Master's Thesis

Title

Implementation and Evaluation of 3D-Point Attribute Streaming for Networked Virtual Reality Services using Edge Computing

Supervisor
Associate Professor Shin'ichi Arakawa

Author
Yuto Uchida

February 2nd, 2023

Department of Information Networking
Graduate School of Information Science and Technology
Osaka University

Master's Thesis

Implementation and Evaluation of 3D-Point Attribute Streaming for Networked Virtual Reality Services using Edge Computing

Yuto Uchida

Abstract

In recent years, XR (X-Reality), including VR (Virtual Reality) and MR (Mixed Reality), has been spreading with lower prices and improved quality of devices. In addition to 2D media such as images and videos, 3D media is exchanged through information networks, and the proportion of 3D media is increasing. And a digital twin, which imitates objects in the real world as 3D objects on a computer, is expected to improve our experience. 3D media is often represented by a point cloud, and some software systems handles the data of point clouds through file input/output. If point cloud information can be transmitted and received in a streaming format, and if point cloud attribute information can also be provided in a streaming format, we can expect to develop a service that allows users to perceive information about objects in remote locations in real time and to interact with people and objects in remote locations. For the point cloud streaming with attribute information, it is necessary to determine the format and protocol for sending and receiving information. In this thesis, we extend the existing implementation on point cloud streaming such that an attribute of each point is also transferred. Instead of changing the data format of points, we preserve existing data format so as not to make a lot of changes on existing implementation, and our attribute data is injected into the end of original data. A set of APIs are prepared for the point attribute streaming. Then, we implement a networked VR application using the APIs to demonstrate the services using 3D-point-attribute streaming. Our application uses a probabilistic representation of plausibility of object on point clouds. The sender injects the probability of the plausibility on the point as point attribute, and the attribute is transferred on network. The receiver picks up the attribute information and when the probability of point is low, more accurate rendering

is performed on the point. In this study, we use a probabilistic field [1] representation as the result of information processing. Probabilistic field representation expresses the plausibility of object identification as a probability, which is calculated by supervised machine learning on the point cloud information. We construct an experimental environment in our laboratory, and confirmed that 3D-point-attribute streaming is performed.

Keywords

Point Cloud
3D Information
Point Attribute
Streaming
Virtual Reality
Edge Computing

Contents

List of Figures

List of Tables

1 Introduction

In recent years, XR (X-Reality), including VR (VirtualReality) and MR (MixedReality), has been rapidly gaining popularity due to the low cost and improved quality of the equipment [2, 3]. Along with this, 3D media have emerged as media flowing over the network, in addition to 2D media such as images and videos, and the proportion of 3D media is increasing every day. For example, museums provide an application that places 3D objects that imitate exhibits in a virtual space and allows users to look around the 3D map in VR [4]. Such a world that imitates the real world and reproduces it virtually as a 3D object on a computer is called a digital twin [5], and there are growing expectations for its use to improve user experience and simulation.

Currently, graphic data or point cloud is used as information data to create 3D objects. When drawing 3D objects using graphic data, a program to represent a place or a scene is programmed in advance, and a scene in which 3D objects of the colors and shapes specified by the program is displayed. The advantage of using graphic data is that details of 3D objects can be created. However, it cannot do anything other than the preprogrammed operation, and a new program needs to be developed for representing other type of scenes. Therefore, the use of graphic data is appropriate for services that handle static information, such as services that exhibit objects in museums, and for those with pre-determined programs, such as live concerts. On the other hand, point cloud is a set of vertex and color data of objects, which are surveyed by RGB-D cameras and 3D laser scanners. Since point cloud is a set of point information, a large amount of data is required to represent curves and irregularities on the surface of an object as a 3D object. However, when drawing a 3D object based on point cloud, a new 3D object can be drawn simply by replacing the data if a generic program for displaying points is created. Therefore, the use of point cloud is considered suitable for services in which users perceive information on real objects in remote locations in real time and interact with people and objects in remote locations.

Cloud computing [6] or edge computing [7] is used for services that provide remotely sensed and aggregated information to users (Figure1). When using cloud computing, you can take advantage of the cloud's massive computing power. When edge computing is used, information processing is performed on a server placed near the sensing device. Therefore,

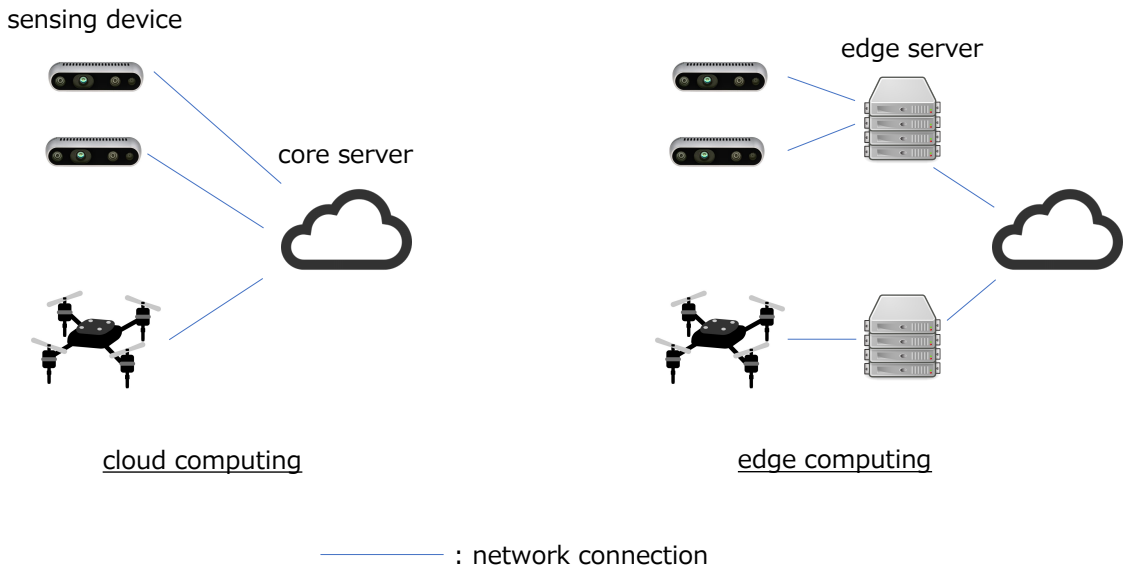

Figure 1: Cloud computing and edge computing

by aggregating or providing users with only specific information, it can be expected to reduce processing delays and network load compared to cloud computing, so this research is implemented using edge computing.

There are some efforts to use point cloud information to represent and utilize information on objects that exist in remote areas, as described in Reference [8, 9]. Reference [8] provides a semantic segmentation process using deep learning, and the results are used for forest management and other environmental conservation applications. In Reference [9], deep learning is used to detect damage to water pipes to identify areas in need of repair. In these services and studies, point cloud is transmitted and received mainly by files, and the results of information processing on point cloud (hereafter, attribute information) are also obtained in file format. However, exchanging information in file format involves disk input/output, which increases transmission delays and impairs the immediacy of information. Therefore, if point cloud can be transmitted and received in a streaming format, and if the attribute information of the point cloud can also be provided in a streaming format, it is expected to develop a service that allows users to perceive information on objects that exist in remote areas in real time and to interact with people and objects in remote areas. For example, by providing the results of object recognition by machine learning

together with point cloud information in a streaming format, point cloud information and its attribute information can be used for services that require real-time performance, such as automatic robot running and hazardous object detection.

