Integration of Real-Time Video Communication and Semantic Segmentation over the Internet Toward Dynamic Occlusion Handling in Mobile Mixed Reality for Landscape Simulation

○ Daiki KIDO[*1], Tomohiro FUKUDA[*2] and Nobuyoshi YABUKI[*3]

*1 Grad. Student, Div. of Sustainable Energy and Environmental Engineering, Grad. School of Engineering, Osaka University.
*2 Assoc. Prof., Div. of Sustainable Energy and Environmental Engineering, Grad. School of Engineering, Osaka University, Ph.D.
*3 Prof., Div. of Sustainable Energy and Environmental Engineering, Grad. School of Engineering, Osaka University, Ph.D.

Keywords: Real-Time video communication; semantic segmentation; Internet; mobile Mixed Reality; dynamic occlusion handling; landscape simulation.

1. Introduction

To form a good landscape, it is essential to discuss the project actively among stakeholders. Stakeholders consist of experts such as architects and government, and non-experts such as residents. Then it is difficult for stakeholders, especially the non-experts, to imagine the planned landscape, especially a three-dimensional (3D) object that has not existed yet. To facilitate the lively discussion, it is effective to visualize the planned landscape in the planning and design phase.

Recently, the use of mixed reality (MR) has attracted attention in the architecture, engineering and construction (AEC) field. MR is a technology that merges the real and virtual worlds [1)], and MR can produce a realistic landscape simulation by overlaying a 3D computer graphic (CG) model of a new building on a camera view of the planning site.

One of the technical challenges in MR is the occlusion problem, which occurs when virtual objects hide physical objects that should be rendered in front of virtual objects. As MR is realized by overlaying 3DCG models on a camera view, it is difficult to render the correct order of the real and virtual objects from the line-of-sight. Incorrect occlusion may cause user confusion related to the depth perception of the user. Properly occlusion handling is an essential issue for the spatial understanding between the real and virtual objects.

Previous studies of occlusion handling methods have been divided into three categories: model-based method, depth-based method and contour-based method.

Model-based methods create a 3DCG model of the occlusion target (occlusion model) in pre-processing and compare the depth of the virtual objects with the occlusion model to handle occlusion [2)]. In these methods, if the occlusion targets move or change its shape over time such as vegetation, it is difficult to handle occlusion because these methods need to create an occlusion model in pre-processing.

Depth-based methods acquire the depth information of the occlusion target from a 3D sensing camera in real-time such as RGB-D camera or stereo camera, and compare the depth of the virtual objects with the acquired depth information to handle occlusion [3), 4)]. These methods can handle occlusion dynamically in real-time, however have a limitation of the distance to obtain depth information and are unsuitable for a large-scale landscape simulation outdoor.

Contour-based methods detect and track the silhouette of the occlusion target and handle occlusion. In our previous study, we developed a MR system for landscape design simulation that performs dynamic occlusion handling using real-time semantic segmentation based on deep learning [5)] (Figure 1). Semantic segmentation is a technique to link each pixel in an image to a class label. Since real-time semantic segmentation processing involves heavy processing and needs a high-end computer, we developed a system in which client device transfers image frames to a server, which performs semantic segmentation processing on those frames and sends processed frames back to the client. To realize the client-server communication in this system, a web application was developed and utilized. However, the client-server communication based on a web application is not a versatile method in terms of computer security. It is necessary to access the server computer for the use of the web application. If the web application was deployed to access from everywhere, we have to manage and secure the server computer and it takes a lot of care and cost.

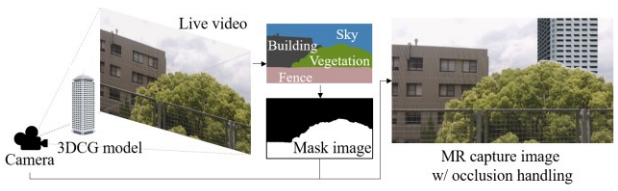

Figure 1. MR w/ occlusion using semantic segmentation [5)]

Then, we focused on real-time video communication technology over the Internet to conduct client-server communication everywhere without deploying the web application. When we use this video communication technology, it is not necessary to manage the server computer in terms of computer security and possible to use this MR system everywhere. Therefore, in our study, we aim to integrate the real-time video communication and semantic segmentation processing over the Internet based on deep learning toward dynamic occlusion handling in mobile MR for landscape simulation.

2. Proposed system

2.1. SYSTEM OVERVIEW

The overview of our proposed system is shown in Figure 2. Considering that the MR system for landscape simulation was developed in a game engine in our previous study [5], we integrated a game engine and video communication technology.

