특집논문 (Special Paper)
방송공학회논문지 제29권 제5호, 2024년 9월 (JBE Vol.29, No.5, September 2024)
https://doi.org/10.5909/JBE.2024.29.5.616
ISSN 2287-9137 (Online) ISSN 1226-7953 (Print)

안드로이드 기반의 증강현실 서비스를 위한 포인트 클라우드 비디오 렌더러 설계 및 구현

김 민 석[a)], 이 수 연[a)], 김 아 영[a)], 안 은 빈[a)], 서 광 덕[a)‡]

Design and Implementation of Point Cloud Video Renderer for Android-Based AR Service

Min-suk Kim[a)], Su-yeon Lee[a)], Ayoung Kim[a)], Eun-bin An[a)], and Kwang-deok Seo[a)‡]

요 약

포인트 클라우드는 정밀한 세부 정보를 제공하여, 자율주행, VR, AR 등의 분야에서 높은 정밀도를 요구하는 응용 프로그램에 유리하다. 하지만, AR 및 모바일 환경에서 포인트 클라우드의 채택은 여러 기술적 문제로 제한되고 있다. 높은 계산 복잡도와 처리 능력, 배터리 소모 증가, 최적화된 렌더링 엔진의 부족, 데이터의 방대함 등이 주요 문제점이다. 본 논문에서는 이러한 문제를 확인하고 렌더러 성능 개선과 향후 연구 방향을 모색하고자 Unity와 Google AR Core를 사용하여 안드로이드 기반 AR 환경에서 포인트 클라우드 렌더러를 구현하였다. 특히 이 과정에서 PLY 파일 구조를 분석하고 데이터를 파싱하여 Unity에서 렌더링하는 방법을 설명하며, Android 14(API Level 34), RAM 4GB 환경에서의 실험 결과를 제시한다. 실험 결과, 모바일 환경에서의 처리 지연과 메모리 한계로 인한 문제점을 확인하였으며, 이를 해결하기 위한 추가 연구의 필요성을 제시한다.

Abstract

Point clouds are crucial for high-precision applications in fields like autonomous driving, robotics, VR, and AR. Despite their advantages, the widespread adoption of point clouds in AR and mobile environments faces various technical challenges. These include high computational complexity, increased processing power and battery consumption, lack of optimized rendering engines, and large data sizes. This paper identifies these issues and aims to improve renderer performance and explore future research directions by implementing a point cloud renderer in an Android-based AR environment using Unity and Google AR Core. We analyze the structure of PLY files, parse the data, and render it on top of Unity. Experimental results highlight issues such as processing delays and memory limitations. While the renderer performs well in a PC environment, mobile devices exhibit lower visual quality due to hardware limitations. Further research is needed to address these challenges, such as optimizing parsing code and downscaling point cloud data to reduce rendering delays and memory usage.

Keyword : Point Cloud, AR Service, Point Cloud Video Player

Ⅰ. 서 론

최근 포인트 클라우드는 3차원 데이터를 표현하는 유망한 방식으로 주목받고 있다. 포인트 클라우드는 3D 좌표계에서 점들의 집합으로 구성되며, 각 점은 객체의 표면이나 장면 내의 특정 위치를 샘플링한 것이다[1]. 이러한 표현 방식은 높은 수준의 세부 정보를 제공할 수 있어, 다양한 산업 분야에서 활용될 수 있으며 특히, 높은 정밀도가 요구되는 자율주행, 로보틱스, 가상 현실(VR) 및 증강 현실(AR)과 같은 응용 분야에 유리하다[2].

이러한 포인트 클라우드의 장점에도 불구하고, 증강 현실 및 모바일 환경에서 포인트 클라우드의 광범위한 채택은 다음과 같은 여러 기술적 및 구조적 문제 때문에 여전히 제한적이다[3]. 첫째, 포인트 클라우드 렌더링의 계산 복잡도는 기존의 mesh와 같은 3D 모델에 비해 훨씬 높다. 이는 모바일 장치의 중요한 제약 사항인 처리 능력과 배터리 소모 증가를 초래한다. 포인트 클라우드 데이터를 실시간으로 렌더링하는 것은 고사양의 하드웨어를 요구하며, 모바일 환경에서는 충족시키기 어려운 경우가 많다.