Therefore, this study implements a streaming method that includes point cloud information and its attribute information. For point cloud-only streaming, the implementation is already available in Intel Realsense Software Development Kit [10]. With this, the point cloud could be sent to the end device (user) and attribute information at the end device. However, with this design, as the point cloud data volume and users increase, more data is exchanged over the network in a multiplicative manner, placing a greater load on the network bandwidth. Therefore, it is necessary to devise a streaming method for data that includes attribute information in addition to point clouds. When devising a new communication method, it is necessary to determine the format and the protocol for sending and receiving information for communication integrity. To add new data to an existing data communication system, one can either modify the existing data structure or keep the existing data structure and add new data. The former can improve application efficiency by compressing data and optimizing the data structure for subsequent processing, but it is expensive due to the wide variety of program modification points. The latter, on the other hand, is less expensive and easier to deal with implementation defects, making it a suitable method to use at the beginning of a project.

In this study, probabilistic field [1] is used as attribute information. The probability field expresses the plausibility as a probability after object identification, which is calculated by supervised machine learning on point cloud information. Specifically, it is a set of information that is a normalized representation of the predicted probability for the maximum likelihood label in the object recognition results for each object in the point cloud. By using a probabilistic field, it is expected that, for example, the amount of information processing for a point group with low probability can be increased to improve visibility at the terminal end, or that the amount of information processing can be reduced while maintaining visibility by thinning out points of an object composed of a point group with high probability.

2 Networked virtual reality services

2.1 Supposed virtual reality services

In recent years, XR (X-Reality), including VR (Virtual Reality) and MR (Mixed Reality), has spread rapidly due to the low cost and improved quality of the equipment [2,3]. With the advent of XR, it is becoming possible to experience the world created virtually on a computer as if it were the real world. For example, a world that imitates the real world and reproduces it virtually as a 3D object on a computer is called a digital twin [5], and there are growing expectations for its use to improve user experience and simulation.

3D models can be created using computer graphics or point cloud data, which is data obtained by surveying real space. This study deals with point cloud data that can be surveyed by RGB-D cameras and 3D laser scanners. Other sensing devices can be used that can measure the distance, or depth, from the device to the subject. A point cloud is three-dimensional information about an object calculated from this depth information, and by mapping the color information acquired by the camera to it, it is possible to represent a real object. Since point cloud data is obtained through surveying, it can be used to create 3D models that accurately reproduce the dimensions and structure of surveyed objects in a virtual space. However, because point cloud information is three-dimensional information, its large data volume is often problematic. For example, it is about 60 [MB] or more for a 3D object of a thing surveyed by a 3D laser scanner and about 300 [MB] or more for a 3D object of a landscape including mountains and lakes surveyed by an aerial laser.

The service envisioned in this study is a service that uses point cloud to analyze and represent information in remote areas. By using point cloud, real-world information can be faithfully reproduced, and attribute information analyzed from this information can be used in the service. For example, the aging of water pipes is analyzed [9] and individuals are identified based on the characteristics of surrounding environmental objects [11]. These services acquire point cloud from remotely located sensing devices, and it is transferred over the network. Considering the implementation of these functions using a network, in the case of Reference [9], the network model would look like Figure3 with cloud computing and Figure2 with edge computing. The respective images attached at the bottom of the figure are cited from Reference [9]. From left to right: real objects, point clouds, attribute

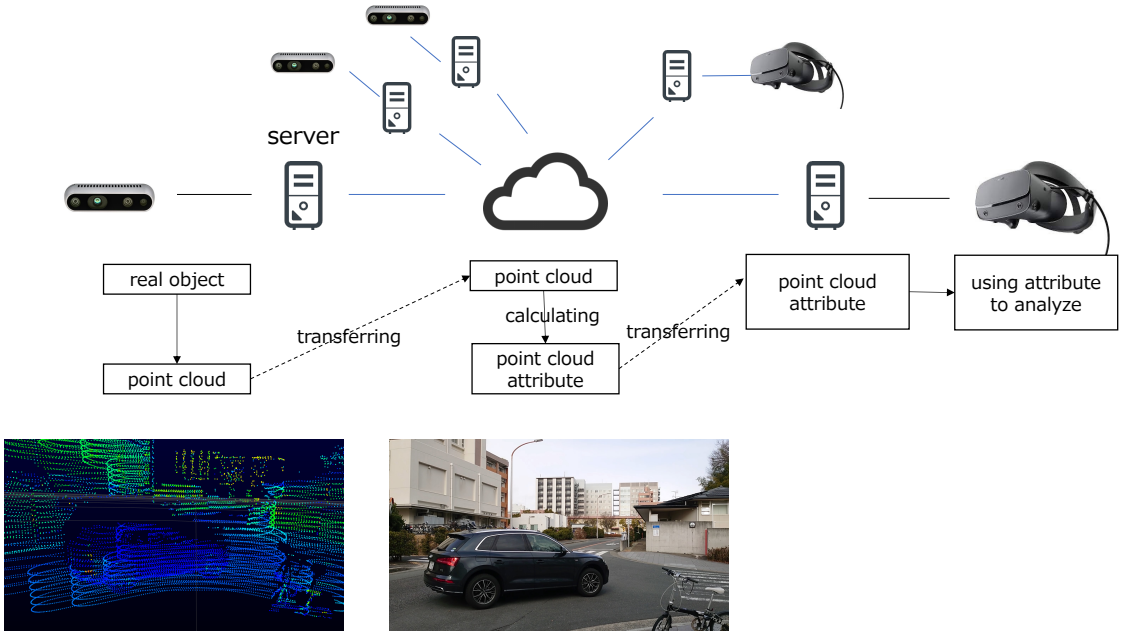

Figure 2: Network model (with cloud server)

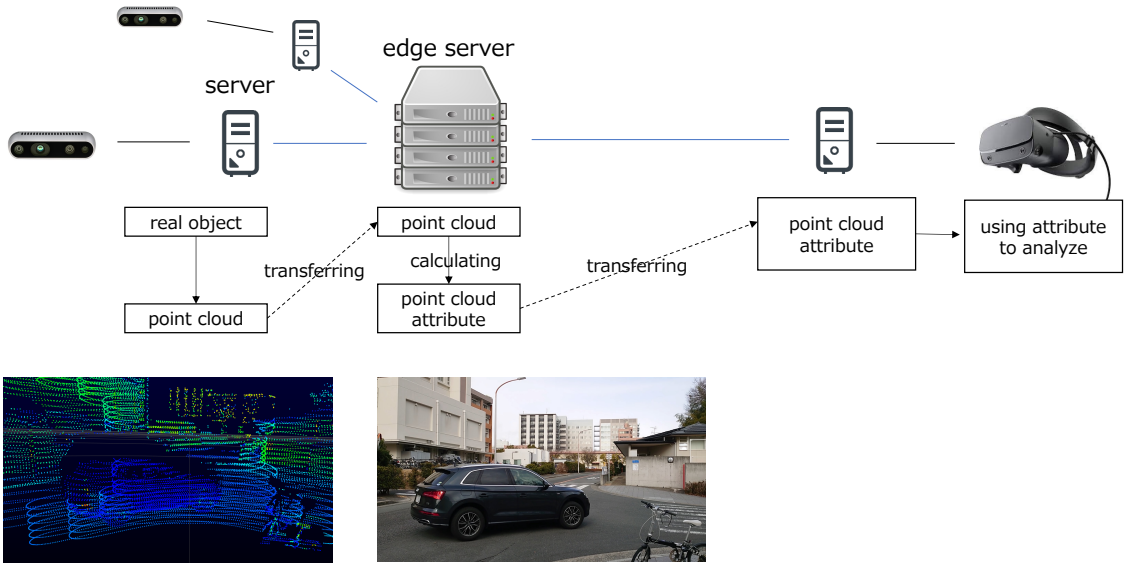

Figure 3: Network model (with edge server)

information, and analyzing using attribute information. First, a point cloud is surveyed from a real object, in this case a water pipe. This is done by the computer to which the sensing device is connected, both in the case of cloud computing and edge computing. This information is then transferred to the cloud and edge servers, respectively, and attribute

information is calculated. Cloud computing [6] can be used to take advantage of the large-scale computing capabilities of the cloud. With edge computing [7], point cloud data can be computed at the location where it was measured, thus delivering the data to the user with less network latency. The attribute information calculated by each server is then transferred together with the point cloud to the computer that performs the analysis. In Reference [9], the attribute information contributes to a surface reconstruction focused on the damage location and ultimately to an estimate of the damage volume.

