Light VR client for point cloud navigation with 360° images

Clement Dluzniewski, Jérémie Le Garrec, Claude Andriot, Frédéric Noël

Light VR Client for Point Cloud Navigation with 360° Images

Clément Dluzniewski[1, 2], Jérémie Le Garrec[1], Claude Andriot[1], and Frédéric Noël[2]

[1]Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France
[2]G-SCOP : Univ. Grenoble Alpes, CNRS, Grenoble INP, G-SCOP, 38000, Grenoble, France

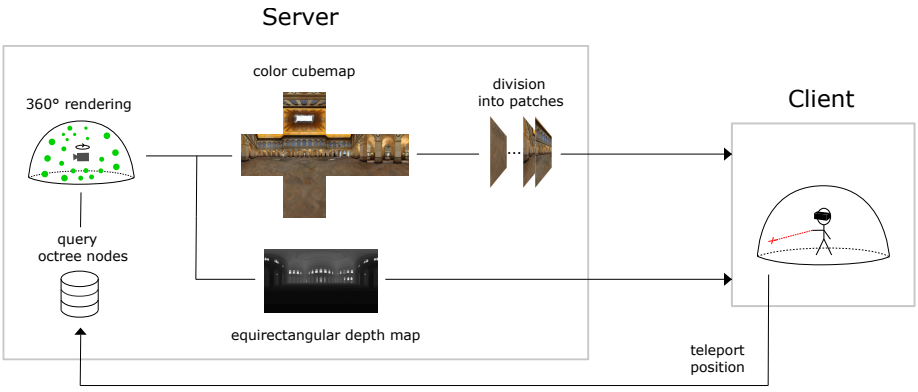

Figure 1: Our omnidirectional remote point cloud viewer. First, the server queries visible points from the camera position. Then, points are projected on a cubemap image and a omnidirectional depth map is created. The patches of the cubemap and the depth map are sent to the user. The user visualizes the omnidirectional image in an HMD and selects a teleportation destination. When the teleportation command is triggered, a request is sent to the server to retrieve the rendering of the point cloud at the pointed 3D coordinates. The camera is then moved at the desired position and the process is repeated.

Abstract

Since point clouds require a large amount of data to be visually pleasing, they tend to be voluminous. Hence, hardware with limited computational and memory capabilities may not be able to handle such large data structures. Here, we propose a light VR client to explore a static point cloud, stored in a remote server, through 360° images. The client visualizes in an HMD the omnidirectional rendering of the point cloud and moves to another position with a teleportation metaphor. The main advantage of our proposition is the ability to work on modest hardware without a continuous high bandwidth.

Index Terms: Computing methodologies—Computer graphics—Graphics systems and interfaces—Virtual reality; Computing methodologies—Computer graphics—Shape modeling—Point-based models;

1 Introduction

A point cloud is a simple data structure defined as an unorganized collection of 3D coordinates augmented with attributes such as color or normal. Unlike other representations, point clouds do not contain connectivity information between points, which easily leads to holes that degrade photorealism [6]. A structure with many points is then necessary to obtain a photorealistic rendering, which induces huge computational and memory requirements.

This paper proposes the idea of remote visualization of static point clouds with 360° images. The whole point cloud is stored on a server with high processing capacity, and an omnidirectional rendering of the point cloud [2] is created at coordinates requested by the user. Then, modest hardware can immersively explore the point cloud by rendering the omnidirectional image in an HMD. The user navigates in the environment from viewpoint to viewpoint with teleportation like in Google Street View or QuickTime VR. This discrete navigation scheme enables exploration without the need for continuous high bandwidth.

The low hardware requirements and the ability to work with unstable network make the system a good support for collaborative tele-immersion inside dense point clouds environments. For example, it may be used for construction project monitoring [4] to collaboratively review progress of a LIDAR scanned worksite with a simple web browser.

[1] firstname.lastname@cea.fr
[2] firstname.lastname@grenoble-inp.fr

2 Overview

Our goal was to create a client with metaphors close to traditional VR. To navigate, the user cast a laser to an area in the point cloud and is teleported at the selected position. Internally, a request is sent to the server and it responds by sending the omnidirectional rendering of the point cloud at the targeted position. The user navigates through the point cloud by successive teleportation. This mechanism was achieved by exploiting an omnidirectional depth map of the point cloud directly sent by the server. The whole communication scheme between the server and the client is illustrated Fig. 1.

We expect that the target-based travel metaphor is well-suited for this discrete communication scheme, as the server only sends information punctually and hence uses bandwidth sparingly.

2.1 Server

On the server-side, the point cloud is handled with a nested octree [5] to have a hierarchical level of detail that is appropriate to the viewing scale at the requested coordinates. If the user wants to visualize

closer a part of the point cloud, the LOD increased by going in-depth in the nested octree for the nodes representing this particular part.

A cubemap inspired projection [1] was chosen to transmit the omnidirectional rendering of the point cloud. The cubemap projection consists in projecting the pixels of the sphere on the faces of a circumscribed cube and arrange these faces (called patches) on a common image. Here, instead of combining these patches on the same image, the patches are sent independently to the user to progressively display the point cloud rendering instead of waiting for the complete result. Getting this projection is similar to project the visible points on 6 perspective cameras with different orientations, and a point is not visible by the user if it does not belong to any of the 6 camera frustums. To create an individual patch, the first step is to select the points in the octree to project on the camera. Thanks to the greater computational capability of the server, all the points in the camera frustum with a projection size greater than 1 pixel are selected. Then, these points are projected on a image using a splat rendering algorithm [3] to avoid sparse rendering. The selection and projection steps are repeated 6 times to get all the cubemap patches. An individual patch is directly sent to the user when its rendering is completed.