둘째, 최적화된 포인트 클라우드 렌더링 엔진의 부족은 개발자들이 다양한 하드웨어 구성에서 원활하게 실행될 수 있는 효율적인 솔루션을 구현하는 데 어려움을 겪게 한다. 현재의 증강 현실 플랫폼은 주로 mesh 기반 렌더링에 최적화되어 있으며, 이러한 형식에 대한 광범위한 지원과 하드웨어 가속이 가능하다. 포인트 클라우드 렌더링으로의 전환은 새로운 렌더링 알고리즘, 데이터 구조 및 하드웨어 지원의 개발을 포함하여 기존 인프라에 상당한 변화를 요구한다. 이는 관련 산업계가 포인트 클라우드 기술을 채택하는 데 있어 더디게 만드는 주요 요인 중 하나다.

셋째, 포인트 클라우드 데이터의 특성상 데이터의 양이 방대하다는 점도 큰 도전 과제 중 하나다. 포인트 클라우드 데이터는 고해상도일수록 더욱 많은 포인트를 포함하게 되며, 이는 저장 및 전송에 있어 큰 부담이 된다. 이에 MPEG (Moving Picture Experts Group)은 V-PCC(Video-based Point Cloud Compression)[4]와 G-PCC(Geometry-based Point Cloud Compression)[5]를 통해 압축 표준을 제정하였다. V-PCC는 기존의 비디오 압축 기술을 활용하여 포인트 클라우드를 압축하는 방식이며, G-PCC는 기하학적 정보를 기반으로 포인트 클라우드를 압축하는 방식으로, 이 두 가지 표준은 포인트 클라우드 데이터의 효율적인 저장 및 전송을 가능하게 하였다.

결론적으로, MPEG은 V-PCC 및 G-PCC와 같은 압축 표준을 통해 포인트 클라우드 데이터를 효율적으로 압축하고 저장 및 전송을 가능하게 하였지만[6], 증강 현실 및 모바일 환경에서 포인트 클라우드의 광범위한 사용은 렌더링에서의 높은 계산 복잡도 및 최적화된 포인트 클라우드 렌더링 엔진의 부족 등의 문제로 인해 방해받고 있다. 따라서 본 논문에서는 Unity를 활용한 안드로이드 기반 증강 현실 환경에서 포인트 클라우드 렌더러를 구현한다. 이를 통해 안드로이드 기반의 증강 현실 환경에서 포인트 클라우드를 렌더링할 수 있도록 하고, 포인트 클라우드를 렌더링 할 때 발생하는 문제를 확인한다. 이후 문제점 분석 과정을 거쳐 본 논문의 렌더러의 렌더링 성능 개선과 향후 연구 방향을 모색하고자 한다.

Ⅱ. 관련 연구 분석

포인트 클라우드 렌더링(Point Cloud Rendering)은 3D 스캔 데이터를 시각화하는 기술로 복잡하고 큰 데이터를 포함하는 경우가 많아 이를 효율적으로 렌더링하는 방법이 중요한 과제로 대두되고 있다. 이로 인해 다양한 렌더링 기술이 개발되었고, 각 기술은 장점과 단점을 가지고 있다. 이러한 다양한 렌더링 기술의 개발은 포인트 클라우드의 시각적 품질과 효율성을 향상시켜 다양한 장치에서 일관된 사용자 경험을 제공하는 데 기여하고 있다.

Schuetz(2015)[7]는 대규모 포인트 클라우드를 웹 브라우저에서 렌더링할 수 있는 Potree라는 오픈 소스 도구를 소

a) 연세대학교 소프트웨어학부(Division of Software, Yonsei University)
‡ Corresponding Author : 서광덕(Kwang-deok Seo)
E-mail: kdseo@yonsei.ac.kr
Tel: +82-33-760-2788
ORCID: https://orcid.org/0000-0001-5823-2857
※본 연구는 과학기술정보통신부 및 정보통신기획평가원의 SW중심대학 지원사업(2019-0-01219)과 교육부의 재원으로 한국연구재단(2021R1F1A1048404)의 지원을 받아 수행되었음.
· Manuscript August 6, 2024; Revised August 29, 2024; Accepted August 29, 2024.