2.2 Transferring point cloud data

Services using point clouds obtain point cloud information from sensing devices placed in remote locations. Possible methods are by file transfer and by streaming. Currently, most of applications on point cloud uses the file transfer.

2.2.1 Non streaming

Sending and receiving point cloud information by file transfer has the advantage of being easy to implement. This is because the file format defines the interfaces and protocols for transmission and reception, and there is no need to devise these on one's own. However, transferring information in file format is prone to large transmission delays because disk input/output is involved. Therefore, it should be used for services that do not require immediacy, such as identifying damaged areas of a structure. For example, the document [11] is implemented in a way that allows a point cloud in file format to be transferred over a network to receive a result file.

2.2.2 Streaming

Streaming point cloud makes it possible to use the information with low transmission delay. This makes it suitable for services such as digital twin, where the simulation results are to be reflected in the real world in real time. However, although an API (Application Programming Interface) for streaming only point cloud exists, but a framework for streaming including attribute information has not been devised. For example, a point cloud is streamed from a computer with a camera connected to another computer in the

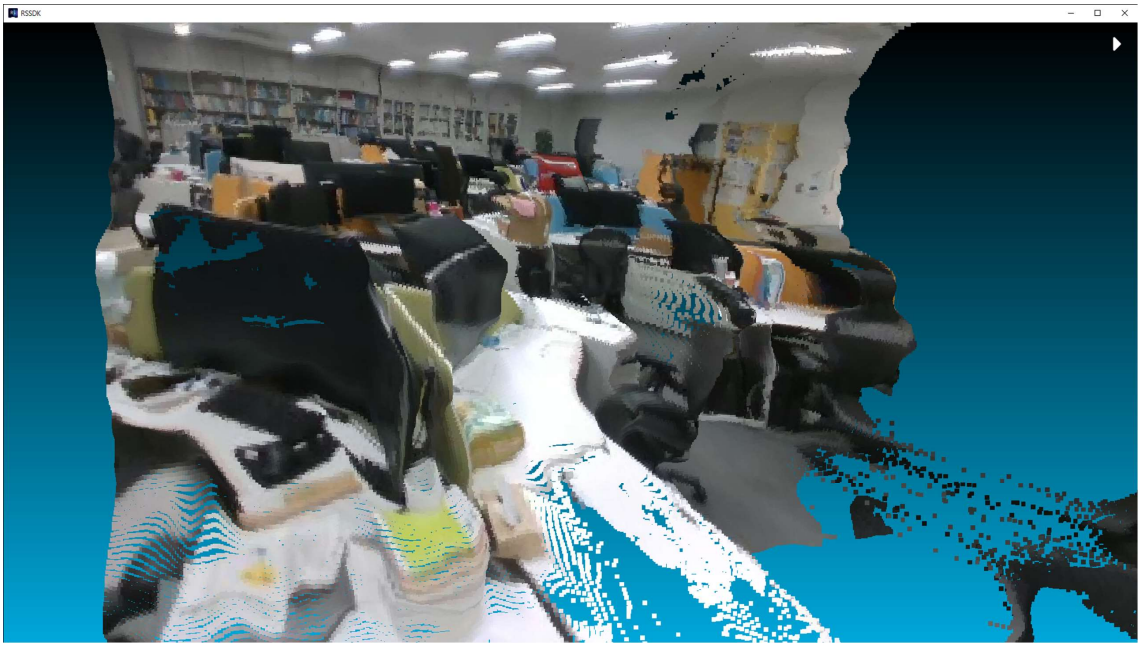

Figure 4: Example output of streaming point cloud

network, and the displayed result is shown in Figure4. In the figure, it can be verified that the point cloud data transferred over the network has been transferred correctly, while retaining the actual values. As mentioned earlier, since attribute information is necessary to apply point cloud to services, this study proposes a framework for streaming point cloud and attribute information.

2.3 Attribute data of point clouds

Services using point clouds are diversified by attribute information, which is the result of analyzing and statistically processing point cloud information. For example, there is probability field information to represent point cloud information probabilistically, object recognition result information for each point comprising the point cloud, or simply a flag indicating the location of damage. Therefore, attribute information is essential for the application of point cloud information to services. It is possible to send point cloud to

an end device and assign attribute information at the end device, but in this design, the more the point cloud data volume and users increase, the more data is exchanged over the network in a multiplicative manner and the greater the load on the network bandwidth. Therefore, it is necessary to devise a data streaming method that includes attribute information in addition to point cloud.

3 Implementation of streaming point clouds and attribute data

In this study, we will implement a streaming method that includes information associated with point clouds for application.

3.1 Existing approach for streaming point cloud

Point cloud only streaming has already been implemented in the open-sourced Intel Realsense Software Development Kit (librealsense) [10].

This section describes the implementation. The implementation in librealsense achieves point cloud streaming by using the live555 [12] functionality. The sensing device can be an Intel Realsense series RGB-D camera, and these can acquire depth (Figure5) and color (Figure6) information. The format of the depth information is a float type that stores the distance to be measured in each element of a quadratic array with a size equal to the resolution ($width * height$). Color information is transmitted as RGB image data. In other words, the raw data is divided into depth and color information. And the sending system transmits these separately to the receiving system. The receiving system then synchronizes the received depth and color information to compose the point cloud information. In the composition of point cloud information, the depth information is calculated and stored as a pair of coordinates x, y, z from the depth, and the color information is stored as it is. However, the depth and color information are captured by a separate camera in the sensing device, so the angles of view do not match. Therefore, in librealsense, in addition to the coordinate and color information, a correspondence table (UV map) between the coordinate and color information is added to the point cloud information.

3.1.1 Sending point clouds

It is implemented in the librealsense, which is OSS. Deliver depth or color information surveyed by the sensing device to the receiving system by the following method (Figure7). First, stablish a connection with the receiving application via RTSP (Real Time Streaming Protocol). Next, depth and color information, which is the source of point cloud information, is obtained from the sensing device. Data acquisition from the sensing device is

