JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 29 No. 12, pp. 99-108, December 2024
https://doi.org/10.9708/jksci.2024.29.12.099

3D Visualization Technology for Mass-data using Instance Mesh

Chul-Hee Lee*, Sang-Young Choi*

*Professor, Dept. of Cloud Security, Korea Polytechnics, Daejeon, Korea

[Abstract]

Large-scale data visualization is playing a key role in various fields such as system monitoring, cybersecurity threat detection, and data center resource management due to technological advancements such as cloud computing and the increase of data centers. However, there is a performance limitation in real-time 3D visualization of a large amount of information. In this paper, we propose a 3D visualization approach based on InstanceMesh technology to solve this problem. The proposed technology efficiently renders multiple data with a single draw call, effectively utilizing CPU and GPU resources. Experimental results show that InstanceMesh can effectively process large-scale data compared to existing technologies, achieving a frame rate of 110 fps and a low memory usage of 1 MB. Through this, it is expected that future studies will strengthen the data interaction function and utilize it in various fields such as real-time analysis and predictive simulation.

▸ **Key words**: 3D Visualization, Instance Mesh, WebGL, Data visualization, Monitoring, Performance Optimization

[요 약]

대규모 데이터 시각화는 클라우드컴퓨팅, 데이터센터의 증가 등 기술의 발전에 따라 시스템 모니터링, 사이버 보안 위협 탐지, 데이터센터 자원 관리 등 다양한 분야에서 핵심적인 역할을 하고 있다. 하지만 방대한 양의 정보를 3D로 실시간 시각화 하는 데 성능 한계가 있다. 본 논문에서는 이러한 문제를 해결하기 위해 인스턴스메시(InstanceMesh) 기술 기반의 3D 시각화 접근 방식을 제안한다. 제안된 기술은 단일 드로우 콜로 여러 데이터를 효율적으로 렌더링하여 CPU와 GPU 리소스를 효과적으로 활용한다. 실험 결과, InstanceMesh 활용 시 프레임 수 110fps와 1MB의 낮은 메모리 사용량을 당성하여 기존 기술 대비 대규모 데이터를 효과적으로 처리할 수 있음을 확인하였다. 이를 통해 향후 연구에서는 데이터 상호작용 기능을 강화해 실시간 분석과 예측 시뮬레이션 등 다양한 분야에서 활용될 것으로 기대된다.

▸ **주제어**: 3D시각화, 인스턴스메시, 웹GL, 데이터시각화, 모니터링, 성능 최적화

• First Author: Chul-Hee Lee, Corresponding Author: Sang-Young Choi
*Chul-Hee Lee (cjfclrchr@gmail.com), Dept. of Cloud Security, Korea Polytechnics
*Sang-Young Choi (csyong95@gmail.com), Dept. of Cloud Security, Korea Polytechnics
• Received: 2024. 09. 27, Revised: 2024. 12. 02, Accepted: 2024. 12. 17.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

대규모 데이터 시각화는 과학 연구, 엔지니어링, 데이터 분석, 보안 모니터링, 대량시스템 성능 모니터링 등 다양한 분야에서 중요한 역할을 한다. 데이터 과학자들은 데이터 시각화를 통해 데이터를 분석하며, 이를 통해서 유용한 정보를 얻는다[1]. 하지만 데이터 세트의 크기와 복잡성이 계속 증가함에 따라 의미 있는 통찰력을 추출하기 위해서는 보다 효율적인 처리 및 시각화 기술이 필요하다.

데이터 시각화를 위해 전통적으로 Matplotlib, VisFlowConnect, FloVis와 같은 2D기반의 시각화 기술이 사용되었으나 데이터 양의 증가와 표현해야 하는 객체의 증가 등에 따라 처리성능 뿐만 아니라 시각화의 효과 또한 보증하기 힘든 한계가 있다. 2D시각화 기술은 x,y두 개의 축으로 데이터를 표현해야 하기 때문에 데이터간의 공간 관계를 정확하게 표현하는데 한계가 있다. 이로 인해 사용자에게 효과적인 가시성을 제공하지 못할 수 있다. 이와 같은 2D기반의 시각화 기술의 한계 개선을 위해 최근 D3.js, Three.js와 같은 3D시각화 기술을 사용하여 데이터 세트의 크기와 복잡성 증가에 대처하기 위한 노력을 하고 있으나, 이 방법 또한 한계점이 존재한다. 특히 고차원 데이터 세트 또는 실시간 스트리밍 데이터를 처리할 때 성능 병목현상이 발생되어 렌더링 속도가 느려지고 상호 작용이 제한되어 유의미한 통찰을 제공하는데 한계가 있다.

또한, 3D 시각화 기술에서는 각 데이터를 오브젝트로 처리하는 과정에서 3D 장면을 렌더링 하는 부분에서 CPU, GPU 성능과 메모리 자원이 필요하며[2], 특히 실시간 시각화 애플리케이션인 경우 렌더링 속도 저하, 상호 지연 등의 문제가 발생한다[3]. 조밀한 데이터인 경우 객체가 서로 겹치거나 가려지는 시각적으로 식별하기 어려움을 동반한다.

렌더링시 자원 사용을 줄이기 위한 방법으로 여러 방법이 있다. 파이프라인 기법[3], 단순화 기법(LOD) 기법을 사용하기도 한다[4]. 하지만 단순화 기법은 결국 가까운 곳은 선명하게 보이고 먼 곳은 희미하게 보이는 알고리즘으로 데이터의 중요도 또는 공간상의 위치를 보다 명확하게 표현하지만 정확한 데이터 표현에는 한계가 있다.

웹 브라우저에서는 3D 표현을 하기 위해서는 WebGL 기술을 사용한다. WebGL이란 에서 추가적인 플러그인을 사용하지 않고, OpenGL ES 2.0 기반 API를 이용하여 브라우저의 HTML canvas에 렌더링하여 3D 웹 콘텐츠 제작을 가능하게 하는 프로그램이다. WebGL 프로그램은 컴퓨터의 그래픽 처리 장치(GPU)에서 실행되는 JavaScript 나 특수 효과(셰이더 코드)코드로 구성되고. WebGL 요소들은 다른 HTML 요소들과 혼합될 수 있고 페이지나 페이지 배경의 다른 부분과 합성될 수 있다.[5]

Three.js는 JavaScript 3D 라이브러리로, WebGL의 복잡한 렌더링 과정을 추상화하여 카메라, 조명, 렌더링 등을 쉽게 구현할 수 있도록 돕는다[6]. InstanceMesh는 Three.js에서 제공하는 성능 최적화 기법으로, 동일한 지오메트리를 여러 번 복제하여 한 번의 드로우 콜로 여러 개의 인스턴스를 동시에 렌더링하는 기술이다. 이를 통해 대규모 데이터를 처리할 때 성능을 크게 향상시킬 수 있으며, 메모리 사용량을 절감할 수 있다.

