StrataVR: Out-of-Core Bricked Volume Streaming in Unity for Interactive Virtual Reality

Furqaan Nadeem
IDEAS
NIBIB, National Institutes of Health
Bethesda, Maryland, USA
nadeemf2@nih.gov

Xinrui Zou
IDEAS
NIBIB, National Institutes of Health
Bethesda, Maryland, USA
xinrui.zou@nih.gov

Brian Lee
Advanced Imaging and Microscopy (AIM) Resource
NIBIB, National Institutes of Health
Bethesda, Maryland, USA
brian.lee2@nih.gov

Harshad D. Vishwasrao
Advanced Imaging and Microscopy (AIM) Resource
NIBIB, National Institutes of Health
Bethesda, Maryland, USA
harshad.vishwasrao@nih.gov

Dorian B. McGavern
Viral Immunology and Intravital Imaging Section
NINDS, National Institutes of Health
Bethesda, Maryland, USA
dorian.mcgavern@nih.gov

Raisa Z. Freidlin
IDEAS
NIBIB, National Institutes of Health
Bethesda, Maryland, USA
raisa@mail.nih.gov

***Abstract*—Large volumetric datasets often exceed GPU memory, making responsive exploration in virtual reality (VR) difficult. StrataVR is a Unity-based out-of-core volume renderer that stores datasets externally and loads a small multiresolution working set as the view changes. To our knowledge, it is the first Unity-based out-of-core VR volume renderer demonstrated on a multiterabyte dataset. A C++ tool prepares bricks and metadata, while view-aware planning and asynchronous streaming keep relevant data in a fixed GPU pool. Coarser resident bricks remain visible until finer data arrives, providing continuous feedback and progressive refinement. The working system supports multichannel visualization, empty-region skipping, channel controls, cross sections, cutouts, and diagnostic views. The interactive e-poster will show how StrataVR turns a multiteratbye dataset into a responsive VR experience. Animations will explain how StrataVR works, while Unity and headset footage will show it in use.**

***Index Terms*—out-of-core rendering, bricked volume streaming, virtual reality, Unity, scientific visualization**

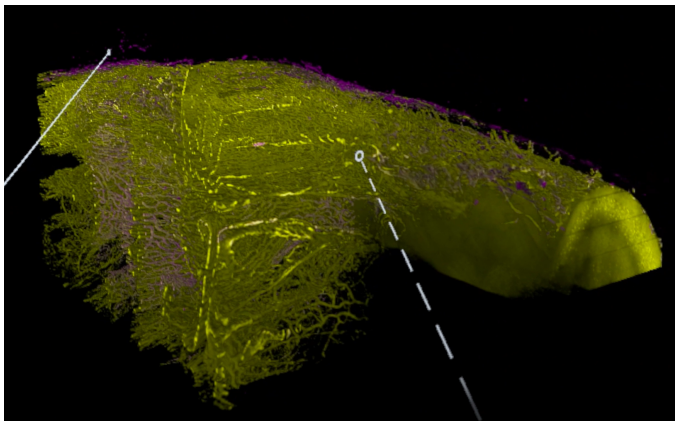


Fig. 1. StrataVR displaying a large streamed volume in VR.

I. Interactive E-Poster Summary

StrataVR is a Unity-built volume renderer pipeline for exploring large volumetric datasets in virtual reality (VR). It uses out-of-core rendering, where the dataset is stored outside the Unity application, divided into multi-resolution three-dimensional bricks, and only the bricks needed for the current view are streamed through CPU and GPU memory.

The project builds on established large-volume methods, including multiresolution brick working sets, ray-guided selection, and page-table addressing [1], [2]; Tuvok provides a scalable example [3]. Related immersive systems include BorgVR for out-of-core rendering and syGlass and iDaVIE for interactive volume analysis [4], [5], [6]. StrataVR's contribution is to connect these ideas in a single Unity VR pipeline for preprocessing, bounded streaming, fixed GPU residency, and shader-level fallback.

This pipeline starts with a C++ builder that reads up to four aligned image channels. The builder first checks their dimensions and data types and then constructs a hierarchy of levels of detail (LODs). LOD 0 preserves the original samples, and each LOD after that halves the voxel dimensions along the X, Y, and Z axes, resulting in eight times fewer voxels. Each LOD is divided into fixed-size bricks. Neighboring voxels are stored as a border so trilinear filtering remains continuous

across brick boundaries without visible seams. Bricks can be independently compressed and retrieved, so the complete dataset never needs to be preloaded.

While packing bricks and downsampling the data, the builder records metadata for each brick. This includes value ranges; occupancy, defined as the fraction of voxels above a background threshold; histograms; active signal bounds; and parent error, which estimates the detail lost when a coarser parent is used. A required max-only macrocell hierarchy divides each brick into smaller regions and stores their maximum values. The renderer uses these summaries to reject bricks that cannot affect the image and to skip empty intervals inside resident bricks before sampling voxels.

The dataset folder can be selected in the Unity VR application, which reads and validates the hierarchy, indices, and required metadata. Bounded caches keep planning memory controlled as the dataset grows. During runtime, a persistent working-set planner periodically and asynchronously decides which bricks are needed for the current viewing state. It considers what both eyes can see, how much detail a brick would contribute on screen, and its distance from the viewer. The planner first reserves visible coarse coverage. For finer bricks, priority begins with projected detail and is adjusted using visible signal, parent error, current residency, and whether the brick is already loading.

The planner keeps coarse coverage available while finer data loads so that when the user turns toward a new part of the volume, a coarse brick is shown instead of nothing. As finer children become available, they progressively refine the coarser parent. Coverage bricks are protected from eviction until suitable replacements are ready. The system also keeps a larger history of recently viewed high-detail bricks, reducing unnecessary reloads during camera turns.