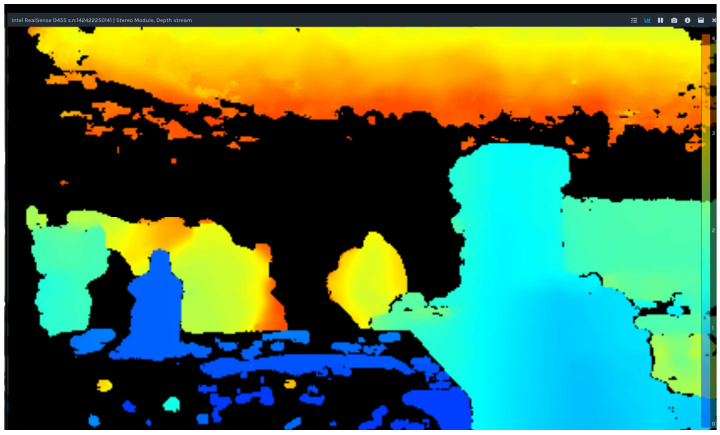

Figure 5: Example of depth frame

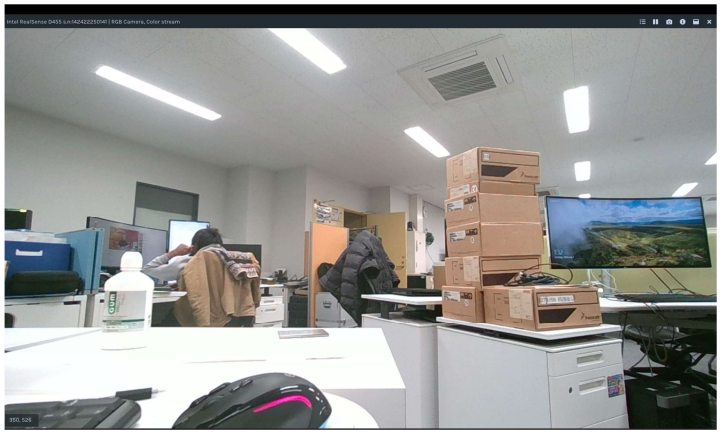

Figure 6: Example of color frame

performed using `poll_for_frame` function. This function is a queue function that accumulates the data surveyed by the sensing device. If there is no first data, it waits until timeout, and if there is, it returns it as a return value. The acquired data is moved to the Real-time Transport Protocol (RTP) message area. Then, when `afterGetting` function notifies the system that the message is ready, the color and depth information is delivered to the receiving system as a message by RTP, respectively. By repeating these steps,

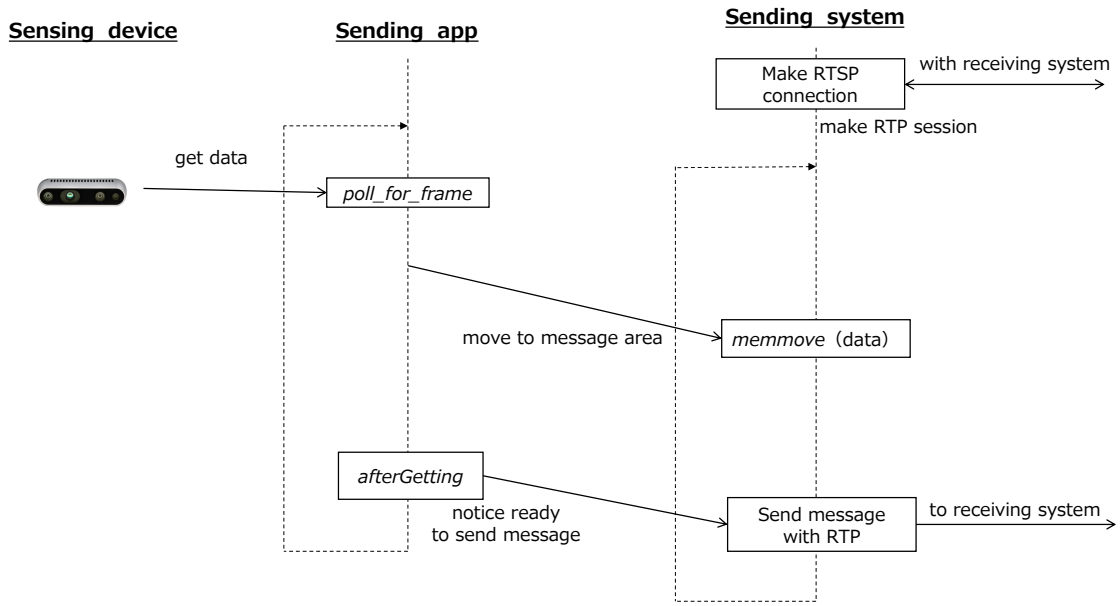

Figure 7: Sending point clouds (librealsense)

streaming distribution of point clouds is performed.

3.1.2 Receiving point clouds

As with the transmission, the reception is implemented in librealsense. The receiving system uses the following method to receive a point cloud (Figure8). Establish a connection with the sending application by RTSP. Through RTP, messages are received from the sending application and stored in a buffer as color or depth information. Next, the `sync` function is used to match the received color information (depth information) with the depth information (color information) that already exists in the buffer. If found as a result of matching, they are combined and entered into `rs2_wait_for_frame` queue as a single piece of data. If not found, it returns to receiving RTP messages again. The function `WaitForFrame` called in the application calls the system function `rs2_wait_for_frame` and receives a reference to the point cloud data frame. Since this is not a point cloud state, but is still depth and color information, it is cast to point cloud format by `process_frame` and used by the application. By repeating these steps, streaming distribution of point clouds is performed.

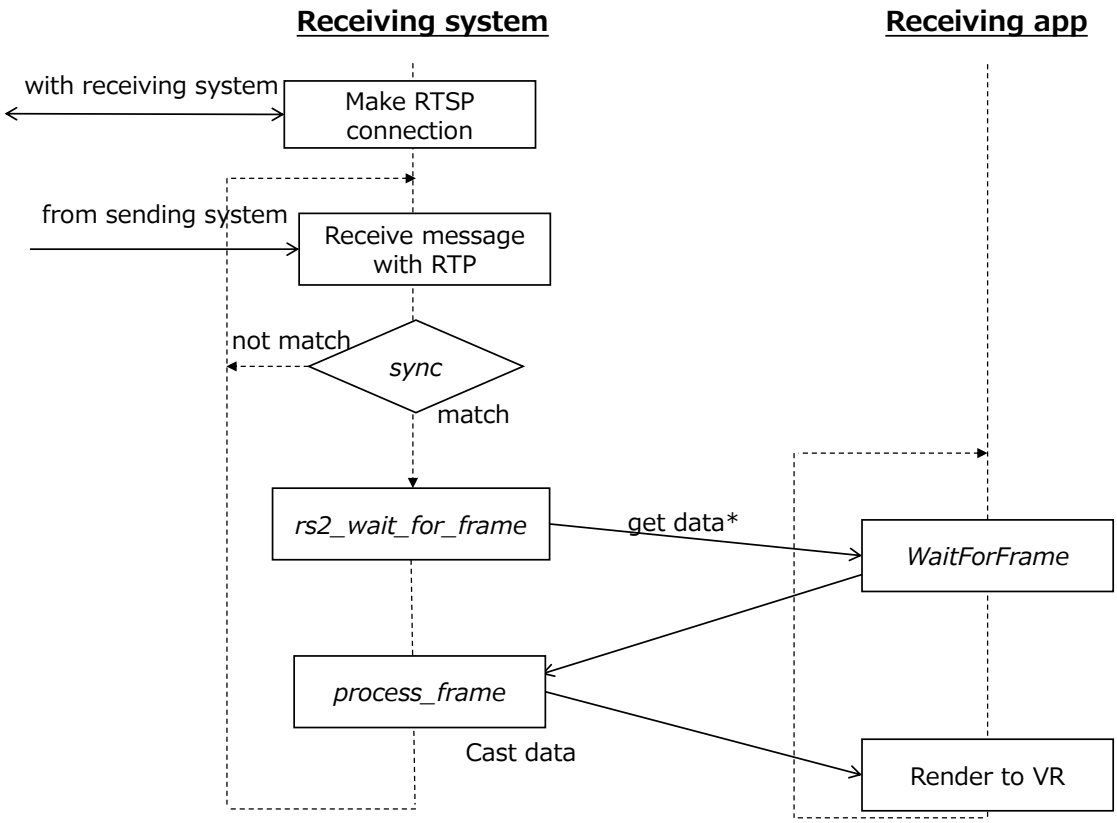

Figure 8: Receiving point clouds (librealsense)

3.1.3 Protcol and format for streaming point clouds

As mentioned above, RTSP is used to establish communication for streaming point clouds and RTP is used for streaming data [13]. RTSP is a protocol based on the Transmission Control Protocol (TCP), and RTP is a protocol based on the User Datagram Protocol (UDP). The point cloud streaming format using RTP is shown in Figure9, where the message, including the header data and main data, is divided into packets and streamed. Header data includes message size, timestamp, frame counter, sensing device FPS, and timestamp domain. The main data structure is in lz4 format for depth information and in jpeg format for color information, and the data is compressed in the respective compression formats. The packet contains the payload and RPT header into which the message is divided. The maximum length of the packet is 1494 [bytes], of which 42 [bytes] is the RTP header. In other words, the maximum payload length is 1452 [bytes].