본 논문에서는 webgl 기반 three.js 라이브러리의 InstanceMesh 기술을 이용하여 3차원 위에 대용량 데이터를 시각화하는데 성능 병목 현상을 줄여 사용자 경험에 효과적일 수 있는 시각화 처리 기술을 제안한다.

II. Preliminaries

1. Related works

1.1 Visualization Technology

1.1.1 javascript D3.js

Data-Drive-Documents의 약자로 자바스크립트 라이브러리인 D3.js는 웹브라우저상에서 동적이고 인터렉티브한 정보시각화를 구현하기 위한 도구이다.

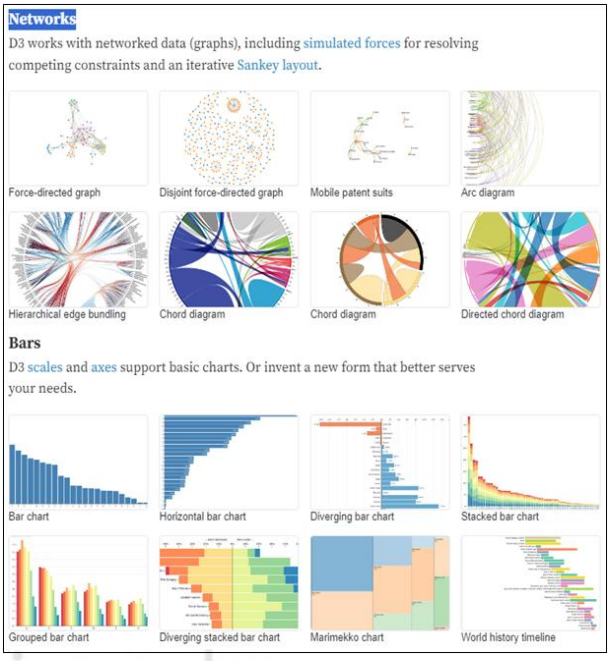

Fig. 1. D3.js web page Example

D3의 장점으로는 웹 표준인 HTML, SVG, CSS에서 파생되어 쉽게 접근이 가능하며, 각종 2D 기반의 차트(treemap, bar chart, line chart, pie chart, hitmap, hierarchies, networks 등)에 필요한 기능을 함수 단위로 제공해 준다. Fig. 1.은 D3에서 제공하고 있는 시각화 예제 갤러리 중 Network 탭을 보여 준다.

1.1.2 Python Matplotlib

Matplotlib[7]은 Python에서 정적, 애니메이션 및 대화형 시각화를 생성하기 위한 라이브러리이다. 3D 차트 관련 시각화 기능 또한 제공하지만, 2D 그래프를 그리는데 특화 되어 있으며, 선그래프, 산점도, 히스토그램, 파이차트, 히트맵 등의 다용한 종류의 그래프를 python 언어를 사용하여 그릴 수 있다. Fig. 2.는 Matplotlib 공식 홈페이지에서 제공하고 있는 예제 문서이다.

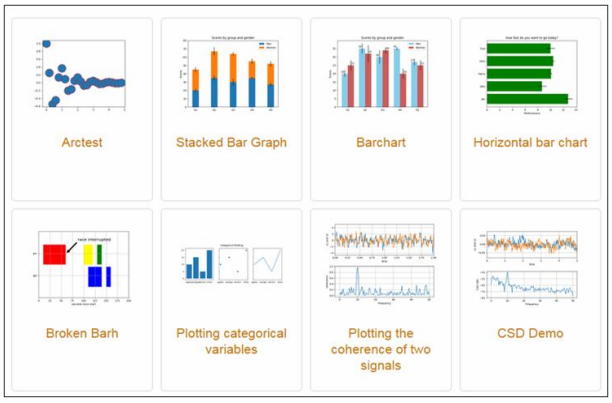

Fig. 2. Examples provided by Python Matplotlib

1.1.3 VisFlowConnect

Netflow 데이터를 사용하여 보안 상황인지를 목적으로 개발된 VisFlowConnect는 외부 domain과 내부 시스템의 네트워크 트래픽 흐름을 나타내는 시각화 도구이다. VisFlowConnect를 통한 대규모 네트워크상의 트래픽의 시각화를 하여 데이터를 확인하는 경우에는 너무나 많은 선들이 화면을 채우게 되어 이슈 식별이 어려워지는 문제가 있다.[8] Fig. 3.은 내부 호스트들과 외부 머신들 간의 관계를 방향과 트래픽 양을 포함하여 시각화해 보여주는 것이다. 2차원의 평면상에 표시된 평행한 축들은 각각 내부 또는 외부의 시스템들의 그룹을 나타내며, 축 상의 한 개의 점은 하나의 IP를 할당받은 시스템을 나타낸다. 그리고 각각의 축 위에 점으로 나타난 시스템들을 잇는 선은 해당 시스템들 간의 데이터 흐름을 표시한다.

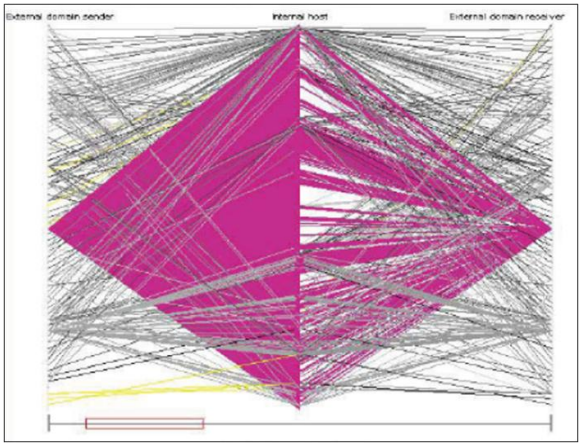

Fig. 3. VisFlowConnect

1.1.4 FloVis

FloVis[9]는 네트워크 트래픽 흐름을 다양한 방법으로 보여주기 위해 디자인된 시각화 도구 모음이다. Netflow 연결을 보여주는데 초점을 두며 시스템의 포트별 시간의 변화와 데이터 흐름의 양을 보여주는 NetBytes Viewer를 제공하며, 관찰할 대상 시스템들 간의 흐름을 표현하여 사용자가 네트워크 상황을 한눈에 볼 수 있도록 하는 장점이 있다. 하지만 이 또한 VisFlowConnect와 마찬가지로 다수의 선으로 연결이 표시됨에 따라 이슈정보를 식별하기 어려움이 있다.