개하였다. Potree는 LIDAR와 같은 소스에서 수집된 수십억 개의 포인트를 실시간으로 시각화할 수 있는 장점이 있다. 이러한 접근법은 웹 기반으로 데이터 공유와 분석이 용이하다는 장점이 있지만, 복잡한 데이터 세트에서는 성능 저하가 발생할 수 있으며 초기 설정과 최적화 과정이 복잡할 수 있다. 이로 인해 Potree는 실시간 렌더링에 장점을 보이지만, 데이터 처리에 있어 한계가 있다.

이와 같이 포인트 클라우드 데이터를 직접적으로 렌더링하는 방법의 한계를 극복하기 위해, 신경망 기반 렌더링 기법이 등장하였다. 이러한 방법들은 포인트와 해당 Feature Descriptor를 2D 평면에 투영하여 희소 이미지를 얻고[8], U-Net[9]과 유사한 네트워크를 사용하여 이미지를 복원한다. 그러나 이 접근법은 2D 이미지 복원에 의존하므로 3D 일관성을 보장하기 어렵기 때문에 인접한 시점에서 동일 객체의 모습이 달라질 수 있다.

이러한 문제를 해결하기 위해, Neural Radiance Field (NeRF)[10] 기술이 포인트 클라우드 렌더러에 통합되었다. NeRF는 인접한 시점에서 동일 객체의 일관성을 유지하며 사실적인 이미지를 생성할 수 있지만, 고차원 특징 벡터를 저장하는 데 대규모의 메모리 소비가 필요하며, 많은 샘플링 포인트와 신경망 접근이 요구되어 렌더링 시간이 증가한다. NeRF의 이러한 한계는 실시간 렌더링이 필요한 응용에서 문제를 일으킬 수 있다.

이로 인해, 최근에는 3D Gaussian Splatting(3DGS)[11]기술 사용을 통해 새로운 표현 방식을 도입하고 있다. 3DGS는 포인트를 3D 가우시안으로 공식화하고, 3D 위치, 색상, 불투명도 및 비등방성 공분산을 포함한 학습 가능한 매개변수를 사용하여 렌더링을 수행한다. 3DGS는 가우시안 래스터화와 α-블렌딩을 적용하여 NeRF보다 더 사실적이고 빠른 렌더링을 달성한다. 다만 3DGS는 다중 시점 이미지와 역전파가 필요하므로 포인트 클라우드만 사용하여 렌더링하는 상황에서는 적용이 어려울 수 있다.

위에서 논의한 포인트 클라우드 렌더링 방법들은 모두 각각의 장점과 단점을 가지고 있으며, 효율적인 포인트 클라우드 렌더링을 목적으로 하고 있다. 그러나 이 논문들은 여전히 포인트 클라우드의 직접적인 처리와 렌더링을 위한 추가적인 연구가 필요함을 시사하며, 모두 모바일 AR 환경에서의 포인트 클라우드 렌더링을 지원하지 않는다.

이에 반해, 본 논문에서 제안하는 포인트 클라우드 비디오 렌더러는 AR 환경에서 포인트 클라우드를 렌더링할 수 있으며 문제점 분석 및 향후 연구 방향에 대한 모색을 통해 렌더러의 성능 개선을 목표로 한다. 따라서 추후 본 논문에서 제안하는 렌더러에 대한 지속적인 연구 및 개발이 이루어진다면 사용자들은 모바일 기기에서도 고품질의 포인트 클라우드 데이터를 시각화하고 상호작용하여 높은 사용자 경험을 기대할 수 있다.

Ⅲ. 제안하는 포인트 클라우드 비디오 렌더러 설계 및 구현

본 논문에서는 안드로이드 기반 AR 환경에서 포인트 클라우드 비디오를 렌더링하기 위한 애플리케이션을 설계하고 구현하였다. 이를 위해 Unity[12] 환경을 기반으로 Google AR Core[13]를 이식하였으며, 구현된 포인트 클라우드 비디오 렌더러의 구조도는 그림 1에 제시되어 있다. 해당 구조도에서 입력으로 사용되는 데이터는 PLY(Polygon File Format)[14] 파일 형식으로 구성된다. PLY 파일 형식은 포인트 클라우드 데이터를 효율적으로 저장하고 표현할 수 있는 유연한 구조를 가지며 점, 색상 등 다양한 속성을 저장할 수 있다. 또한, MPEG 3D Graphics Coding and Haptics Coding 그룹(ISO/IEC JTC1/SC29 WG07)은 포인트 클라우드 데이터를 다루기 위해 PLY 형식을 채택하였다. 따라서 PLY 형식의 포괄적인 데이터 표현 능력 및 국제 표준과의 호환성 유지를 위해 본 논문에서도 PLY 파일을 입력

