LARGE DYNAMIC VOXEL SCENES USING SPARSE VOXEL OCTREES

by

NICOLAS FEDOR
URN: 6683787

A dissertation submitted in partial fulfilment of the
requirements for the award of

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

May 2025

Department of Computer Science
University of Surrey
Guildford GU2 7XH

Supervised by: Joey Sik Chun Lam

I declare that this dissertation is my own work and that the work of others is acknowledged and indicated by explicit references.

Nicolas Fedor
May 2025

Abstract

Write a summary of the work presented in your dissertation. Introduce the topic and highlight your main contributions and results. The abstract should be comprehensible on its own, and should no contain any references. As far as possible, limit the use of jargon and abbreviations, to make the abstract readable by non-specialists in your area. Do not exceed 300 words.

Acknowledgements

Write any personal words of thanks here. Typically, this space is used to thank your supervisor for their guidance, as well as anyone else who has supported the completion of this dissertation, for example by discussing results and their interpretation or reviewing write ups. It is also usual to acknowledge any financial support received in relation to this work.

Contents

List of Figures

List of Tables

Glossary

P	Placeholder

Abbreviations

SVO	Sparse Voxel Octree
SVDAG	Sparse Voxel Directed Acyclic Graph
DDA	Digital Differential Analysis
LUT	Look-Up Table
LOD	Level of Detail
RLE	Run-Length Encoding
SFC	Space Filling Curve
FPS	Frames Per Second

Chapter 1

Introduction

1.1 Problem Statement

Rendering large-scale voxel worlds with dynamic scenes presents significant challenges due to the vast amount of data required to to represent 3D environments. Voxel-based rendering, where the scene is composed of volumetric pixels (voxels), is an attractive alternative to traditional triangle-based rendering for its ability to represent complex geometry and volumetric effects; however, the scalability of voxel-based rendering is limited by the amount of memory required to store the voxel data, and often times the computational cost of updating hyper-compressed voxel data structures.

Sparse Voxel Octrees (SVOs) (Laine & Karras 2010) are a popular data structure for representing voxel scenes, as they provide a compact representation of the scene by storing only the occupied voxels in a tree structure. Octrees, or nodes, can be dynamically subdivided to increase the resolution of the scene, and provide a straighforward way of introducting Level of Detail (LOD). Octrees are also well suited to ray tracing, as they provide a natural way of separating a scene into bounding boxes that can be quickly tested for intersection (Ize 2009). SVOs allow for high compression ratios, and performant ray tracing, for the rendering of large (typically in the millions) of voxels.

SVOs are not without their limitations, however. Updating an SVO can be a straightforward process as only the affected nodes need updating; however, an SVO is usually further compressed into a format better suited for GPU rendering, using techniques such as run-length encoding (RLE) (Eisenwave n.d.*a*) and space filling curves (SFC) (Eisenwave n.d.*b*). Updating these compressed formats, and transferring the new data to the GPU, can be computationally expensive (Crassin 2012). This makes SVOs less suitable for dynamic scenes where voxels can be added, removed, or modified frequently.

1.2 Aims and Objectives

The primary goal of this dissertation is to investigate the feasibility, and identify potential techniques and approaches that could improve the performance of using SVOs for rendering large-scale dynamic voxel scenes. The aims of this dissertation can be broken down into the following objectives:

1. Investigate current techniques for constructing and rendering SVOs.
2. Design a system for updating SVOs, on the GPU, to allow for large dynamic voxel scenes.
3. Develop a renderer, using Vulkan, that addresses the challenges of data transfer between the host and device of large SVOs.
4. Evaluate the performance of the system, and identify potential areas for improvement. See Section 1.2.1.
5. Investigate potential techniques for further improving the performance of SVOs for dynamic scenes.

1.2.1 Performance Metrics

The performance of the system will be evaluated using the following metrics:

Frames Per Second (FPS) The number of frames rendered per second. At a minimum for real-time rendering, and for interactive applications such as games, the system should be able to render at a consistent 30 FPS.

Since FPS isn't always a good indicator of performance, frame time should also be considered and is measured as the time taken to render a single frame including the update and render time. For a consistent 30 FPS, the frame time should be an average of 33.33ms.

Memory Usage The amount of memory used by the SVO on the GPU. The amount of memory an SVO uses is heavily dependant on the resolution of the scene, and the exact compression techniques used. For a 512^3 voxel scene, with a 9 level SVO, the memory usage should be less than 1GB (Crassin 2012, Laine & Karras 2010).

Data Transfer Time The time taken to transfer the updated SVO data between the host and device. TODO

Construction and Update Time The time taken to update the SVO on the GPU. Using the same 512^3 voxel scene, the construction time should be less than 100ms (Crassin 2012, Laine & Karras 2010); the construction of an SVO includes updating the tree structure with new voxels.

Rendered Voxels The number of voxels rendered in the scene. A 512^3 voxel scene consits of 134,217,728 voxels; having a fully populated octree would not be feasible and would mean a majority of the voxels are obsured. Using a suitable voxel scene, at least 1,000,000 visible voxels should be rendered at the 30 FPS target.

1.3 Scope and Limitations

A fully-featured voxel renderer is out of scope for this dissertation, this includes features such as:

Lighting and shading effects Effects such as ambient occlusion, refractions, reflections, and global illumination are not considered in this dissertation as they are renderer specific and not the focus of this dissertation.

Transparency and volumetric effects Rendering transparent voxels, or volumetric effects such as fog, smoke, or fire, add additional complexity to the rendering process, which is not the focus of this dissertation.

Identifying new techniques for SVO compression The focus of this dissertation is on the performance of updating SVOs, not on the compression techniques used to store the SVO on the GPU.

Optimizing the rendering engine A simple Vulkan renderer will be built to demonstrate the performance of the SVO update system, but the focus will be on the SVO update system itself.

1.4 Thesis Outline

1. **Introduction** - A brief introduction to the problem statement, aims and objectives, and the scope and limitations of the dissertation.
2. **Literature Review** - A review and discussion of the current techniques used for rendering voxel scenes, with a focus on Sparse Voxel Octrees; and the history of voxel rendering.
3. **System Design** - A detailed explanation of the system design, including the data structures used, the algorithms for updating the SVO, and the renderer design.
4. **Development** - A more in-depth look at the implementation of the system, as set out in Section **??**.
5. **Testing and Evaluation** - A discussion on the performance of the system, and the results of the evaluation against the metrics set out in Section 1.2.1.
6. **Conclusion** - A summary of the dissertation, including the findings, limitations, and potential future work.

Bibliography

Crassin, C. (2012), Dynamic sparse voxel octrees for next gen real time rendering, Technical report, NVIDIA Research.
URL: https://www.icare3d.org/research/publications/Cra12/04_crassinVoxels_bps2012.pdf

Eisenwave (n.d.*a*), 'Run-length encoding'. Part of the "Voxel Compression Documentation", found at the same URL.
URL: https://eisenwave.github.io/voxel-compression-docs/rle/rle.html

Eisenwave (n.d.*b*), 'Space-filling curves'. Part of the "Voxel Compression Documentation", found at the same URL.
URL: https://eisenwave.github.io/voxel-compression-docs/rle/space_filling_curves.html

Ize, T. (2009), Efficient Acceleration Structures for Ray Tracing Static and Dynamic Scenes, PhD thesis, The University of Utah.

Laine, S. & Karras, T. (2010), Efficient sparse voxel octrees - analysis, extensions, and implementation, Technical report, NVIDIA Research.
URL: https://research.nvidia.com/sites/default/files/pubs/2010-02_Efficient-Sparse-Voxel/laine2010i3d_paper.pdf