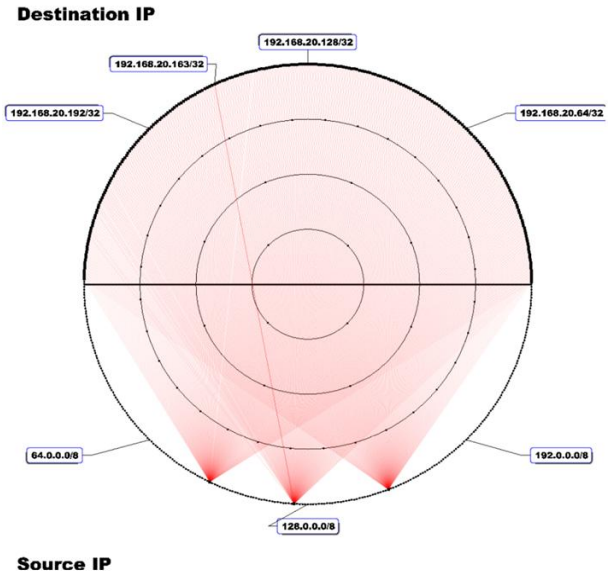

Fig. 4. FloVis

1.1.5 Visualization with sampling and simplification

기존의 대규모 데이터 시각화 연구들은 주로 그래프 데이터 분석과 샘플링 기반 접근법에 초점을 맞춰 진행되었다.

VAS (Visualization Aware Sampling)[10] 와 같은 기법은 데이터를 랜덤 샘플링하거나 층화 샘플링하여 데이터 양을 줄이고 시각적 복잡성을 완화하려는 접근을 사용한다.

또한 모티브 간소화[11]를 통해 특정 조건을 만족 하는 서브 그래프를 그림 Fig. 5과 같이 특정 모양의 도형으로 대체하여 시각화 하는 방식을 적용하는 시도가 있다.

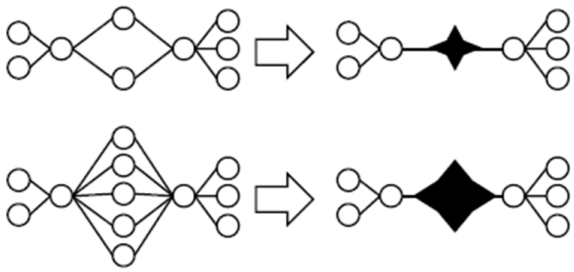

Fig. 5. Visualization example through motif simplification

이와 같이 기존 연구는 데이터를 간소화하거나 선택적으로 시각화하는 데 초점을 맞췄으며, 시각적 복잡성을 낮추는 데 성공했지만, 이는 데이터의 완전성과 세부 정보를 희생하는 결과를 초래할 수 있다. 또한, 대부분의 기존 연구는 고성능 장비나 서버 기반 환경을 전제로 한 접근 방식을 채택하여 범용적인 소비자용 환경에서의 실시간 시각화 구현이 미흡하다.

2. Limitation and Improvement

2.1 Limitation

전통적인 시각화 기술을 분석한 결과 Table.1과 같이 기존 기술의 가장 큰 한계는 데이터 처리에 중심을 두게 되어 가독성 저하 및 이로 인한 이슈 식별의 한계이며, 두 번째 한계점은 효과적인 3D표현의 한계가 있다는 것이 확인되었다.

Table 1. Limitation on Visualization Technique

Technique	Dimension	readability
D3.js	2D	High
Matplotlib	2D, 3D	Medium
VisFlowConnect	2D	Low
FloVis	2D	Low

2.2 Direction for improvement

전통적인 시각화 기술의 한계점인 가시성을 극복하기 위해 우선적으로 해결해야 하는 문제점은 대량 데이터를 효과적으로 처리하는 기술이다. 대량 데이터를 빠르고 효율적으로 병목없이 처리할 수 있는 기술이 기반이 되어야 3D를 포함한 가독성을 높일 수 있는 여러 가지 방법에 대한 적용이 가능하기 때문이다. 본 논문에서는 먼저 대량 데이터를 빠르고 효율적으로 처리할 수 있는 기술 중 하나인 InstanceMesh 기술을 활용하고자 한다. 이 기술을 활용하여 샘플링 및 간소화 접근법의 한계를 극복하고, 대규모 데이터의 세부 정보를 유지하면서 데이터의 가시성과 가독성을 높일 수 있는 시각화 구현을 통해 기존 방법과 제안하는 방법이 성능우위에 있는 것을 검증한다.

III. The Proposed Scheme

1. InstanceMesh

InstanceMesh란 인스턴스 렌더링을 지원하는 특수 버전의 메시이다. 기하학(Geometry)과 재질(Material)은 동일하지만 월드 변형이 다른 다수의 객체를 렌더링해야 하는 경우 InstanceMesh를 사용하면 그리기 호출 수를 줄여 애플리케이션의 전반적인 렌더링 성능을 향상시킬 수 있다. Table 2는 Three.js 공식 예제에서 사용된 조건하에 측정된 결과로, InstanceMesh 높은 fps와 낮은 메모리 사용량을 보이는 것을 확인할 수 있다. 이는 실험 이전에, 이론적으로 그리고 공식 예제에서 보여준 성능 수치를 기반으로 예상된 성능이다. 이러한 결과는 InstanceMesh의 장점을 확인하고, 이를 실제 대규모 데이터 시각화 실험에 적용할 가치가 있음을 시사한다.