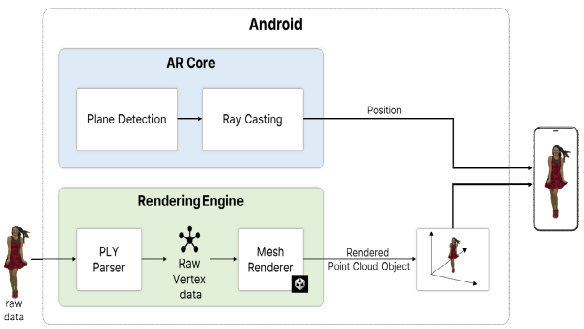

그림 1. 포인트 클라우드 비디오 렌더러 구조도
Fig. 1. Point Cloud Video Renderer Architecture

데이터로 사용한다. Unity는 현재 PLY 파일 형식을 직접 렌더링할 수 있는 솔루션을 제공하지 않기 때문에, PLY 파일을 Unity에서 사용할 수 있도록 하려면 먼저 파일의 구조를 분석하고 파싱해야 한다. 이 과정에는 PLY 파일의 헤더와 데이터 섹션을 해석하고, 포인트 클라우드 데이터를 효율적으로 처리할 수 있는 코드 작성이 포함된다.

1. PLY 파일형식의 구조 분석

포인트 클라우드 데이터를 저장하는 파일 형식중 하나인 PLY 파일 형식은 3D 객체를 polygonal model로 나타내는 형식으로 그렉 터크(Greg Turk)가 1994년 스탠포드 그래픽스 연구소(Stanford Computer Graphics Laboratory)에서 개발하였으며, 구현이 간단하고 쉽지만 다양한 모델에 유용할 만큼 일반적인 형식을 제공하는 것을 목표로 한다. 일반적인 PLY 파일의 구조는 헤더(Header), Vertex List, Face List 및 lists of other elements의 구성요소로 이루어지며 PLY 파일의 형식의 헤더 구조는 표 1과 같다.

헤더의 첫 번째 줄은 ASCII 문자 “ply”와 carriage return(CR)으로 이루어지며 magic number로 사용된다. 헤더의 두 번째 줄은 형식을 선언하는 부분이다. 헤더 이후의 Vertex List, Face List와 같은 실질적인 3D 객체 데이터를 표현하는데 사용된 형식인 format ascii 1.0, format binary_little_endian 1.0 또는 format binary_big_endian 1.0 중 하나의 값으로 선언한다. 헤더의 세 번째 줄에서는 3D 객체 데이터의 구조를 선언하며, 각 요소와 관련된 속성을 식별한다. PLY 파일 형식의 핵심 요소는 Vertex와 Face로 구성되며 헤더에서는 Vertex의 속성으로 Vertex의 수 및 위치 정보인 x, y, z 좌표 및 색상 정보인 r, g, b가 property로 선언되고, Face의 속성으로 Face가 포함하는 Vertex의 수와 해당 Vertex의 index를 지정한다.

2. PLY 파일 파싱을 통한 포인트 클라우드 렌더링

Unity는 PLY 파일형식을 지원하지 않기 때문에 PLY 파일 렌더링을 위해서는 직접 PLY 파일의 헤더를 파싱하고 Unity의 Mesh Filter와 Mesh Renderer를 사용하여 렌더링한다. Unity에서 Mesh Filter와 Mesh Renderer는 3D 모델을 화면에 표현하기 위해 사용되는 컴포넌트다. 두 컴포넌트는 서로 상호작용하여 게임 오브젝트에 객체를 적용하고, 이를 화면에 렌더링하는 역할을 한다. Mesh Filter는 게임 오브젝트에 3D 객체 데이터를 할당하는 역할을 하며 Mesh Renderer는 Mesh Filter에 의해 지정된 3D 객체 데이터를 실제로 화면에 렌더링하는 역할을 한다. 따라서 Unity에서 3D 객체를 렌더링하기 위해서는 PLY 파일의 Vertex List에서 각각의 Vertex에 대한 속성 정보를 Mesh Filter에 전달하고 Mesh Renderer를 통해 최종적으로 렌더링해야 한다. 이를 위해, PLY 파일의 헤더 부분에서 element vertex로 정의되는 총 정점 개수와 각 정점의 위치 정보인 x, y, z 좌표 및 색상 정보인 r, g, b에 해당하는 property 정보를 확인한다. 그리고 해당 정보들을 기반으로 PLY 파일의 Vertex List에서 Vertex의 개수와 실질적으로 사용할 속성