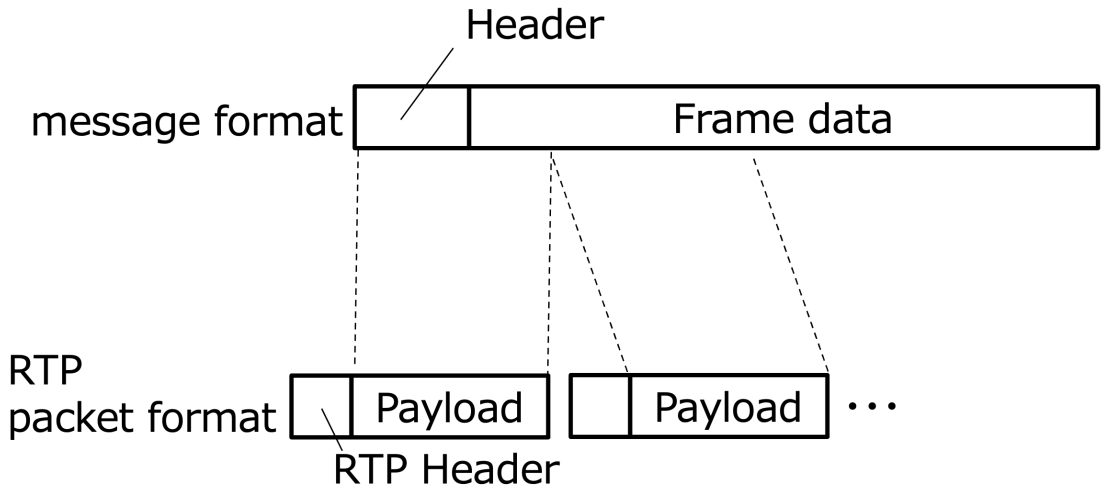

Figure 9: Streaming format for RTP (librealsense)

3.2 Approach to Point Attribute Streaming

This section describes a method for streaming point cloud information together with its attribute information (Point Attribute Streaming). As for protocols, as in Section3.1, RTSP is used for establishing communication and RTP is used for streaming data. The format is the one shown in Figure9 with attribute information added after the main data, which is divided into packets to be streamed by RTP. The implementation is also done by modifying some librealsense programs in which streaming of point cloud information is implemented. The implementation policy is to preserve the existing point cloud data structure and create messages in the form of attribute information attached to the existing data structure for interpretation by the receiver. One possible implementation method would be to modify the data structure of the existing point cloud and change the definition of the information that the points hold, but we will not take this approach. This is because there are various types of data structures for point clouds, ranging from simply arranging the coordinate information of points in order to sorting them in a tree format, and data compression and application processing are implemented for each of these types. For example, as mentioned earlier, the librealsense cited in this article takes the lz4 and jpeg formats for depth and color information, respectively, and each can take its own

compression method when transferred over the network. The method of changing the data structure of the point cloud and assigning attribute information would require changing not only the structure of the points, but also their implementation. In contrast, the approach taken in this study allows for the implementation of a streaming function that includes attribute information that hides the existing data structure and the implementation that manipulates it.

Another possible method would be to send attribute information separately, but this method was not used in this case. This is because a new synchronization system must also be developed by transmitting attribute information separately. By implementing a form of assignment to existing point cloud information, the synchronization of point cloud information and attribute information is achieved by taking advantage of existing synchronization system implementations.

3.3 Implementation of Point Attribute Streaming

The changes are made in the message creation and interpretation portions. In the interpretation part, the attribute information is stored behind the point cloud information when the point cloud information is stored in the main memory, and changes are made so that the attribute information can be read from a reference to the point cloud information.

On the sending side, an API function `rs2_add_prob` was implemented to attach attribute information to messages (Program1).

Program 1: `rs2_add_prob`

```
int fFrameSize;
void rs2_add_probs(byte* prob, byte* dest, int size){
        memmove(prob,dest,size);
        fFrameSize += size;
}
```

The input includes the address of the attribute information, the first address of the insertion destination, and the size. And `memmove` is performed using these. $fFrameSize$, a global variable means the sum total of the forwarded message.

Specific methods are listed below:

Step. 1 The reference to the first address of the point cloud information is added to the size of the reference (fixed $width * height * bpp$ in the format) to calculate the reference to the accuracy information.

Step. 2 Copies the specified size from the first address of main memory to the application area.

This is how the accuracy is extracted from the system area of main memory to the application area.

On the application side, these API functions are used. For Step. 1, an API function "`rs2_get_frame_probs`" is prepared. This function has as input a reference to a frame and an error handler, and returns a reference to attribute information. Therefore, the reference to the accuracy region obtained in Step. 1 and the size of the accuracy information (fixed in the same format as the point cloud information) are given as arguments to the API function that performs Step. 2 and called. Note that the API function, `Copy<T>()`, in the existing implementation can be used to perform Step. 2. Then, a function, `rs2_copy_prob` is implemented that takes a reference to attribute information as input and copies the attribute information to the application area (Program 2)

Program 2: `rs2_get_frame_probs`

```
T valiable;
void rs2_copy_probs(byte* prob){
        Copy<T>(prob, valiable, sizeof(byte) * 307200);
}
```

The transmitter/receiver system is first initiated by the application, which establishes communication via RTSP connections. The following describes the method of sending and receiving point cloud frame data and its attribute information. The operation of the transmitting side is described (Figure10). The sending application first acquires the point cloud frame data from the RGB-D camera. The RGB-D camera queues the surveyed point cloud frame data, so it is acquired from the queue. Upon receiving the point cloud frame data, the sending application uses it as input to calculate attribute information. When the point cloud frame data and attribute information are calculated, the contents are moved

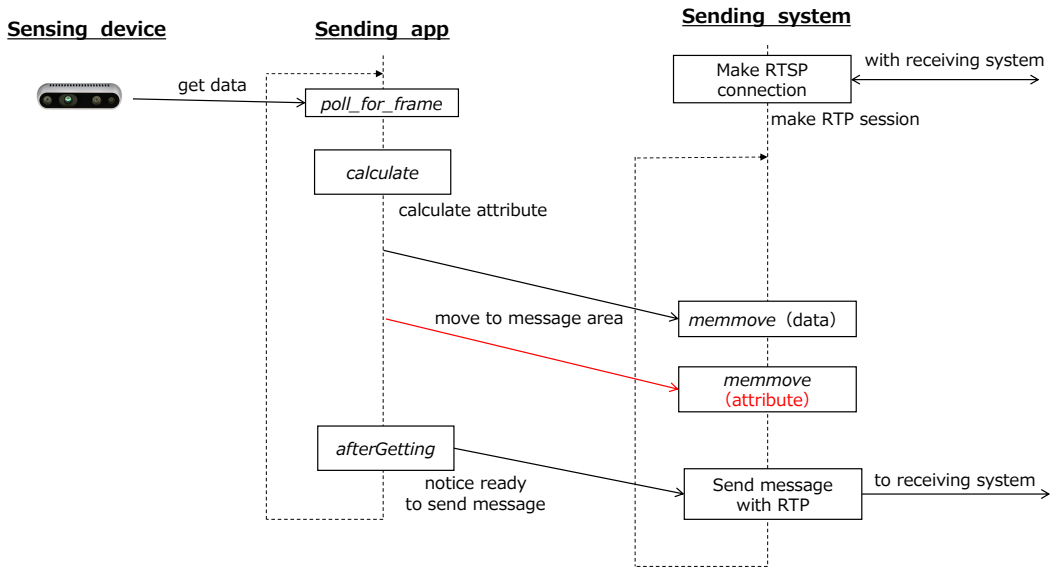

Figure 10: Sending point clouds (Point Attribute Streaming)