Table 2. InstanceMesh Rendering Test

Method	Draw Call	GPU Memory
Naive	10,000	173.75KB 84fps(Fig. 6)
Merge	1	226.25MB 160fps(Fig. 7)
InstanceMesh	1	173.75KB 160fps(Fig. 8)

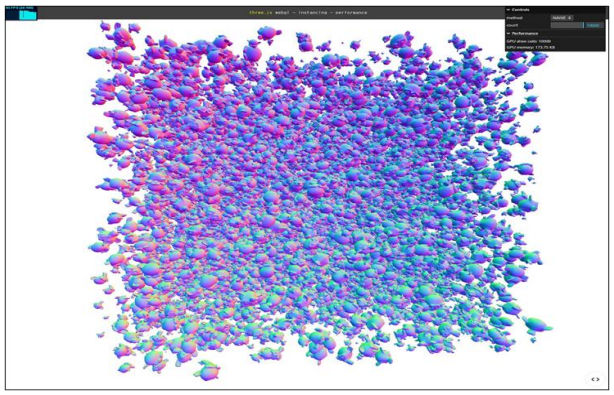

Fig. 6. NavieMesh rendering scene example provided on the threejs official website

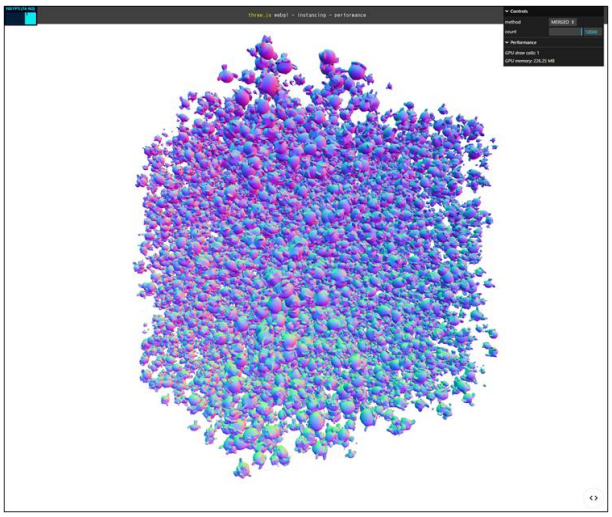

Fig. 7. MergeGeometries rendering scene example provided on the threejs official website

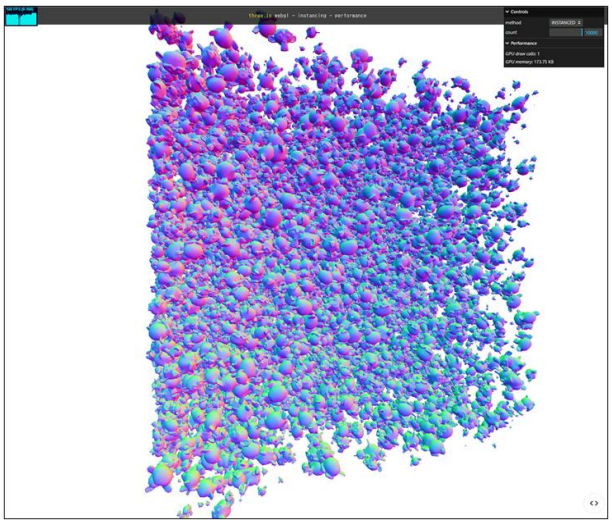

Fig. 8. InstacneMesh rendering scene example provided on the threejs official website

2. InstanceMesh based 3D visualization techniques

대규모 데이터 시각화를 InstanceMesh 기술의 이점을 활용하기 위해 본 연구에서는 시각화할 데이터 세트를 프로그램 내에서 무작위 1차원 값으로 정한 후, 특정 장비에서 발생한 이벤트 데이터를 24시간 기준 1분 데이터를 화면에 표시한다고 가정 했을 경우 1,440의 데이터를 하나의 데이터 세트로 놓고 Bar Chart로 시각화 하였다.

InstanceMesh 기술을 활용하는 과정에서 성능 개선을 위해 특정 파라미터를 선택하고 조정하였다. 대규모 데이터를 효율적으로 시각화하기 위해 new THREE.InstancedMesh() 함수를 중심으로 성능을 최적화하였다. InstanceMesh는 동일한 지오메트리를 여러 개의 인스턴스로 복제하여 렌더링할 때 사용되며, 인스턴스마다 위치나 크기를 개별적으로 설정할 필요 없이 하나의 드로우 콜로 여러 객체를 동시에 렌더링할 수 있다. 이를 통해 렌더링의 효율성을 극대화하였다.

실험은 new THREE.InstancedMesh() 함수를 사용하여 생성된 인스턴스를 시각화하는 방식으로 진행하였다. 여기서 함수의 파라메터 중 geometry는 기본적인 3D 객체의 모양을 정의하고, material은 각 객체의 색상과 투명도를 설정하며, count는 생성할 인스턴스의 개수를 의미한다. 실험에서 사용된 구체적인 값은 아래와 같다.

- geometry: new THREE.CylinderBufferGeometry (0.3, 0.3, 0.3, 3)를 사용하여 각 데이터 포인트를 시각화하였다. 파라메터 값은 순서대로 위 원통의 반지름, 아래 원통의 반지름, 원통의 높이를 나타내며, 마지막 파라메터는 기둥의 윗면과 아랫면 행의 개수를 나타낸다. 결과적으로 0.3의 반지름을 갖는 0.3의 높이의 삼각기둥을 표현한 값이다. 특히 마지막의 3 값은 세그먼트 수를 최소화하여 삼각형으로 데이터를 표현하며, 다음과 같은 이유로 이 값을 설정 하였다.
  1. 성능 최적화 : 세그먼트 수가 증가 할수록 복잡한 지오메트리를 렌더링하는 데 필요한 계산량이 급격히 증가한다. 3이라는 값은 삼각형 형태의 기둥을 만들어 지오메트리 복잡성을 최소화하면서도 렌더링 속도를 극대화할 수 있도록 돕는다. 대규모 데이터를 시각화할 때 이러한 최적화는 필수적이며, 더 복잡한 기하학적 형태를 사용하는 것보다 CPU와 GPU 리소스 사용량을 크게 줄일 수 있다.
  2. 데이터의 명확한 표현: 삼각기둥은 기둥을 형성하는 가장 작은 단위로, 단순하고 직관적인 형태로 데이터를 표현하기에 적합하다. 3개의 면으로 구성된 삼각기둥은 기하학적 복잡성을 최소화하면서도 시각적 혼란을 방지한고 3D 공간에서 입체적인 기둥을 사용하면 데이터의 크기 차이를 높이로 명확하게 나타낼 수 있기 때문에 2D 평면보다 더 많은 정보를 직관적으로 전달할 수 있다.
- material: new THREE.MeshBasicMaterial({ color: featureColor, transparent: true, opacity: 0.5 }) 을 사용하여 객체의 시각적 속성을 정의하였다. 여기서 featureColor는 데이터 특성에 따라 동적으로 변경되는 색상이며, opacity는 반투명 효과를 적용하여 시각적으로 겹침을 최소화하는 데 기여하였다.
- count: 총 737,280개의 인스턴스를 생성하여 성능을 테스트하였다. 이 숫자는 512개의 장비 세트를 가정하여 각 장비가 하루 동안 수집한 데이터를 3D 시각