표 1. PLY 파일의 형식의 Header 구조
Table 1. Structure of a PLY file Header

Structure of a PLY file Header			Description
ply			Beginning of the Header. “ply” followed by carriage return and serves as a magic number
format			ascii/binary, Format version number
Comment			Comments keyword specified
element	<element-name>	<number-in-file>	Identifying elements and properties associated with each element
property	<data-type>	<property-name-1>	
property	<data-type>	<property-name-2>	
...			
element	<element-name>	<number-in-file>	
property	<data-type>	<property-name-1>	
...			
end_header			End of the Header

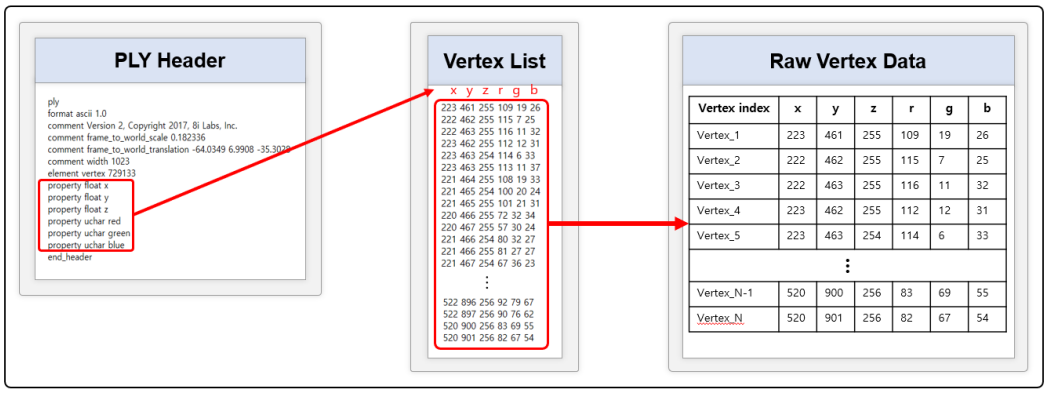

그림 2. Vertex 속성정보 추출 과정
Fig. 2. Vertex Property Information Extraction Process

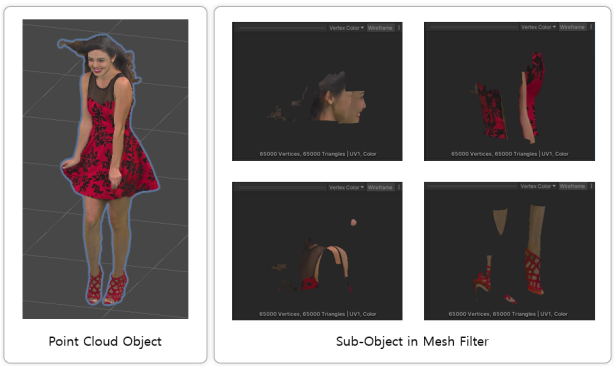

그림 3. 포인트 클라우드 객체 렌더링 결과
Fig. 3. Rendering Results of Point Cloud Objects

정보들을 추출하여 raw vertex 데이터로 구성한다. 그림 2는 MPEG 3D Graphics Coding and Haptics Coding에서 Common Test Condition dataset[15]으로 사용하는 8i Voxelized Full Bodies, version 2 중 redandblack_vox10 시퀀스의 PLY 파일 구조와 Vertex의 속성 정보 추출과정을 나타낸다.