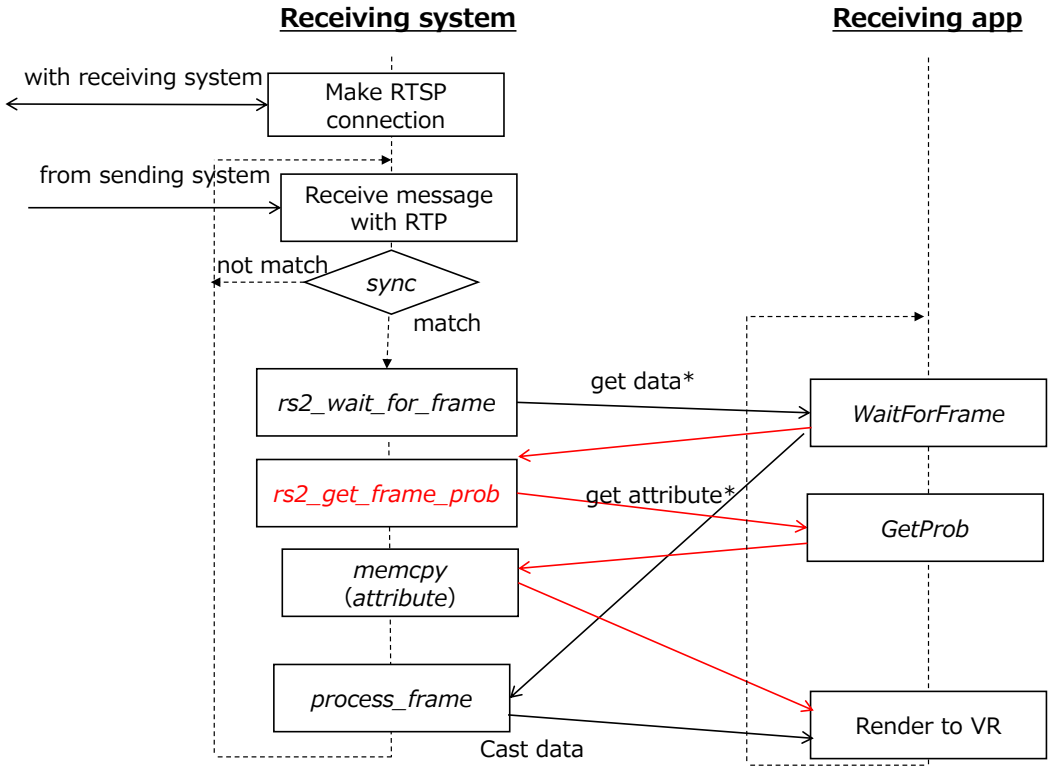

Figure 11: Receiving point clouds (Point Attribute Streaming)

to the message area of the sending system. Finally, the sending application notifies the sending system that it is ready to transmit. The sending application repeats the above process.

Next, the receiver system is described (Figure11). The receiving system receives messages from the sending system via RTP communication. Since the message is divided into packets and transferred, it waits for `rs2_wait_for_frame` to complete receiving the entire message. `rs2_wait_for_frame` is a function that monitors the queue where point cloud frame data received by the receiving system is stored. Storing point cloud frame data in the queue is done in a separate thread that handles live555 data. In other words, when the receiving system completes the reception, the receiving application passes the reference as the return value of `rs2_wait_for_frame`. The NW forwarding system repeats the above process. The receiving application receives a reference to the attribute information from the reference to the point cloud frame data passed from the receiving system from `rs2_get_frame_prob`. Then, from the reference to the attribute information, the entire data is copied to the application area. Finally, the receiving application passes the reference to the received point cloud frame data to the system and casts it into the data format used by the application. The receiving application repeats the above process.

New features added in implementing Point Attribute Streaming are indicated in red. `rs2_wait_for_frame` is a function that monitors the queue where point cloud frame data received by the receiving system is stored. The storage of point cloud frame data in the queue is performed in a separate thread that handles live555.

Figure12 shows the packet received by Wireshark [14]. For easy identification, the attribute information is set as a sequence of ff as byte. The figure shows that the packet can be attached to an RTP message and sent/received as a packet.

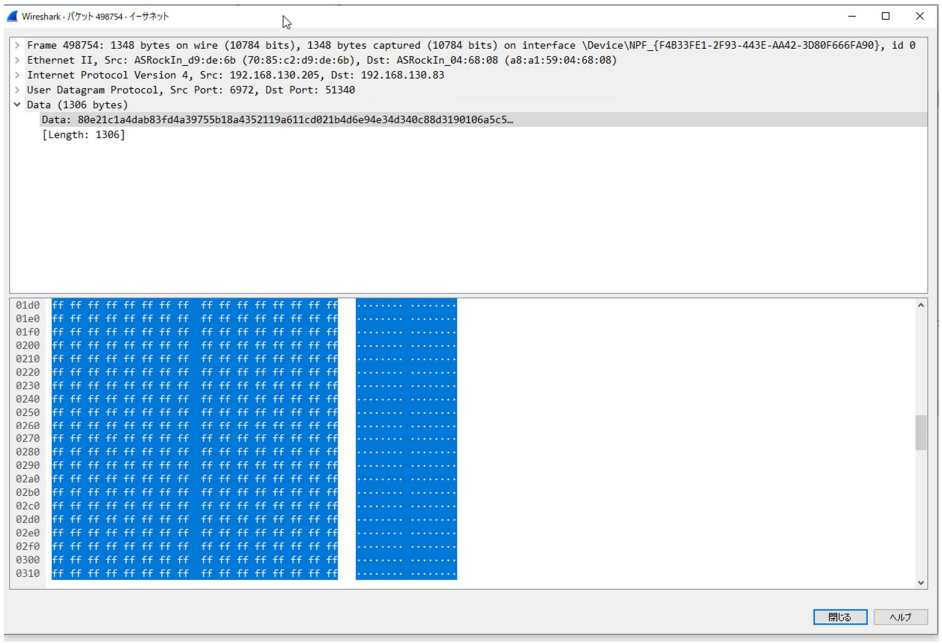

Figure 12: Observation results of packets containing attribute information by Wireshark

4 Applications using Point Attribute Streaming

This section describes an example application using the implementation described in Section 3. The application selectively corrects and renders point cloud information from the point cloud and probability information onto the virtual space. The probability information is the normalized predicted probability of the maximum likelihood label in the object recognition results for each object belonging to the point cloud calculated from the point cloud information, which corresponds to the attribute information.

4.1 Overview

This section provides an overview of the application to be implemented. The implementation is as follows:

1. Acquire point cloud information at the computer connected to the sensing device.
2. Transfer the acquired point cloud information to the edge server.
3. Perform object recognition on the edge server to calculate probability information.
4. Transfer the point cloud and probability information to the computer connected to the VR device.
5. Render the point cloud in virtual space using accuracy information on VR equipment.

Although it is possible to stream only the point cloud and calculate the probability information using a computer connected to the VR device, the design is based on placing an edge server because it requires a lot of computational resources to perform object recognition. Furthermore, assuming the case of streaming to multiple VR devices and the computers to which they are connected, sending point cloud information for learning object recognition to all computers is undesirable because it will occupy a lot of network bandwidth.

Figure13 shows the assumed experimental system in which the above flow is implemented. In practice, the application is implemented without an intervening edge server and communicates between the computer to which the sensing device is connected and the computer to which the VR device is connected.