화하였다. 따라서, 각 장비에서 수집된 데이터는 1,440개의 데이터 포인트로 구성되며, 이를 512개의 장비 세트에 걸쳐 확장한 결과, 총 737,280개의 인스턴스가 생성되었다. 이 값은 다음과 같이 계산된다:

$$1,440 \times 512 = 737,280$$

이러한 설정은 대규모 장비 네트워크에서 수집되는 데이터를 시각화할 때 실험의 현실성을 높이기 위한 가정이다. 예를 들어, 데이터 센터나 대규모 네트워크 인프라에서 다수의 장비가 동시다발적으로 데이터를 수집하는 경우를 상정한 것이다.

대규모 데이터를 처리하는 데 있어 성능 평가 지표로, 얼마나 많은 데이터를 효율적으로 렌더링할 수 있는지 확인하는 데 사용되었다.

이를 통해 불필요한 연산을 줄이고, 렌더링 성능을 크게 향상시켜 실시간 시각화 성능을 극대화하고, 데이터를 명확하고 효율적으로 시각화할 수 있다.

3. Test and Evaluation

3.1 Test method

앞서 Table 2 에서 사용된 데이터는 공식 예제를 바탕으로 예측된 성능을 나타낸 반면, 여기서는 대규모 데이터 세트로 실험을 진행한 결과를 기록하였다. 실험에는 전통적인 HTML 기반 접근 방식인 HTML 태그 기반 3D 그래프 CSS 기반 <img> 태그 렌더링과 제안 하는 기술인 InstanceMesh의 시각화 성능을 측정하였다. 3가지 기술을 비교하여 본 연구에서 제안한 InstanceMesh 기술의 성능 개선 효과를 검증하기 위해, 기존의 3D 시각화 방식과 비교하여 성능 차이를 평가하였다.

비교항목으로는 각 접근 방식에서의 프레임수(fps)와 메모리 사용량을 주요 성능 지표로 설정하였다. 성능 지표중 프레임수는 초당 렌더링되는 프레임의 수를 나타내며, 높은 프레임 수는 렌더링이 부드럽게 이루어지며 사용자가 원활하게 시각화를 탐색 할 수 있음을 의미한다[12]. 또한 메모리 사용량은 렌더링 과정에서 소모되는 메모리 용량을 측정하여, 시각화 기법이 얼마나 효율적으로 자원을 활용하는지 평가한다. 메모리 사용량이 낮을수록 시스템 부하가 줄어들어 더 많은 데이터를 효과적으로 처리할 수 있다.

3.2 Test Environment

실험 환경은 화면 렌더링은 클라이언트 PC사양에 따라 상이함으로 아래 Table 3. 에 제시된 동일한 환경에서 실험을 실시하였다.

실험에 사용된 장비는 Apple M1 CPU와 8GB 메모리를 탑재한 시스템이다. 이 장비를 선택한 이유는, 현대적인 소형화된 하드웨어에서도 데이터 시각화 기술을 적용할 수 있는 가능성을 탐구하는 것이었다. Apple M1칩은 8코어 CPU와 8코어 GPU를 통합한 ARM 기반 SoC(System on Chip)로, 통합 메모리 아키텍처를 사용하여 CPU와 GPU가 같은 메모리 풀을 공유한다. 8GB의 통합 메모리는 일상적인 작업에서 효율적인 성능을 발휘하지만, 대규모 데이터 시각화와 같은 메모리 및 그래픽 집약적인 작업에서는 제한적일 수 있다. 그럼에도 불구하고, 이 장비는 범용적인 환경에서의 실험을 통해 성능 제약을 평가하고, 실용적 시각화 기술을 탐구하는 데 목적이 있다.

실험 결과에서도 이러한 장비 성능의 한계가 드러났다. 512개의 이미지 박스를 배치한 상태에서 성능이 10.8fps로 떨어졌으며, 737,280개의 데이터를 처리하는 것은 메모리 제한과 GPU 성능 부족으로 인해 불가능했다. 8GB 메모리는 대규모 데이터를 처리하기에는 충분하지 않았으며, 통합 GPU의 한계로 인해 복잡한 데이터 시각화가 원활하게 이루어지지 않았다.

그러나 InstanceMesh 기술을 활용함으로써, 이러한 성능 제약을 어느 정도 극복할 수 있었다. InstanceMesh는 단일 드로우 콜을 통해 여러 인스턴스를 효율적으로 처리하므로, CPU와 GPU 리소스를 효과적으로 활용할 수 있다. 결과적으로, InstanceMesh를 사용하면 소비자용 하드웨어에서도 대규모 데이터 시각화가 가능한 수준으로 성능을 최적화할 수 있었다.

이는 소형화된 장비에서도 대규모 데이터를 시각화하는 새로운 가능성을 제시하며, 향후 고성능 하드웨어를 사용한 실험에서는 성능이 더욱 향상될 것으로 기대된다. InstanceMesh 기술의 사용 덕분에, 기존 접근 방식으로는 불가능했던 대규모 데이터의 시각화가 범용적인 소비자용 하드웨어에서도 가능함을 확인할 수 있었다.

