Pacific Graphics (2021)
M. Okabe, S. Lee, B. Wuensche, and S. Zollmann (Editors)

Poster

Volumetric Video Streaming Data Reduction Method Using Front-mesh 3D Data

X. Zhao[1] and T. Okuyama[1]

[1]Nippon Telegraph and Telephone Corporation, Japan

Abstract

Volumetric video contents are attracting much attention across various industries for their six-degrees-of-freedom (6DoF) viewing experience. However, in terms of streaming, volumetric video contents still present challenges such as high data volume and bandwidth consumption, which results in high stress on the network. To solve this issue, we propose a method using front-mesh 3D data to reduce the data size without affecting the visual quality much from a user's perspective. The proposed method also reduces decoding and import time on the client side, which enables faster playback of 3D data. We evaluated our method in terms of data reduction and computation complexity and conducted a qualitative analysis by comparing rendering results with reference data at different diagonal angles. Our method successfully reduces data volume and computation complexity with minimal visual quality loss.

CCS Concepts

• *Information systems* → *Multimedia streaming;* • *Computing methodologies* → *Image compression;*

1. Introduction

Volumetric video or free-viewpoint video is three-dimensional (3D) data that users can watch and interact with in true 6 degrees of freedom (6DoF) while wearing virtual reality (VR)/augmented reality (AR) headsets. The most common data formats for volumetric video are point clouds and textured meshes. However, these types of 3D data are still very large. Mesh with 25K vertices and 50K triangles requires approximately 4000KB per frame, and streaming this data with 30FPS requires around at least 1Gbps of bandwidth. For a video showing more than two people, the required bandwidth can easily pass 2Gbps. This bitrate requirement is far higher than that of the 2D video streaming method used on the market. To overcome this high bandwidth requirement and high decoding cost, we propose a "front-mesh" streaming method that divides 3D volume into multiple one-sided front-mesh data and dynamically selects visible mesh data on the basis of a user's virtual position in a 3D environment.

2. Proposed Method

Volumetric video is 3D data with a performer's 360 surrounding data. Users can view it from any arbitrary perspectives. However, when a user views 3D data on a display, not all the information is fully used. One such case is "back-side culling," where users can only see the front side, facing the user, while the rear side of that data is deleted and not rendered. Applying this when streaming volumetric video usually requires dynamic processing based on the user's viewing frustum. However, this requires constant high computation resources on a server and may not meet high-demand low-latency use-cases such as VR. To solve this issue, we propose a method called front-mesh to process back-side culling offline and create 3D data with only visible meshes. This data is generated in three steps. First, we propose a data division mechanism with predefined viewing angles. We place multiple virtual cameras evenly around the object. Here, as shown in Fig. 1(i), we set N virtual cameras, where each virtual camera is $V_n(n \in [1,N])$ placed in angle θ_n. In Fig. 1(i), we use eight directions ($N = 8$). Each virtual camera has its viewing region $\{R_n|\theta_n - \pi/N \leq R_n < \theta_n - \pi/N\}$. When the user is watching, his/her position angles are calculated relative to the object. Then, this angle is searched for through the viewing region R_n for matching angles, and once the match is found, the virtual camera V_n referenced from viewing region R_n is chosen. Second, we use a visible face selection method, as shown in Fig. 1(i). We select visible meshes for each V_n and eliminate the back-face of the data. Fig. 1(ii) shows an example of the generated front-mesh data. There will be N uniquely different front-mesh data for every frame of the volumetric video. Finally, we combine a compression method called Draco [Goo] to further reduce the size of the data. As a result of this operation, the data is made significantly smaller. Moreover, front-mesh data reduces the encoding and decoding computation complexity of Draco, which results in less processing latency.

3. Evaluation

We implemented the proposed front-mesh generation algorithm with 8 virtual cameras and applied on first 30 frames of 2 open source human volumetric videos "mattis" and "rafa"[ZOS]. The an-

DOI: 10.2312/pg.20211395

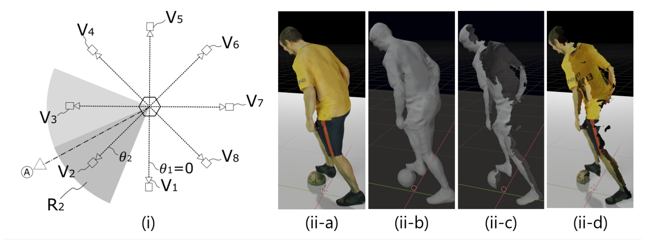


Figure 1: *(i) Virtual cameras V_n and viewing region R_n, and (ii) generated front-mesh data. Left two: original, Right two: front-mesh*

gle 0° is the viewpoint facing straight to the front side of the volumetric video. For compression, we used Draco [Goo] with compression level 10. The background 3D scene and front-mesh data were created on Blender (version 2.90) and its Python API. The rendering images used for quality evaluation are generated with EEVEE on a computer with i7-10750H CPU, 32GB memory, and RTX2080 GPU.