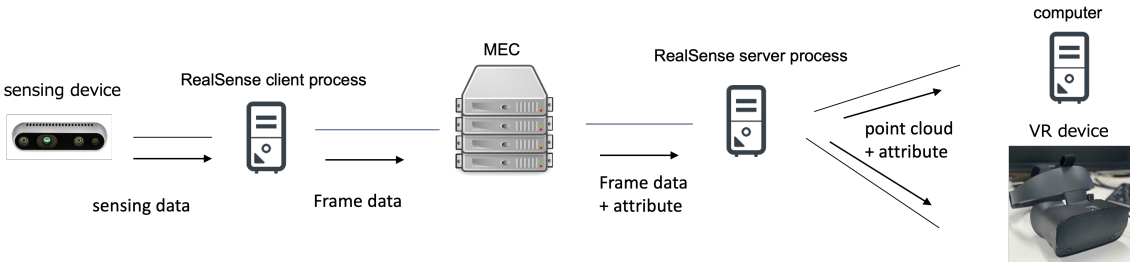

Figure 13: Assumed experimental system

4.2 Sending point clouds and probability data

The application for transmission also calculates probabilities. The probability calculation uses the algorithm implemented in Reference [15], and the function `processFrame` is called in the sending application program. The program for the sending application is coded in C++ and uses a Python program, a Python program, by reading "Python.h" in the package python3-dev. The included script has vertex and color information as input and probability field information as output. The implemented machine learning is performed on the vertex information using PyTorch [16]. The probability field information here refers to the likelihood of each of the 20 labels (Table1) prepared in supervised machine learning, where the label with the highest likelihood is called the inference result and the likelihood for that label is called the probability. The input receives the vertex information. Color information is also required for input, but since the vertex and color information is not synchronized at this point, all color information is assumed to be black (R = 0, G = 0, B = 0). This may result in an overall decrease in probability, which we are considering addressing by lowering the threshold for probability in the receiving application. ScanNet [17], a 3D indoor point cloud data set, is used for training.

By passing the calculated probability and the address of the message to be sent via RTP to `rs2_add_prob`, the application accomplishes the transmission of the probability.

4.3 Receiving and rendering point clouds and probability data

Receiving and rendering point cloud information to the VR device is implemented by modifying the program for Unity in librealsense. This section describes this application.

Table 1: Labels used in the learnig model

Label	Category
0	wall
1	floor
2	cabinet
3	bed
4	chair
5	sofa
6	table

Label	Category
7	door
8	window
9	bookshelf
10	picture
11	counter
12	desk
13	curtain

Label	Category
14	refrigerator
15	shower curtain
16	toilet
17	sink
18	bathtub
19	other furniture

Existing implementations involve rendering a point cloud by passing vertex, uvmap, and texture information to a renderer. In this study, we replace the rendering method and apply the method of corrective rendering only on a part of region.

The method to correct and represent point cloud information in virtual space using probability information is implemented in this program. In this method, based on the nature of the probability information, the correction display is limited to 3D objects with small probability to improve visibility.

Usually, when rendering point cloud information on a virtual space, a technique is used to represent each point as a circle (hereafter referred to as uncorrected rendering. See Figure14). However, because the drawn circles overlap each other in this drawing method, the contour and detail information of each point is lost. Therefore, a corrective rendering method was proposed [18]. Corrective rendering calculates the three-dimensional positioning of the point cloud from the user's perspective and draws each point in the point cloud as a parabolic surface (Figure16) that takes this into account (Figure 14) .

In this application, probability information is used to reduce the processing load because correction rendering is computationally expensive and causes delays when streaming is used. The overall processing load is reduced by using corrective rendering only for points that are considered to have low visibility in part due to the characteristics of the probability information.

This application receives point clouds and probability information from a receiving

Figure 14: Example of uncorrective rendering

Figure 15: Example of corrective rendering

system using Point Attribute Streaming, and renders the point clouds in a virtual space. Points with low probability are those determined to be difficult to discriminate in machine learning. Therefore, correction rendering is applied to improve visibility. Points with high probability can be considered highly visible, so they are represented using uncorrected rendering.

In the implemented method, drawing is achieved by passing vertex and color information to the renderer. Since the data format of the color information differs from that of existing methods, it is necessary to convert the format. In addition, the vertex information created from the data captured by the Realsense camera is upside down. Therefore, as

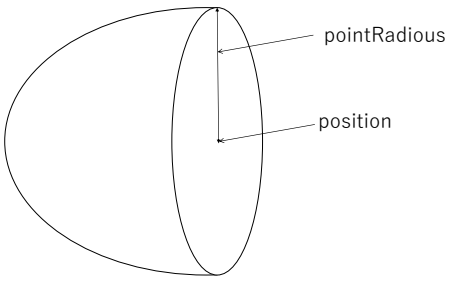

Figure 16: Paraboloid

in the librealsense rendering method, the y value is multiplied by -1 and is turned upside down again. The conversion of color information is explained below. We found that the texture information that can be obtained by the Realsense camera is in RGB24 format, where the RGB values are represented as a float type of 24 [bits] each, in the same order as the vertex information. Therefore, it was possible to provide the same sequence as the color information as input for the implementation method.

4.4 Demonstration of the implemented application

This section displays the output results of rendering by the actual running application and discusses the processing delays that occur. It is expected that the addition of attribute information will increase processing delays compared to streaming only point clouds, but specific values will be sought and discussed. In addition, the operating environment for the implemented application is shown below. The sensing device is an Intel Realsense D435, an RGB-D camera.

The configuration of the sending computer is shown in Table2 and that of the receiving computer in Table3.

As a subject, we will use the rest area of our room A610 (Figure17). We considered it appropriate because of the variety of objects arranged in both size and shape.

The final output is made to a VR device, the Oculus Rift S (Figure18).

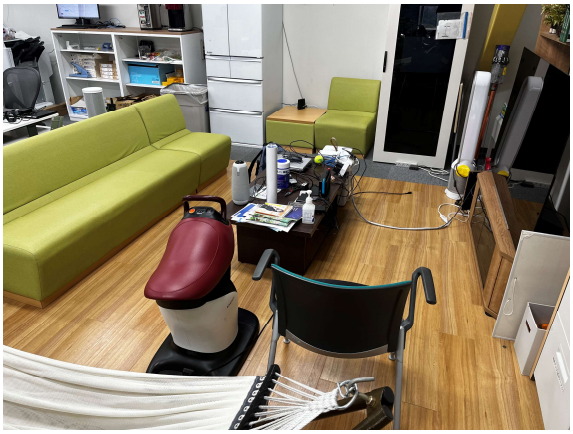

Figure 17: A610 rest space

Table 2: Specs of the computer which sends data

OS	Ubuntu 18.04
CPU	Intel Core i9-9900K 3.60GHz
RAM	64GB

Table 3: Specs of the computer which receives data

OS	Windows 10 pro 1903
CPU	AMD Ryzen Threadripper 3960X 24-Core Processor 3.80 GHz
GPU	Radeon RX 5500 XT
RAM	32.0 GB

Figure 18: Oculus Rift S

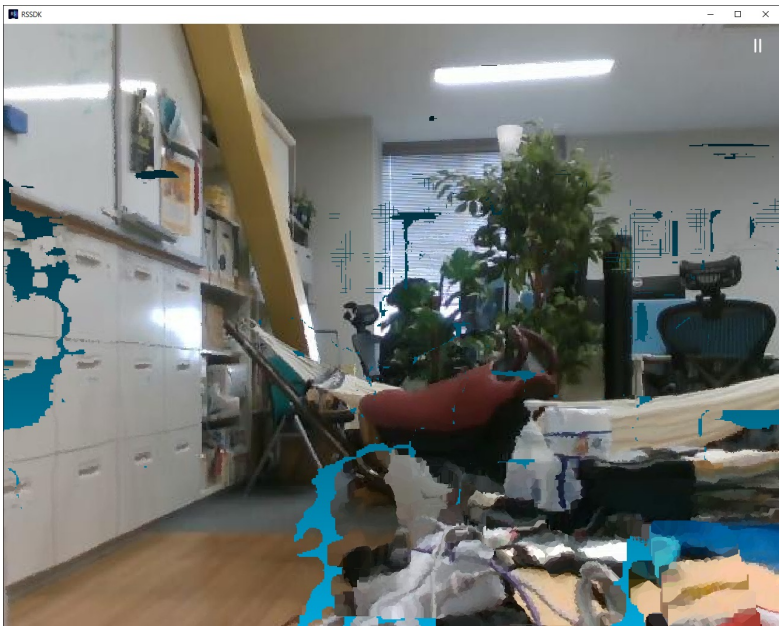

Figure 19: Corrective rendering output with the implemented application

The rendering obtained as a result of the execution of the implemented application is shown in Figure19.