본 연구에서는 Three.js 버전 109를 사용하여 실험을 진행하였다. 해당 버전은 실험 당시 InstanceMesh 기능을 안정적으로 지원하는 최신 버전 중 하나였으며, 실험 환경 내에서 안정성과 호환성을 유지하기 위해 사용되었다. 이후 Three.js의 최신 버전이 릴리스되었지만, 연구의 일관성과 성능 분석을 위해 버전 109로 실험을 완료하였다.

최신 버전은 성능 최적화 및 추가 기능을 제공할 가능성이 있으며, 이는 향후 연구에서 더욱 심층적으로 다룰 예정이다. 시간이 제한된 관계로 이번 실험에서는 최신 버전과의 비교 실험을 수행하지 못했으나, 차후 연구에서는 최신 버전의 업데이트된 기능 및 성능 개선 요소를 반영한 실험을 진행할 계획이다.

Table 3. Test equipment performance table

Chip	Spec
Cpu	Apple M1
Memory	8Gb
Browser	Chrome(ver.120)
threejs	ver.109

3.3 Test Procedure

3.3.1 Preparation

실험에서는 각 시각화 기법이 오브젝트의 개수 그리고 렌더링 복잡성에 따라 어떻게 성능이 달라지는지 평가하기 위해 데이터 세트를 각각 다르게 정의하고 적용하였다.

시각화할 데이터 세트는 특정 장비에서 발생한 이벤트 데이터를 24시간 기준 1분 데이터를 화면에 표시한다고 가정 하여 1,440의 데이터를 하나의 데이터 세트로 정의한다.

- HTML 태그 기반 접근: HTML 기반 3D 그래프에서는 단순한 2D 요소를 이용해 데이터 차트를 구현하였다. 이 방식은 텍스처를 사용해 3D 효과를 시뮬레이션한다. 실험에서 겨우 2,880개의 작은 객체만을 렌더링했음에도 성능이 2.4fps까지 떨어졌고, 더 이상의 렌더링은 시스템이 이를 감당할 수 없었다. 이러한 성능 한계로 737,280개의 데이터를 시각화하려는 목표는 불가능 하였다.

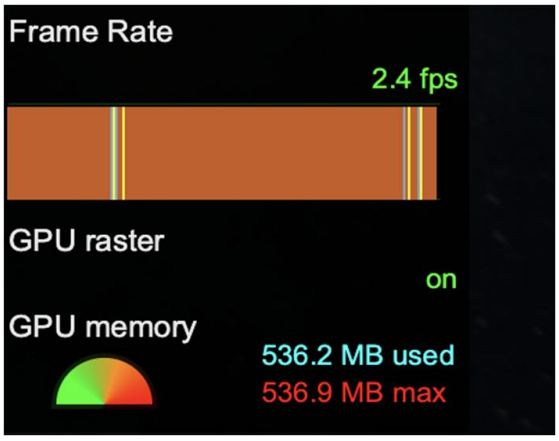

Fig. 9. Performance of HTML-Based 3D Rendering in Chrome Developer Tools

이로 인해, HTML 태그 기반 접근 방식이 복잡한 3D 객체의 시각화에 있어서 심각한 한계를 드러냈다. 결국, 이 방식으로는 대규모 데이터 시각화는 물론, 적은 양의 데이터조차도 효과적으로 처리할 수 없는 현실에 직면했다. 이러한 문제는 대규모 데이터를 처리하기 위한 보다 효율적이고 강력한 시각화 기술의 필요성을 더욱 부각시킨다. 시각화 결과는 Fig. 8과 같다.

- CSS 기반 <img> 태그 기반 접근: CSS 기반 3D 그래프는 Three.js의 THREE.CSS3DRenderer를 사용하여 구현되었다. 이 방식은 실제 3D 모델을 사용하지 않고, CSS의 <img> 태그를 3D 공간에 배치하여 렌더링하는 방법이다. 실험에서는 737,280개의 데이터를 시각화하려는 목표를 세웠으나, CSS 기반 접근 방식에서는 그 수의 일부인 512개의 이미지 박스만 배치했음에도 성능이 급격히 저하되어 10.8fps로 떨어졌다. 이는 성능 한계로 인해 737,280개의 데이터 시각화는 시도조차 불가능했음을 보여준다.

  CSS3DRenderer의 transform 속성을 사용해 각 이미지의 위치와 회전을 설정했으며, 한 지점으로부터 z축을 따라 확산되도록 배치하였다. 각 이미지는 z축 방향으로 거리를 두며 배치되었으며, 이를 통해 데이터 간의 구분을 직관적으로 나타내고자 하였다. 하지만 단순한 이미지 박스만으로도 성능이 크게 저하되는 문제가 발생했다. 특히 512개의 객체만으로도 이러한 성능 저하가 나타난 것은, CSS 기반 접근 방식이 대규모 데이터 시각화에 절대적으로 적합하지 않다는 것을 시사한다. 이와 같은 한계로 인해 더 복잡한 데이터 세트, 예를 들어 737,280개에 달하는 대규모 객체를 다루기에는 CSS 접근 방식이 불가능하다. 시각화 결과는 Fig. 9와 같다.

Fig. 10. Performance of CSS-Based 3D Rendering in Chrome Developer Tools

- Three.js Geometry Merge: Geometry Merge 방식에서는 여러 개의 작은 지오메트리(여기서는 Box Geometry(1, 1, 150))를 하나의 Geometry로 병합하여 렌더링하였다. 각각의 지오메트리는 무작위로

위치를 설정한 후 THREE.GeometryUtils.merge() 함수를 통해 병합된다. 이 방식의 목표는 개별 객체를 하나의 큰 지오메트리로 합침으로써, GPU의 드로우 콜(draw call) 수를 줄여 성능을 최적화하는 것이다. 실험에서는 1,440개의 객체를 병합하여 렌더링했으며, 그 과정에서 객체의 position이 무작위로 설정되었다. 그러나 이 방식은 많은 개체를 초기화하는 과정에서 메모리 사용량이 급격히 증가하는 문제가 발생하였으며, 특히 메모리 로딩 단계에서 성능 병목 현상이 나타났다.