이렇게 구성된 raw vertex 데이터는 Mesh Filter에 전달되어 3차원 형상정보로 정의되게 되며 총 42개의 객체로 분할되어 각각 Mesh Renderer를 통해 포인트 클라우드 객체로 렌더링 된다.

이후 최종적으로 Unity Engine에서 하나의 Game Object로 생성된다. 그림 3은 Mesh Filter와 Mesh Renderer에 포인트 클라우드 데이터가 렌더링 된 결과다.

3. Android 기반 증강 현실 환경에서 포인트 클라우드 비디오 재생 실험

실험은 삼성 갤럭시 노트 23 울트라를 사용한 Android 14(API Level 34), RAM 4GB 환경에서 진행하였다. 재생한 테스트 시퀀스는 2.2 절에서 설명한 8i Voxelized Full Bodies, version 2 중 redandblack_vox10 시퀀스를 사용하였다. 표 2는 테스트 시퀀스에 대한 자세한 정보를 나타낸다.

재생 실험의 과정은 다음과 같은 3가지의 단계를 거쳐 최종적으로 포인트 클라우드 객체로 렌더링 된다. 첫 번째 단계는 Google AR Core의 AR Plane Manager를 통해 모바일 증강 현실 환경에서 포인트 클라우드 객체가 렌더링 될 바닥을 현실 세계로 부터 검출하고, Polygon Plane으로 표현한다. 두 번째 단계에서는 모바일 기기 화면의 중점으로부터 Ray Casting을 통해 Polygon Plane과 가상의 광선이

표 2. 8i Voxelized Full Bodies, version 2 - redandblack_vox10
Table 2. 8i Voxelized Full Bodies, version 2 - redandblack_vox10

Content Category	Test material dataset filename	Frames	fps	# Pts	Geometry Precision	Peak Value (p)	Attributes
Dynamic Objects	8i VFB - Red_and_Blackb	300	30	~700,000	10 bit	1023	R,G,B

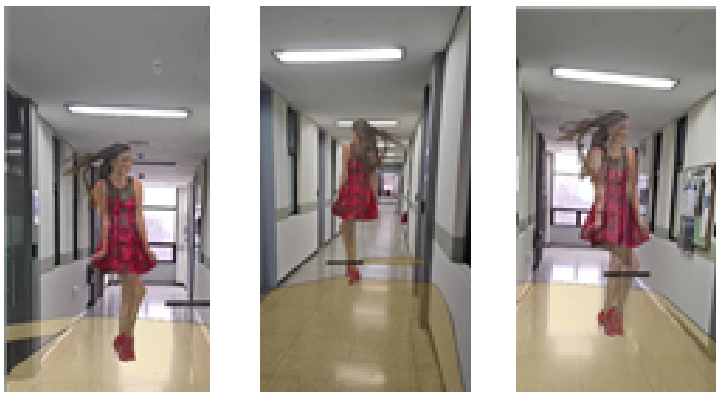

그림 4. AR 환경에서의 포인트 클라우드 객체 렌더링 결과
Fig. 4. Rendering Results of Point Cloud Objects in an AR environment

교차하는 지점의 좌표를 Position으로 추출한다. 이때, Position은 Vector3 형식으로 제공된다. 마지막 단계로 제공된 Position의 위치에 테스트 시퀀스가 렌더링 되어 포인트 클라우드 비디오를 재생할 수 있다. 재생 결과는 그림 4와 같다.

안드로이드 기반의 증강 현실 환경에서 포인트 클라우드 비디오를 재생하기 위한 렌더러 애플리케이션을 테스트한 결과, 프레임을 파싱하는 과정에서 한 프레임 당 약 5초의 지연이 발생함을 확인하였다. 이러한 지연을 방지하고 원활한 재생을 위해서 모든 프레임을 미리 로드한 후 재생 실험을 진행하였으며, 위와 같은 경우에 RAM의 여유 공간 부족으로 인한 애플리케이션이 강제 종료 되는 문제를 방지하기 위해 포인트 클라우드 비디오의 전체 용량에 상응하는 메모리 공간이 충분히 확보되어야 한다. 또한, PC와 모바일 장치 간의 하드웨어 성능 차이로 인해 모바일 기기에서 렌더링된 포인트 클라우드 객체의 시각적 품질이 상대적으로 낮은 것을 확인할 수 있었다.