2.2. REAL-TIME VIDEO COMMUNICATION

As explained in chapter 1, we focused on real-time video communication technology to realize client-server communication for real-time semantic segmentation processing in mobile MR. To realize real-time video communication, we adopted WebRTC (Web Real-Time Communications) [6], which enables web applications and sites to capture and optionally stream audio and/or video media between browsers in real-time. WebRTC consists of several interrelated application programming interfaces (APIs) and protocols which work together to achieve real-time communication.

In our system, the packages of WebRTC were installed to both a client device and a server computer. When the communication with WebRTC is started, peer to peer (P2P) connection between the client device and the server computer needs to be established. At the timing of establishing a P2P connection, the stream data that is to be transferred should be also specified. Then, a client device can specify the image frames acquired from a webcam connected to a client device as the stream data, however a server computer doesn't have any data to transfer. Therefore a canvas element that has no data was specified as the stream data in a server computer at first. After the P2P connection is established, processed frames were copied to the canvas element that was specified as the stream data and transferred to the client device (Figure 3).

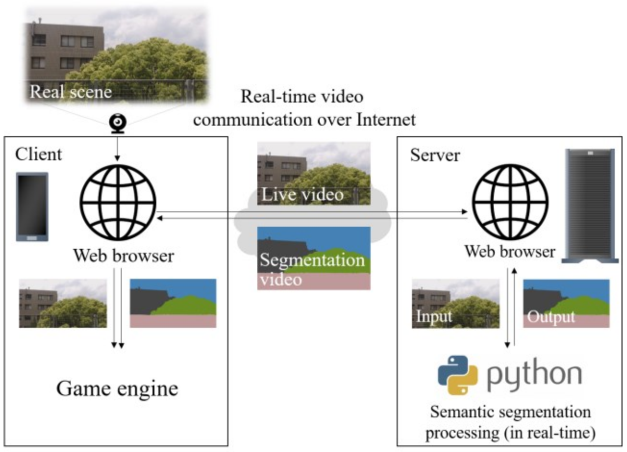

Figure 2. System overview.

2.3. BROWSER-PYTHON COMMUNICATION (SERVER)

A server computer performs semantic segmentation processing on the acquired webcam frames. However, the webcam frames were acquired on the web browser written in JavaScript and semantic segmentation processing was performed with python programming language. Thus in our system, a JavaScript-Python communication was implemented

In JavaScript, Ajax (asynchronous JavaScript and XML) was used for JavaScript-Python communication. With Ajax, web applications can send and receive data from a server asynchronously without reloading the whole web page. In our system, webcam frames were transferred to python with Ajax.

In python, flask was used for JavaScript-Python communication, which is a micro web application framework written in python. In our system, webcam frames were received with flask and performed semantic segmentation processing, and sended processed frames back to JavaScript. The communication flow between JavaScript and python is shown in Figure 4.

2.4. BROWSER-GAME ENGINE COMMUNICATION (CLIENT)

Webcam frames and semantic segmentation images in the web browser need to be transferred to a game engine in the client device to develop a MR system. Since it is difficult to send those images from JavaScript to game engine directly, we developed an intermediate server at the localhost in client device using flask.

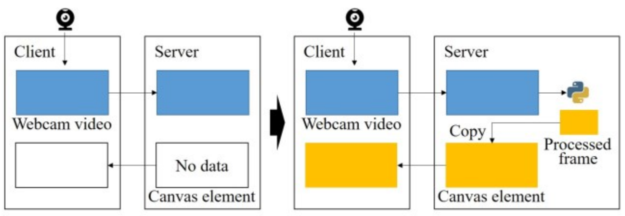

Figure 3. WebRTC communication in our system (Left: At the timing of P2P connection, Right: After the P2P connection).

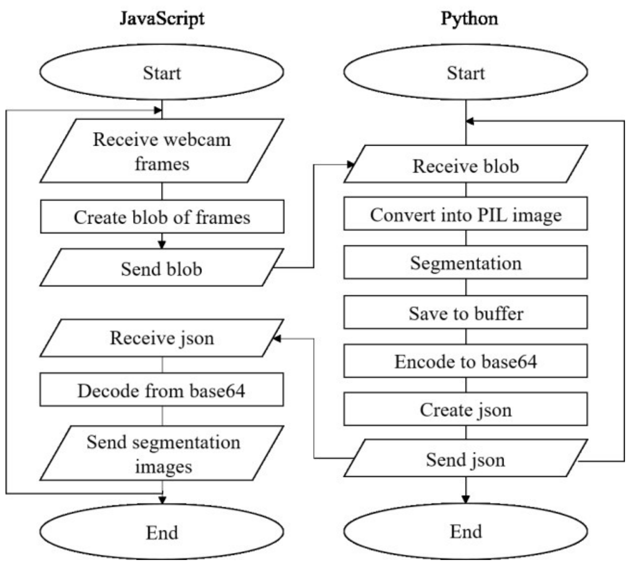

Figure 4. Communication flow between JavaScript and python.