- Three.js InstanceMesh 방식은 동일한 지오메트리를 여러 인스턴스로 복제하여 렌더링하는 방식으로, 코드에서는 new THREE.InstancedMesh(geometry, material, count)를 사용하여 737,280개의 인스턴스를 생성하고 이를 효율적으로 배치 및 렌더링하였다. 각 인스턴스의 위치와 크기는 실험에서 동적으로 설정되었으며, 이를 통해 대규모 데이터를 효과적으로 시각화하였다. 데이터를 표현할 boxGeometry를 생성하고 BasicMaterial을 적용한 이후 Mesh를 만든 후 새로운 Merge를 위한 Geometry에 병합하여 scene에 올리는 방법을 사용하였다. 세부적인 절차는 scene에 표현할 Material과 Geometry를 생성한다. 이후 computerVertexNormals() 함수를 사용하여 Geometry의 정접 법선을 계산하여 적절한 쉐이딩을 보장 할 수 있도록 한다. threejs에서 제공하는 THREE.Matrix4() 함수를 사용하여 변환 행렬을 위한 새로운 Matrix4 인스턴스를 생성하고, threejs에서 제공하는 THREE.InstancedMesh() 함수의 파라메터로 위에서 생성한 Material과 Geometry 사용하여 InstanceMesh를 생성한다. 시각화 결과는 Fig. 10과 같다.

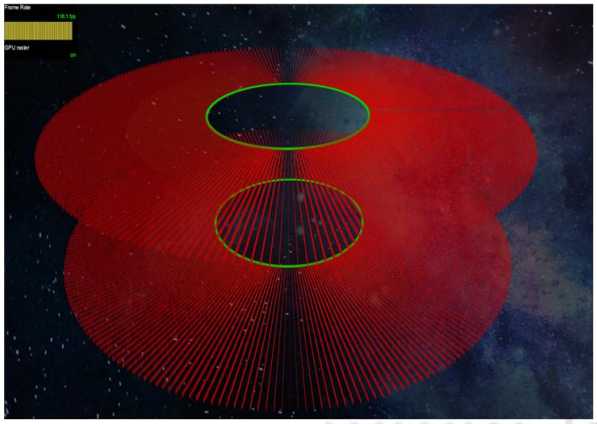

Fig. 11. InstanceMesh-Based 3D Rendered Screens

3.3.2 Performance Measurement

Chrome 개발자 도구의 프레임 렌더링 통계 기능을 사용하여 각 접근 방식의 프레임 수(fps)와 메모리 사용량을 측정하였다. 이 측정은 각 시각화 방식이 데이터를 실제로 어떻게 렌더링하며, 그 과정에서 어떤 자원을 소모하는지를 평가하는 데 목적이 있다.

3.3.3 Evaluation Criteria

프레임 수(fps)와 메모리 사용량은 시각화 기술의 성능을 평가하는 핵심 지표로, 실시간 시각화에서의 사용자 경험을 직접적으로 결정짓는 요소다. 프레임 수가 높을수록 시각화가 부드럽고, 반응 속도가 빠르며, 메모리 사용량이 적을수록 시스템의 안정성과 확장성이 높아진다.

4. Test Comparison

4.1 Test Result and Analysis

실험 결과 Table 4.에서 보여지는 바와 같이 초당 프레임 수(fps)와 메모리 사용량 측면에서 InstanceMesh 기술의 탁월한 성능 지표를 보여주었다. 예를 들어, 전통적인 기존의 HTML 기반 및 CSS <img> 기반 방식은 객체마다 독립적인 드로우 콜을 발생시켰다. 예를 들어, HTML 기반 방식은 2,880개의 객체를 렌더링할 때 각각 개별적으로 호출되어 드로우 콜이 과도하게 증가하며, 2.4fps로 성능이 급격히 저하되었다. CSS <img> 기반 방식은 512개의 객체에서도 10.8fps로 제한되었는데, 이는 각각의 객체를 처리하는 드로우 콜이 여전히 많아 GPU와 CPU 자원의 비효율적 활용으로 이어졌기 때문이다.

Three.js의 Geometry Merge 기술과 비교할 때, InstanceMesh는 동일한 지오메트리를 여러 인스턴스로 복제하여 단일 드로우 콜로 처리하여. 737,280개의 객체를 렌더링하면서도 단일 드로우 콜 방식이 GPU 부하를 줄이고, 110fps의 높은 프레임 속도를 유지하며, 메모리 사용량을 1MB 이하로 최소화하였다. 결론적으로 합리적인 메모리 사용량을 유지하면서 대규모 데이터 세트를 렌더링하는 데 탁월한 성능을 보여주었다.

Table 4. Test results performance indicators

Tech	Frame	GPU Mem	object count
HTML DIV-based 3D	2.4	High (<500MB)	2,880 (2sets)
CSS Image-based 3D	10.8	High (<500MB)	512
Geometry Merge	60	High (<500MB)	737,280 (512sets)
InstanceMesh	110	Low (1MB)	737,280 (512sets)

IV. Conclusions

3D 기반의 시각화에서 WebGL의 GPU 가속 렌더링과 같은 최신 기술은 성능을 향상시키는 데 중요한 역할을 하고 있으며, 최근 연구에서는 가상환경 교육 시스템[13], 브이월드 공간정보,[14] 다양한 과학 분야에서 고차원 데이터를 Three.js를 이용하여 실시간으로 시각화하는 데 그 장점을 활용하고 있다. 이러한 기술은 특히 물리학, 의학, 생물학 분야에서 실시간 시각화를 통해 복잡한 데이터 세트를 효과적으로 처리하는 데 중요한 기여를 하고 있다.[15]

대규모 데이터의 활용이 증가하면서, 대규모 데이터 시각화에 대한 요구 또한 커지고 있다. 그러나 현재의 기술은 대용량 데이터를 효과적으로 처리하는 데 있어 한계를 보인다. 본 논문에서는 InstanceMesh 기술을 이용하여 적은 드로우 콜의 기반으로 높은 fps를 유지하는 것과 메모리 사용량을 줄여 대규모 데이터 3D 시각화 처리를 하는 방법을 제안 하였다.

실험 결과, InstanceMesh 기술은 HTML 기반 방식(2.4 FPS)이나 CSS 기반 방식(10.8 FPS), 그리고 Geometry Merge 방식(60 FPS)에 비해 110 FPS의 성능을 기록하며, 대규모 데이터 처리에서 성능 우위를 보였다(Table 4 참조). 이를 통해, 기존 접근 방식에서는 불가능했던 737,280개의 데이터 시각화를 소비자용 하드웨어에서도 구현할 수 있음을 입증하였다.