The result of rendering without processing is shown in Figure20. Comparing these results, it can be confirmed that the power cord in the lower left corner in particular has poor visibility of details, which was recognized from the attribute information, and corrective rendering was applied to that area. Visibility was improved by corrective rendering, to the point where the cords were clearly visible crossing each other.

This indicates that the implemented streaming method can be applied to the application. The processing of attribute information calculation has a negative effect on the performance of the application. But this is not the issue. Because there are prospects for improvement with the implementation of MEC servers, and the processing power of computers is increasing daily.

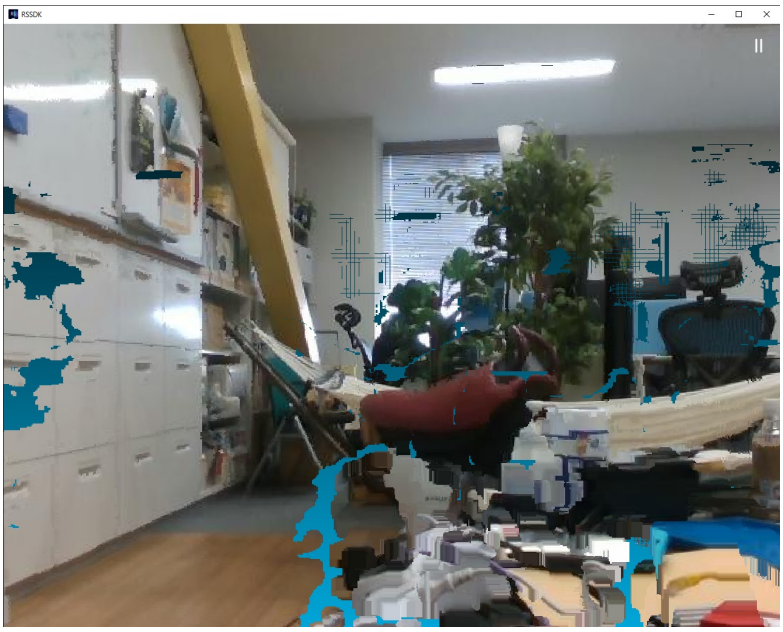

Figure 20: Uncorrective rendering output with the implemented application

5 Conclusion

In this study, we implemented a streaming method for point cloud information sensed remotely in real time and its attribute information, and an application using the method to the digital twin. Since the method for streaming point cloud information sensed remotely in real time already exists, we modified and implemented it so that it can stream not only point cloud information but also attribute information. When adding attribute information to the format, we did not change the structure of the point cloud information, but changed the streaming format. By the implementation of application, we indicate that remote information processing based on attribute information is possible. And we also observed performance degradation due to machine learning processes.

The calculation of attribute information is designed to be performed at the edge server, rather than at the end computer. This is done to consider the impact on computing resources and network bandwidth. But in practice, the function is included in transmission application.

So, as future work, the relay function is required when installing a MechServer. Another issue is to be able to decide whether to use attribute information when establishing a session.

Acknowledgments

In completing this research, Associate Professor Shin'ichi Arakawa of the Graduate School of Information Science and Technology, Osaka University, gave me concrete directions for our research through his precise suggestions and advice. I would like to express my sincere gratitude to him. Also, I would like to express our deepest gratitude to Professor Masayuki Murata of the Graduate School of Information Science and Technology, Osaka University, for his continuous and enthusiastic guidance and teaching.

I would also like to express my deepest gratitude to Associate Professor Yuichi Ohsita of the Institute for Open and Transdisciplinary Research Initiatives, Osaka University, Assistant Professor Daichi Kominami of the Graduate School of Information Sciences, Osaka University, Assistant Professor Tatsuya Otoshi of the Department of Business Administration, Graduate School of Economics, Osaka University, and Specially Appointed Assistant Professor Masaaki Yamauchi of the Graduate School of Information Science and Technology, Osaka University, for their continued guidance and support. Finally, I would like to express my gratitude to my family, friends, and laboratory for their support in various ways.

References

[1] Y. Ahmine, A. Dey, and A. I. Comport, "PNeRF: Probabilistic Neural Scene Representations for Uncertain 3D Visual Mapping," *arXiv preprint arXiv:2209.11677*, 2022.

[2] V. T. Minh, N. Katushin, and J. Pumwa, "Motion tracking glove for augmented reality and virtual reality," *Paladyn: Journal of Behavioral Robotics*, pp. 160 – 166, Mar. 2019.

[3] C. Trepkowski, D. Eibich, J. Maiero, A. Marquardt, E. Kruijff, and S. Feiner, "The effect of narrow field of view and information density on visual search performance in augmented reality," in *Proceedings of 2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, Mar. 2019, pp. 575–584.

[4] "MORI ART MUSEUM," https://www.mori.art.museum/jp/digital/03/, (Accessed on 01/04/2021).

[5] M. Singh, E. Fuenmayor, E. P. Hinchy, Y. Qiao, N. Murray, and D. Devine, "Digital twin: Origin to future," *Applied System Innovation*, vol. 4, no. 2, 2021. [Online]. Available: https://www.mdpi.com/2571-5577/4/2/36

[6] C. Gong, J. Liu, Q. Zhang, H. Chen, and Z. Gong, "The characteristics of cloud computing," in *Proceedings of the 2010 39th International Conference on Parallel Processing Workshops*, Sep. 2010, pp. 275–279.

[7] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[8] "Scanx," https://scanx.jp/, (Accessed on 01/03/2023).

[9] G. Pang, N. Wang, H. Fang, H. Liu, and F. Huang, "Study of damage quantification of concrete drainage pipes based on point cloud segmentation and reconstruction," *Buildings*, vol. 12, no. 2, 2022. [Online]. Available: https://www.mdpi.com/2075-5309/12/2/213

[10] "GitHub - IntelRealSense/librealsense: Intel RealSense SDK," https://github.com/IntelRealSense/librealsense, (Accessed on 12/19/2022).

[11] S. Yamada, H. Rizk, and H. Yamaguchi, "An accurate point cloud-based human identification using micro-size lidar," in *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, 2022, pp. 569–574.

[12] "Live555.com," http://www.live555.com/, (Accessed on 02/04/2023).

[13] W. Frank, "RTSP Live Streaming," EnGenius, Tech. Rep.

[14] "Wireshark," https://www.wireshark.org/, (Accessed on 08/23/2021).

[15] H. Sato, S. Arakawa, and M. Murata, "Proposal and evaluation of 3D-point object estimation method based on probability space representation ," in *Technical Committee on Communication Quality*, 2022, pp. pp.39–44(CQ).

[16] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[17] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3D reconstructions of indoor scenes," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 2432–2443.

[18] S. M. Fraiss, "Rendering large point clouds in Unity," Master's thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Sep. 2017.