese].
[2] S. Sakaguchi and M. Makino, “A mixed reality-based fire evacuation drill system,” Proceedings of the International Workshop on Advanced Imaging Technology (IWAIT), 2021.
[3] Meta, “Unity Scene Overview,” [Online]. Available: https://developers.meta.com/horizon/documentation/unity/unity-scene-overview. Accessed: Aug. 2025.
[4] C. Ii, Y. Ichino, and H. Mitsuhara, “Gaze Visualization and Its Effects in Reviewing VR Evacuation Training,” in *Proc. IEICE General Conf.*, 2024, pp. 495–498. [in Japanese].
[5] H. Taheri and Z. C. Xia, “SLAM; definition and evolution,” *Engineering Applications of Artificial Intelligence*, vol. 97, Art. no. 104032, 2021.
[6] S. Sharma, S. Verma, M. Kumar, and L. Sharma, “Use of motion capture in 3D animation: motion capture systems, challenges, and recent trends,” in *Proc. 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, 2019, pp. 289–294.
[7] Sony, “mocopi: Mobile motion capture,” [Online]. Available: https://www.sony.jp/mocopi/. Accessed: Aug. 2025. [in Japanese].
[8] T. Hu, F. Yang, T. Scargill, and M. Gorlatova, “Apple vs. Meta: A Comparative Study on Spatial Tracking in SOTA XR Headsets,” in *Proc. 30th Annu. Int. Conf. Mobile Computing and Networking (ACM MobiCom ’24)*, Washington, DC, USA, Nov. 2024, pp. 2120–2127, doi: 10.1145/3636534.3696215.
[9] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numerische Mathematik*, vol. 1, pp. 269–271, 1959, doi: 10.1007/BF01386390.