그러나 고성능 GPU를 사용하는 환경에 대한 추가적인 실험은 포함하지 않았다. 따라서 향후 연구에서는 고성능 GPU를 활용한 연구를 통해 InstaceMesh 기술의 최대 성능과 확장 가능성을 평가하는 것이 중요하다. 또한 일반적으로 3D 기반 시각화에서는 빛의 종류와 개수, 사용된 material의 특성, post-effect(후처리 효과) 적용 여부 등 다양한 렌더링 요소가 성능과 결과물에 큰 영향을 미친다. 이러한 요소들은 InstanceMesh의 활용에 있어 제약으로 작용할 수 있다. 특히, InstanceMesh는 한 번의 드로우 콜로 여러 데이터를 동시에 렌더링하는 방식이므로, 각 인스턴스를 개별적으로 조작하거나 상호작용하는 데 한계가 있다. 이는 사용자가 개별 데이터를 세밀하게 조작하거나 상호작용하기 어려운 환경을 제공할 수 있다는 문제점을 야기한다.

따라서 향후 연구에서는, InstanceMesh의 성능 최적화를 유지하면서도 개별 인스턴스에 대한 조작 및 상호작용을 가능하게 하는 기술적 방법에 대한 추가적인 연구가 필요하다. 이를 통해 대규모 데이터 시각화에서 실시간 상호작용을 제공하며, 사용자가 각 데이터에 개별적으로 접근하고 조작할 수 있는 기능을 개선할 수 있을 것이다.

REFERENCES

[1] Park, Y., Cafarella, M,, and Mozafari, B. “Visualization-aware sampling for very large databases”, 2016 IEEE 32nd International Conference on Data Engineering (ICDE), pp. 755-766, May 16-20, 2016. DOI: 10.1109/ICDE.2016.7498287

[2] Masood, Mirza Danial. "Benchmarking Performance of CPU and GPU through 3D Rendering." National University of Computer and Emerging Sciences, July 2020, DOI: 10.13140/RG.2.2.35978.90564

[3] Seok-Hyun Kim, “Rendering States Changing Costs Reducing Techniquefor Real-time 3D Graphics”, Journal of the Korea Academia-Industrial cooperation Society, vol. 10, no. 8, pp. 1843-1849, 2009. DOI: 10.5762/KAIS.2009.10.8.1843

[4] Tae-Gwon Kim, Eun-Seok Lee, Byeong-Seok Shin, “An Acceleration Technique of Terrain Rendering using GPU-based Chunk LOD”, Journal of Korea Multimedia Society vol. 17, no. 1, pp. 69-76, January 2014. DOI: 10.9717/kmms.2014.17.1.069

[5] "WegGL Overview," The Khronos Group Inc., n.d. [Accessed: April 1, 2024]. Available: https://www.khronos.org/webgl/.

[6] "Three.js," n.d. [Accessed: March 30, 2024]. Available: http://threejs.org/. (Published: May 8, 2014).

[7] “matplotlib”, n.d. [Accessed: November 10, 2024]. Available: https://matplotlib.org/.

[8] Jae-beom Park, Huy-kang Kim, Eun-jin Kim,“Design and implementation of the honeycomb structure visualization system for the effective security situational awareness of large-scale networks”, Journal of the Korea Institute of Information Security & Cryptology, vol. 24, no. 6, pp. 1197-1213, 2014. DOI: 10.13089/JKIISC.2014.24.6.1197

[9] T. Taylor, D. Paterson, J. Glanfield, C. Gates, S. Brooks, and J. McHugh, "Visualization for Computer Security: Exploring NetFlow Data Using Activity Diagrams and Connection Bundles," 2009 Cybersecurity Applications & Technology Conference for Homeland Security, Washington, DC, USA, 2009, pp. 39-44. DOI: 10.1109/CATCH.2009.18

[10] Koutra, Danai, Di Jin, Yuanchi Ning, and Christos Faloutsos. "Perseus: An Interactive Large-Scale Graph Mining and Visualization Tool." Proceedings of the VLDB Endowment, vol. 8, no. 12, 2015, pp. 1924-1927, DOI: 10.14778/2824032.2824102

[11] Useok Kwak**, In-Ju Na****, Hyeonji Kim*, Kyeong-Jun Lee*, In Seo*, Wook-Shin Han***, “The State of the Art in Visualizing Large Graph Data“ Annual Conference of KIPS, vol. 2017, no. 4, 2017, pp. 802-803. DOI: 10.3745/PKIPS.y2

[12] K. T. Claypool and M. Claypool, "On frame rate and player performance in first person shooter games," Multimedia Systems, vol. 13, no. 1, pp. 3-17, 2007. DOI: 10.1007/s00530-007-0081-1

[13] Jeong, Ji Seong, Oh, Won-kun, and Yoo, Kwan-hee. "Design

and Implementation of WebGL-Based Physics Education System Based on Virtual." Journal of the Korea Computer Game Society, vol. 27, no. 2, 2014, pp. 41-48. Korea Computer Game Society, DOI: 10.22819/kscg.2014.27.2.005

[14] Lee, Ahyun, and Insung Jang. "Spatial Information Platform with VWorld for Improving User Experience in Limited Web Environment." Electronics, vol. 8, no. 12, 2019, article 1411, DOI: 10.3390/electronics8121411

[15] Franke, Loraine, and Daniel Haehn. "Modern Scientific Visualizations on the Web." MDPI Informatics, vol. 7, no. 4, 2020, article 37, DOI: 10.3390/informatics7040037

Authors

Chul-Hee Lee received the B.S. degree in Information Computer Science from Konyang University, Korea, in 2015, and he is currently pursuing the M.S. degree in Information Security at the Graduate School of Information Security, Sejong Cyber University, Seoul, Korea. Mr. Lee is a visiting professor at Dept. of Cloud Security in Korea Polytechnics, Daejeon Korea. His research interests include large-scale data visualization, 3D rendering technologies, and cloud-security.

Sang-Young Choi received his B.S. degree in Mathematics and M.S. degree in Computer Science, both from Hannam University in 2000 and 2003, and Ph.d degree in Interdisciplinary of Information Security from Chonnam National University in 2014. Dr. Choi is a assistant professor at Dept. of Cloud Security in Korea Polytechnics, Daejeon, Korea. His research interests are in security education, web security, network security and cloud computing security.