Ⅳ. 결 론

본 논문에서 제안한 렌더러 애플리케이션을 평가한 결과를 통해 크게 2가지 문제점을 확인할 수 있었다. 첫째, 프레임을 파싱하는 과정에서 약 5초의 지연이 발생한다는 점을 확인하였다. 이러한 지연은 프레임 데이터를 실시간으로 처리하는 데 있어 큰 제약 요소가 되며, 원활한 비디오 재생을 보장하기 위해서는 모든 프레임을 사전에 로드하는 방식을 채택해야 한다는 결론에 이르게 되었다. 이와 같은 접근은 포인트 클라우드 비디오의 전체 용량에 상응하는 상당량의 메모리 공간을 필요로 하며, 메모리 용량 부족으로 인해 모바일 장치에서는 애플리케이션이 강제 종료되는 등의 문제가 발생할 수 있다. 이러한 문제는 특히 모바일 장치의 RAM 용량이 제한적일 때 더욱 두드러지며, 이로 인해 사용자 경험에 심각한 영향을 미칠 수 있다. 둘째, PC와 모바일 장치 간의 하드웨어 성능 차이로 인해, 모바일 기기에서 렌더링 된 포인트 클라우드 객체의 시각적 품질이 상대적으로 낮다는 점도 확인되었다. 이는 고해상도 포인트 클라우드 데이터의 렌더링과 관련된 성능 차이에서 기인하며, 모바일 장치의 제한된 그래픽 처리 능력으로 인해 시각적 품질이 저하되는 현상이 나타난다. 이러한 품질 저하는 모바일 장치에서의 사용자 경험 수준을 떨어뜨릴 수 있으며, 이는 향후 연구에서 개선이 필요한 중요한 요소로 고려될 수 있다.

이러한 연구 결과를 바탕으로, 포인트 클라우드 비디오 재생의 성능을 보완하기 위해 다음과 같은 연구 방향이 필요하다. 첫째, 원활한 비디오 재생을 보장하기 위해 프레임을 사전에 로드하는 방식 외에도 메모리 요구 사항을 줄이는 기술적 접근이 필요하다. 예를 들어, 포인트 클라우드 데이터의 다운스케일링 또는 메모리 효율성을 높이는 기법들이 검토되어야 한다. 이러한 방법은 특히 모바일 장치에

서의 성능 향상에 기여할 수 있으며, 데이터 처리와 렌더링을 보다 원활하게 만들어줄 수 있다. 둘째, PLY 파일 파싱 과정에서의 지연을 줄이기 위해 파싱 코드를 최적화하는 것이 중요하다. 파싱 코드의 최적화와 알고리즘 개선을 통해 데이터 처리 속도를 높이면, 모바일 환경에서도 보다 매끄러운 비디오 재생이 실현될 수 있다. 코드의 성능을 개선하는 작업은 전체 애플리케이션의 효율성을 높이는 데 중요한 역할을 하며, 최적화된 파싱 코드는 지연 시간을 최소화하고 사용자 경험을 향상시킬 수 있다. 이와 같은 연구 방향은 안드로이드 기반 증강 현실 환경에서 포인트 클라우드 비디오의 재생 효율성을 높이는 데 기여할 수 있으며, 다양한 장치에서 일관된 사용자 경험을 제공할 것이라고 기대할 수 있다.

참 고 문 헌 (References)

[1] P. A. Chou, M. Koroteev and M. Krivokuća, “A Volumetric Approach to Point Cloud Compression-Part I: Attribute Compression,” IEEE Transactions on Image Processing, Vol.29, pp. 2203-2216, 2019. doi: https://doi.org/10.1109/TIP.2019.2908095.

[2] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu and M. Bennamoun, “Deep Learning for 3D Point Clouds: A Survey,” in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 43, no. 12, pp. 4338-4364, 1 Dec. 2021. doi: https://doi.org/10.1109/TPAMI.2020.3005434.

[3] W. Pasman and F. W. Jansen, “Distributed low-latency rendering for mobile AR,” Proceedings IEEE and ACM International Symposium on Augmented Reality, New York, NY, USA, 2001, pp. 107-113. doi: https://doi.org/10.1109/ISAR.2001.970520.