Brick requests pass through a bounded priority queue, persistent background file reading, optional decompression, and a bounded CPU upload queue. The main thread transfers small batches into fixed-capacity three-dimensional GPU atlases. The streamer reuses a brick if it is already resident; otherwise, it assigns an unused GPU slot or replaces lower-priority data. Loading more of the dataset changes which bricks are resident without increasing the pool's memory footprint.

A page table stores each resident brick's logical-to-physical mapping and is updated only after its upload completes. The table translates each logical LOD coordinate into a physical atlas location. If a requested brick is absent, the shader moves up the hierarchy until it finds a resident parent. This lets the same view contain full-resolution regions, coarse regions, and regions that are still loading.

For each eye, the shader intersects a viewing ray with the volume bounds, applies any active cross sections or cutouts, chooses an LOD, and looks up bricks while marching through the remaining interval. Active bounds and max-only macrocells advance the ray past empty regions. Direct volume rendering spaces samples uniformly in the selected LOD's voxel units, using at most half a voxel per step. A matched phase keeps the two eyes coherent, while an adaptive iteration ceiling limits GPU work without changing the spatial sampling lattice.

Together, these mechanisms bound CPU and GPU memory use, preserve immediate visual feedback during head motion, and progressively improve image quality as data arrives. An evaluation test was done using the multiterabyte microscopy corpus shown in the poster; the first visible brick arrived in 9.94 s. Disk reads reached 285.76 MiB/s, decoded data reached 380.54 MiB/s, and uploads reached 120.76 MiB/s. Median request latency was 1.89 s and median VR frame time was 14.41 ms; P95 values were 7.44 s and 100.31 ms. Table I records the full demonstrated configuration.

ACKNOWLEDGMENT

The example microscopy dataset used in this work was prepared by Harshad D. Vishwasrao of the Advanced Imaging and Microscopy (AIM) Resource, National Institute of Biomedical Imaging and Bioengineering, for the laboratory of Dorian B. McGavern at the National Institute of Neurological Disorders and Stroke. StrataVR's original rendering foundation was based on Matias Lavik's open-source UnityVolumeRendering project [7]; the present system has since been substantially reworked and extended for out-of-core bricked streaming and VR. OpenAI Codex assisted with code development, testing, text editing, and layout; the authors reviewed and approved the resulting software and text [8].

TABLE I
FULL 15 TB CORPUS EVALUATION (SINGLE RUN)

Test protocol: 5.0 s warm-up and 60.2 s capture. Storage and runtime configuration materially affect these results.

Measurement	**Recorded result**
Dataset	Example microscopy corpus, prepared by Harshad D. Vishwasrao (NIBIB AIM) for Dorian B. McGavern's laboratory (NINDS); 15 TB source; $13{,}023 \times 22{,}792 \times 9{,}695$; 3 channels; 19.352 TiB built folder
Software	StrataVR application version 0.1; native builder 0.9.0
Hardware	Intel Core Ultra 9 290HX Plus; 64,980 MiB RAM; RTX 5090 Laptop GPU (24,137 MiB, D3D11); Meta Quest Pro via Quest Link; 58.2 TiB NTFS RAID-0
Preprocessing	215,748.66 s (59.93 h); Gaussian downsampling
Brick size / LODs	$128 \times 128 \times 64$ inner ($130 \times 130 \times 66$ stored); 7 LODs
GPU atlas	8,169.4 MiB; 960 allocated slots, 912 usable
First visible brick	9.942 s after `LoadDataset` began
Read / decoded / upload	285.76 / 380.54 / 120.76 MiB/s
Request latency	1,887.87 ms median; 7,443.69 ms P95
VR frame time	14.41 ms median ($\approx$69 FPS); 100.31 ms P95
Telemetry	1,581 frame and 772 request samples; 9,086 events; 0 dropped

REFERENCES

[1] J. Beyer, M. Hadwiger, and H. Pfister, "State-of-the-art in GPU-based large-scale volume visualization," *Computer Graphics Forum*, vol. 34, no. 8, pp. 13–37, 2015, doi: 10.1111/cgf.12605.

[2] T. Fogal, A. Schiewe, and J. Krüger, "An analysis of scalable GPU-based ray-guided volume rendering," in *Proc. IEEE Symp. Large-Scale Data Analysis and Visualization*, pp. 43–51, 2013, doi: 10.1109/LDAV.2013.6675157.

[3] T. Fogal and J. Krüger, "Tuvok, an architecture for large scale volume rendering," in *Proc. Vision, Modeling, and Visualization*, pp. 139–146, 2010, doi: 10.2312/PE/VMV/VMV10/139-146.

[4] C. Hrycak and J. Krüger, "An investigation of the Apple Vision Pro for out-of-core ray-guided volume rendering with BorgVR," in *Proc. Vision, Modeling, and Visualization*, 2025, doi: 10.2312/vmv.20251236.

[5] S. Pidhorskyi, M. Morehead, Q. Jones, G. Spirou, and G. Doretto, "syGlass: Interactive exploration of multidimensional images using virtual reality head-mounted displays," arXiv:1804.08197, 2018, doi: 10.48550/arXiv.1804.08197.

[6] A. Sivitilli *et al.*, "iDaVIE v1.0: A virtual reality tool for interactive analysis of astronomical data cubes," *Astronomy and Computing*, vol. 56, Art. no. 101109, 2026, doi: 10.1016/j.ascom.2026.101109.

[7] M. Lavik, "UnityVolumeRendering," GitHub repository. [Online]. Available: https://github.com/mlavik1/UnityVolumeRendering

[8] OpenAI, "Codex," *OpenAI Developers*. [Online]. Available: https://developers.openai.com/api/docs/guides/code-generation