Data Size
The data size of "mattis" was 3101KB, and "rafa" was 3141KB on average. The proposed front-mesh data reduced "mattis" and "rafa" data to an average 53%-59% and 56%-61% from their original sizes, respectively. The proposed method reduced volumetric video data to more than 40% of the original volume. The conventional method compressed "mattis" data to 61KB whereas the proposed method compressed it to 39KB. Therefore, the proposed method reduced the size of "mattis" data by more than 35% compared with the conventional method.

Computation Complexity
First, we examine encoding and front-mesh generation tasks in a server. As in Fig. 2(a), the conventional compression method used 30 frames averaging 85.7 ms each whereas the proposed method took 67.5 ms. Thus, the proposed method shortened encoding time by a maximum 18 ms per frame. On the other hand, front-mesh creation took 53 ms. However, this can be omitted on streaming applications, since the process is done offline. Second, we see decoding and memory loading tasks in a client. As shown in Fig. 2(b), the conventional method took 25.3 ms whereas the proposed method took 18.4 ms. Thus, the proposed method shortens the decoding time by a maximum 7 ms per frame. On the other hand, memory loading took 730 ms for conventional 3D data and 360 ms for proposed front-mesh data. Therefore, the overall computation time for the proposed method was over 370 ms shorter than that for the conventional method.

Visual Quality
As shown in Fig. 1, the proposed method uses fixed angle front-mesh data. When users are inside the same angle range, the front-mesh data created with the same virtual camera is used. To evaluate the visual quality of this diagonal viewing, we used the first 30 frames of "mattis" front-mesh data created with camera angle 0° and filled with additional cameras with 10° angle intervals with front-mesh as the center and with three distance scales of 1, 2, and 3 m to simulate the diagonal viewing. We render 30 frames of the original 3D data, front-mesh data as video, and compared the

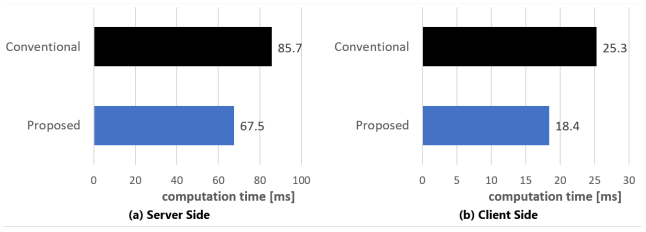
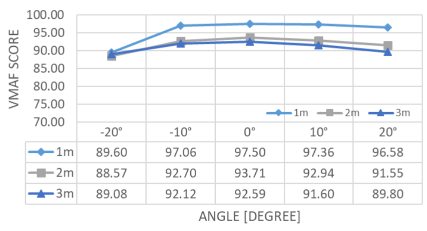


Figure 2: *(i) Encoding time in server. (ii) Decoding time in client device*



	-20°	-10°	0°	10°	20°
1m	89.60	97.06	97.50	97.36	96.58
2m	88.57	92.70	93.71	92.94	91.55
3m	89.08	92.12	92.59	91.60	89.80

Figure 3: *VMAF of the proposed method with various diagonal positions and distances*

two videos with two objective quality metrics: structure similarity (SSIM) [WBRS04] and video multi-method assessment fusion (VMAF) [Net]. The proposed method retains most of the required mesh data that needs to be rendered and scored 0.98 with SSIM even at the closest 1-m distance. For VMAF, as shown in Fig. 3, the proposed method has the highest quality at a close-up of 1 m where few back faces can be seen. The graph also showed that viewing angles close to the virtual camera angle 0° have higher quality but the changes are subtle at 10° difference.

4. Conclusion

We propose a method using front-mesh data to reduce data size and client computation cost. We evaluated our method in terms of data reduction and computation complexity and conducted a qualitative analysis by comparing rendering results with reference data. Our method successfully reduces data volume and computation complexity with minimal visual quality loss. For future work, we plan to investigate a streaming mechanism with front-mesh data over a network and conduct a subjective quality evaluation with users.

References

[Goo] GOOGLE. *Draco 3D Graphics Compression*. URL: `https://google.github.io/draco/` (visited on 01/27/2021).

[Net] NETFLIX. *GitHub - Netflix/vmaf: Perceptual video quality assessment based on multi-method fusion.* URL: `https://github.com/Netflix/vmaf` (visited on 08/03/2021).

[WBRS04] WANG, ZHOU, BOVIK, ALAN CONRAD, RAHIM SHEIKH, HAMID, and SIMONCELLI, EERO P. "Image Quality Assessment: From Error Visibility to Structural Similarity". *IEEE TRANSACTIONS ON IMAGE PROCESSING* 13.4 (2004).

[ZOS] ZERMAN, EMIN, OZCINAR, CAGRI, and SMOLIC, ALJOSA. *Textured Mesh vs Coloured Point Cloud: A Subjective Study for Volumetric Video Compression 2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*. ISBN: 9781728159652.