[4] ISO/IEC JTC 1/SC 29, ISO/IEC 23090-5:2023, Information technology − Coded representation of immersive media − part 5: visual volumetric video-based coding (V3C) and video-based point cloud compression (V-PCC), ISO/IEC, 2023.

[5] ISO/IEC JTC 1/SC 29, ISO/IEC 23090-9:2023, Information technology − Coded representation of immersive media − Part 9: Geometry-based point cloud compression, ISO/IEC, 2023.

[6] S. Schwarz et al., “Emerging MPEG Standards for Point Cloud Compression,” in IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 9, no. 1, pp. 133-148, March 2019. doi: https://doi.org/10.1109/JETCAS.2018.2885981.

[7] Schütz, Markus. Potree: Rendering large point clouds in web browsers. Diss. Technische Universität Wien, 2015. doi: https://doi.org/10.34726/hss.2015.27710.

[8] T. Hu et al., “Trivol: Point cloud rendering via triple volumes,” Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 20732-20741, 2023. doi: https://doi.org/10.48550/arXiv.2303.16485.

[9] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” Medical image computing and computer-assisted intervention – MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18. doi: https://doi.org/10.1007/978-3-319-24574-4_28.

[10] B. Mildenhall et al., “Nerf: Representing scenes as neural radiance fields for view synthesis,” Communications of the ACM, Vol. 65, Issue 1, pp. 99-106, 2021. doi: https://doi.org/10.1145/3503250.

[11] B. Kerbl et al., “3D Gaussian Splatting for Real-Time Radiance Field Rendering,” ACM Trans. Graph., Vol. 42, Issue 4, pp 139:1-139:14, 2023. doi: https://doi.org/10.48550/arXiv.2308.04079.

[12] Google AR Core, https://developers.google.com/ar/develop (accessed July. 20, 2024).

[13] Unity, https://docs.unity3d.com/Manual/index.html (accessed July. 20, 2024).

[14] G. Turk, “The ply polygon file format,” Recuperado de, 1994.

[15] ISO/IEC JTC1/SC29 WG7, Common Test Conditions for V-PCC, ISO/IEC JTC1/SC29 WG7 Doc. N19613, Online, October 2020.

저 자 소 개

김 민 석

- 2024년 2월 : 연세대학교 컴퓨터정보통신공학부 학사
- 2024년 3월 ~ 현재 : 연세대학교 전산학과 석사과정
- ORCID : https://orcid.org/0009-0002-5665-3740
- 주관심분야 : 기계 학습을 위한 비디오 코딩, 몰입형 미디어

저 자 소 개

이 수 연

- 2022년 2월 : 연세대학교 컴퓨터정보통신공학부 학사
- 2022년 8월 ~ 현재 : 연세대학교 전산학과 석사과정
- 주관심분야 : 몰입형 미디어, 멀티미디어 통신 시스템

김 아 영

- 2016년 2월 : 연세대학교 컴퓨터정보통신공학부 학사
- 2016년 3월 ~ 2024년 2월 : 연세대학교 전산학과 박사
- ORCID : https://orcid.org/0000-0002-3793-1365
- 주관심분야 : 기계 학습을 위한 비디오 코딩, 몰입형 미디어, 멀티미디어 통신 시스템

안 은 빈

- 2016년 8월 : 연세대학교 컴퓨터정보통신공학부 학사
- 2017년 3월 ~ 현재 : 연세대학교 전산학과 석박사 통합과정
- ORCID : https://orcid.org/0000-0001-7681-4682
- 주관심분야 : 기계 학습을 위한 비디오 코딩, 몰입형 미디어, 멀티미디어 통신 시스템

서 광 덕

- 1996년 2월 : 한국과학기술원(KAIST) 전기공학과 학사
- 1998년 2월 : 한국과학기술원(KAIST) 전기공학과 석사
- 2002년 8월 : 한국과학기술원(KAIST) 전기공학과 박사
- 2002년 8월 ~ 2005년 2월 : LG전자 선임연구원
- 2005년 3월 ~ 현재 : 연세대학교 소프트웨어학부 교수
- 2012년 9월 ~ 2013년 8월 : 미국 플로리다 대학교 초빙교수
- ORCID : http://orcid.org/0000-0001-5823-2857
- 주관심분야 : 비디오 코딩, 시각적 통신, 디지털 방송, 멀티미디어 통신 시스템