To be able to recover the real 3D points positions on the client-side, the server also sends a omnidirectional cubemap. Instead of progressively sending patches of depth map, the server sends it in only one image with an equirectangular projection [1] because manipulating a partial depth map is not interesting for user interactions.

2.2 VR Client

On the user-side, the cubemap patches of the point cloud are directly displayed in the HMD in the order they are received. After the reception of the depth map, the user casts a laser ray toward a destination like the traditional teleportation metaphor. With the help of the depth map, the global coordinates in the point cloud of the pointed pixel are computed. When the teleportation command is triggered, a request is sent to the server to retrieve the rendering at these global coordinates.

To help the user visualize the pointed position, an algorithm was developed to locally add a reticle on the virtual environment floor without server communication. The algorithm projects a 3D object on the omnidirectional image with respect to occlusions thanks to the depth map.

3 Evaluation

Because our system is designed to immersively explore point clouds without the need for continuous high bandwidth, its network usage is compared to two others methods. The first is a point streaming like Potree [3] consisting to directly transmit the points inside the client frustum organized in a octree. The second is a render streaming approach consisting to render the perspective view at client coordinates on the server and then transmit the rendering to the client. These methods have to send new information each time a rotation is performed, while with our approach the user can perform as many rotations as needed without additional requests. The bandwidth is evaluated on a controlled network with low latency by teleporting a client into 4 predefined positions and performing 2 horizontal rotations for each position in a point cloud of 730 millions points.

The measured bandwidth is shown in Fig. 2. The bandwidth is used continuously for both the streaming approaches. The point streaming server needs to send points to refine the rendering after a change in position and orientation, and the render streaming server needs to send the current video frame with a high frequency. The omnidirectional image server, in opposition to other methods, does not need to send data continuously. The graph shows punctual bandwidth usage with spikes corresponding to changes in position, without additional data received when rotations are processed, as

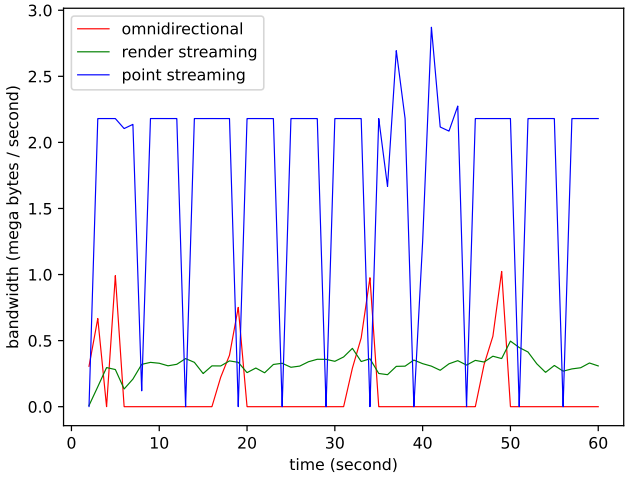

Figure 2: Comparison of the bandwidth usage for the tested approaches. Point streaming and render streaming use the bandwidth consistently while our omnidirectional approach uses the bandwidth punctually after requesting a new position.

expected. When adding up all the amount of data, we find that the point streaming server sends around 15 times more bytes than the omnidirectional image server, and the render streaming server sends around 2.5 times more bytes than the omnidirectional image server. This result shows that our method also uses fewer data for teleportation navigation, even when counting the depth map. Also, we compared the number of frames per second (fps) of the different approaches on the client-side. Measurements indicate that our method is about 1.5 faster than the render streaming approach and 2.5 faster than the point streaming approach, but all clients appear quite fast (fps values of all clients are above 200). Thus, these results provide evidences that our method is well-suited to navigate in a static point cloud with a teleportation metaphor.

Acknowledgments

This study takes place in Persyval Laboratory Of Excellence (French labex) and was supported by French governement funding managed by the National Research Agency under the Future Investments Program under grant ANR-21-ESRE-0030 / CONTINUUM.

References

[1] Z. Chen, Y. Li, and Y. Zhang. Recent advances in omnidirectional video coding for virtual reality: Projection and evaluation. *Signal Processing*, 146:66–78, May 2018. doi: 10.1016/j.sigpro.2018.01.004

[2] M. Comino, C. Andújar, A. Chica, and P. Brunet. Error-aware construction and rendering of multi-scan panoramas from massive point clouds. *Computer Vision and Image Understanding*, 157:43–54, Apr 2017. doi: 10.1016/j.cviu.2016.09.011

[3] M. Schütz. Potree: Rendering Large Point Clouds in Web Browsers. Master's thesis, TU Wien, Sep 2016.

[4] L. Waugh, B. Rausch, T. Engram, and F. Aziz. Inuvik Super School VR Documentation: Mid-Project Status. In *Cold Regions Engineering 2012*, p. 221–230, Aug 2012. doi: 10.1061/9780784412473.022

[5] M. Wimmer and C. Scheiblauer. Instant Points: Fast Rendering of Unprocessed Point Clouds. In *Proceedings of the 3rd Eurographics / IEEE VGTC conference on Point-Based Graphics*, SPBG'06, p. 129–137. Eurographics Association, Jul 2006.

[6] E. Zerman, C. Ozcinar, P. Gao, and A. Smolic. Textured Mesh vs Coloured Point Cloud: A Subjective Study for Volumetric Video Compression. In *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*, p. 1–6, May 2020. doi: 10.1109/QoMEX48832.2020.9123137