In JavaScript, we adopted Ajax as explained in 2.3 and transferred the webcam frames and semantic segmentation images to the intermediate server.

In a game engine, GET method in HTTP communication was used to request the data and the game engine received the webcam frames and semantic segmentation images from the intermediate server (Figure 5).

3. Verification experiment

A verification experiment was conducted to evaluate the developed system. We adopted ICNet [7] as a semantic segmentation method, cityscapes [8] as a dataset for ICNet based on our previous study [5], and Unity as a game engine. We used a WebRTC communication platform provided by Skyway [9].

We compared the accuracy of the semantic segmentation processing and processing speed in Unity between the case that the communication was over the local area network (LAN) and over the Internet. In the LAN communication, Unity and python to perform semantic segmentation were directly communicated without video communication based on our previous study [5]. The target site of semantic segmentation was selected the same site whose view from a camera consists of vegetation, building and fence.

In the accuracy test, 3 frames of each communication were selected and IoU (Intersection over Union) was calculated, which is the index to measure the accuracy of image segmentation. IoU was calculated with the ground-truth annotation image and predicted segmentation image (Figure 6).

In the client-server communication, the server computer was connected to LAN/Internet with a cable and the client device was connected to LAN/Internet with wi-fi. The communication speed between client-server over the LAN was measured with the ping command, and the communication speed on the Internet was measured using a web site [10]. Measured communication speed was shown in Table 1. We used a surface pro 6 tablet computer with Intel Core i5-8250U of CPU, 8GB of RAM, and Intel UHD Graphics 620 of GPU as a client device and video camera, and a desktop computer with Intel Core i7-8700K of CPU, 32GB of RAM, and NVIDIA GeForce GTX 1080Ti 11GB of GPU as a server computer. The client device was panned to acquire the video (Figure 7). The resolution of the acquired video was 1024 × 576 pixels. Verification results are shown in Figure 8, 9 and Table 2.

In this verification, we confirmed that the real-time video communication over the Internet and semantic segmentation processing were integrated and video frames and semantic segmentation images were displayed in Unity.

As Figure 8, 9 and Table 2 show, the accuracy of semantic segmentation over the Internet was lower than over the LAN. It is considered that this lower accuracy over the Internet was caused by the bitrate of the video communication with WebRTC. Bitrate is the number of bits that are conveyed or processed per unit of time. The transferred video frames were compressed and the data size became smaller because of the low bitrate of video communication. Thus, it is necessary to improve the bitrate for

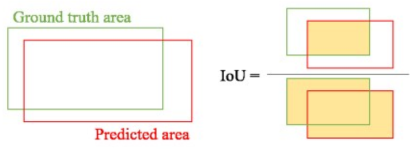

Figure 6. IoU (Intersection over Union) calculation.

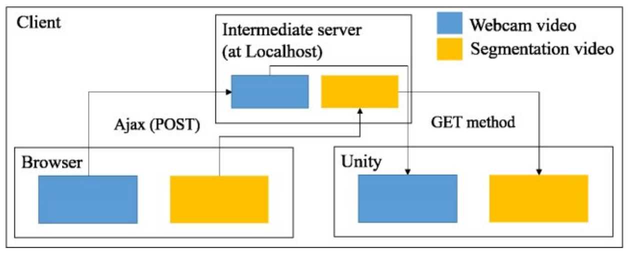

Figure 5. Communication between JavaScript and game engine.

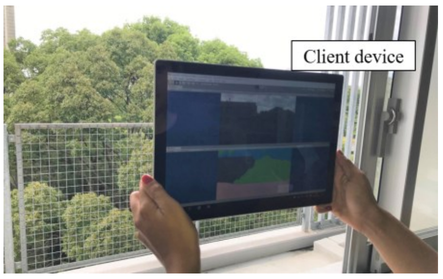

Figure 7. Scene of verification experiment.

Table 1. Measured communication speed.

	Communication	Speed [Mbps]
LAN	Client-Server	19.2
Internet	Brower(cli.)-Server[10)	Up: 113.8, Down: 118.7
	Browser(ser.)-Server[10)	Up: 626.5, Down: 625.5

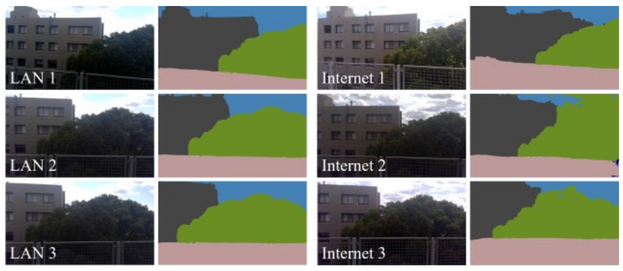

Figure 8. Verification results.

more accurate segmentation.

As Table 2 shows, processing speed in Unity over the Internet was much faster than over the LAN. However, though client-server communication over the Internet for uploading video frames and downloading segmentation images in client device was about 25 fps (frames per second), the processing speed of intermediate server-Unity communication in client device was about 5 fps. It was revealed that the communication speed between the intermediate server and Unity in the client device should be improved.

With this system, the camera view can be segmented and handled as one likes in a game engine. We can use this system not only for dynamic occlusion handling in MR for landscape simulation as explained in chapter 1, but also for visual environmental assessment in real-time; the ratio of each object for example.

Table 2. Processing speed in Unity.

Communication	Processing speed [fps]
LAN	6~7
Internet	15~16

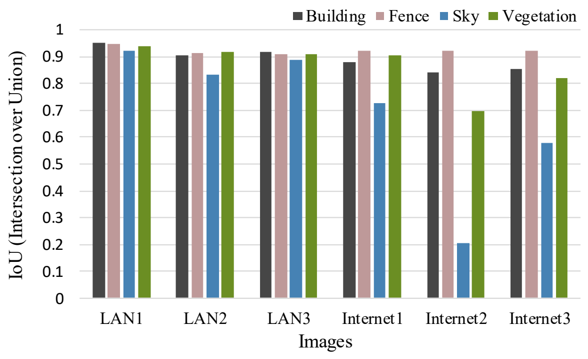

Figure 9. IoU of semantic segmentation images.

4. Conclusions and future work

In this study, we integrated the real-time video communication and semantic segmentation processing based on deep learning over the Internet toward dynamic occlusion handling in mobile MR for landscape simulation. We conducted a verification experiment, and we confirmed that real-time video communication over the Internet and a game engine were integrated.

In the future works, the bitrate of video communication and the communication between the intermediate server and a game engine in client device should be improved. And this system should be integrated with a MR system for dynamic occlusion handling.

Acknowledgements

This research has been partly supported by JSPS KAKENHI Grant Number JP19K12681.

References

1) Milgram, P. and Kishino, F.: 1994, A Taxonomy of Mixed Reality Visual Displays, IEICE Transactions on Information Systems, E77-D, 12, 1321-1329.
2) Inoue, K., Fukuda, T., Cao, R., and Yabuki, N.: 2018, Tracking Robustness and Green View Index Estimation of Augmented and Diminished Reality for Environmental Design: PhotoAR+DR2017 project, Proceedings of the 23rd International Conference on Computer-Aided Architectural Design Research in Asia (CAADRIA 2018), 339-348.
3) Du, C., Chen, Y., Ye, M., and Ren, L.: 2016, Edge snapping-based depth enhancement for dynamic occlusion handling in augmented reality, In the 15th IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2016), 54-62.
4) Holynski, A. and Kopt, J.: 2018, Fast depth densification for occlusion-aware augmented reality, ACM Transactions on Graphics (TOG), 37 (6), 194, 1-11.
5) Kido, D., Fukuda, T., and Yabuki, N.: 2019, Development of a Semantic Segmentation System for Dynamic Occlusion Handling in Mixed Reality for Landscape Simulation, Proceedings of the 37th eCAADe and 23rd SIGraDi Conference, 1, 641-648
6) Jennings, C., Hardie, T., and Westerlund, M.: 2013, Real-time communications for the web, IEEE Communications Magazine, 51, 4, 20-26.
7) Zhao, H., Qi, X., Shen, A., Shi, J., and Jia, J.: 2018, ICNet for Real-Time Semantic Segmentation on High-Resolution Images, In Proceedings of European Conference on Computer Vision (ECCV 2018), 418-434.
8) Cordts, M., Omran, S., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B.: 2016, The cityscapes dataset for semantic urban scene understanding, In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016), 3213-3223.
9) Skyway – Enterprise Cloud WebRTC Platform – : <https://webrtc.ecl.ntt.com/> (accessed 3 October 2019).
10) Speedtest by Ookla –The Global Broadband Speed Test: < https://www.speedtest.net/> (accessed 3 October 2019).