3D VISUALIZATION OF LARGE ASTRONOMICAL DATA CUBES

The Problem

A modern radio astronomy telescope - SKA (the Square Kilometer Array)

With the development of larger and larger modern radio astronomy telescopes, astronomical data cubes are rapidly increasing in both size and complexity. Most astronomical visualization tools are simply not equipped to deal with these large data sets (> 1TB). One notable exception is CARTA, which is able to handle large cubes by implementing a scalable client-server solution. However, CARTA lacks any sort of 3D rendering capabilities, which is important in the analysis of astronomical data.

Objectives

Working alongside the lead developer of CARTA, we set the following objectives:

- Create a 3D visualization prototype application that would be able to render very large data cubes at reasonable speeds when it has access to the necessary computational hardware – specifically large enough storage and powerful GPU(s).
- The prototype should allow users to view and interact with the cubes on a commodity workstation in a lag-free manner, but also be detailed enough to allow for scientific visual analysis.

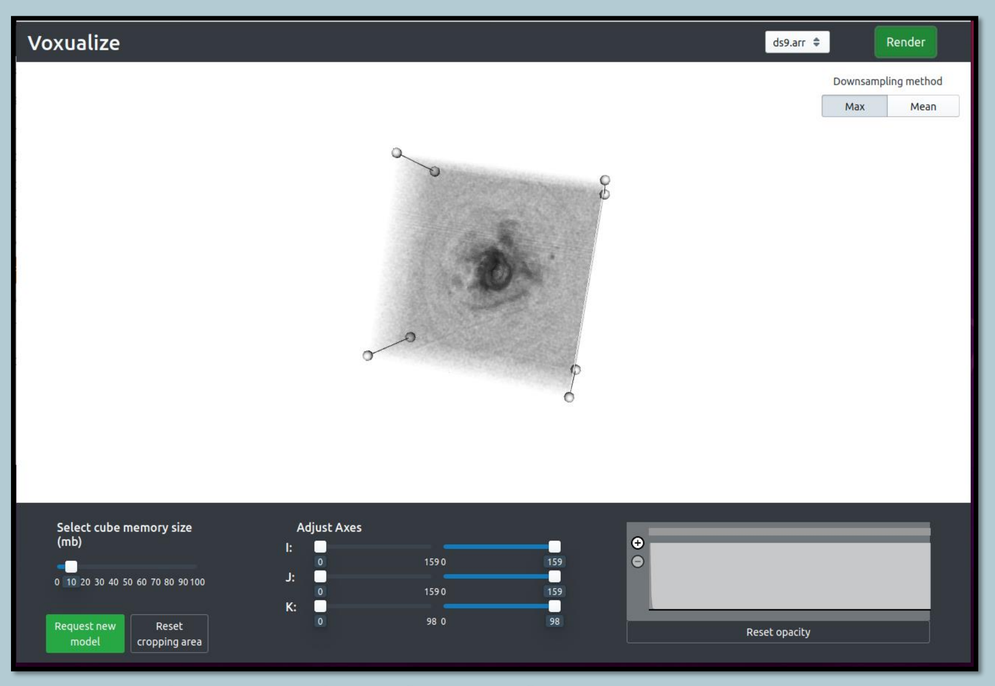
Our Hybrid-Rendering Approach

We opted to build a web application with a client-server architecture to satisfy our objectives. The server was written in C++, the client in JavaScript and built with React.js, and the communication protocol with Google's Remote Procedure Calls (gRPC).

It implements a hybrid-rendering technique that renders a low-fidelity model on the client side to allow for smooth interaction.

The server then renders a high-fidelity model and continually captures high quality 2D images that are streamed to the client and overlay the low-fi model.

This allows for a more detailed analysis.



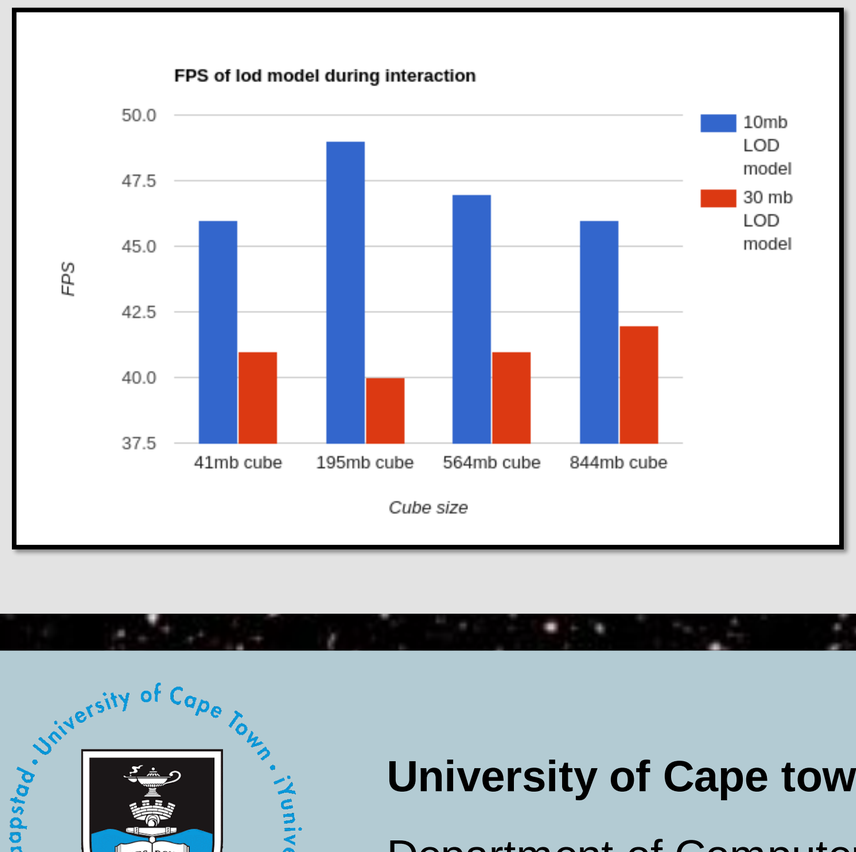
The user interface of our prototype.

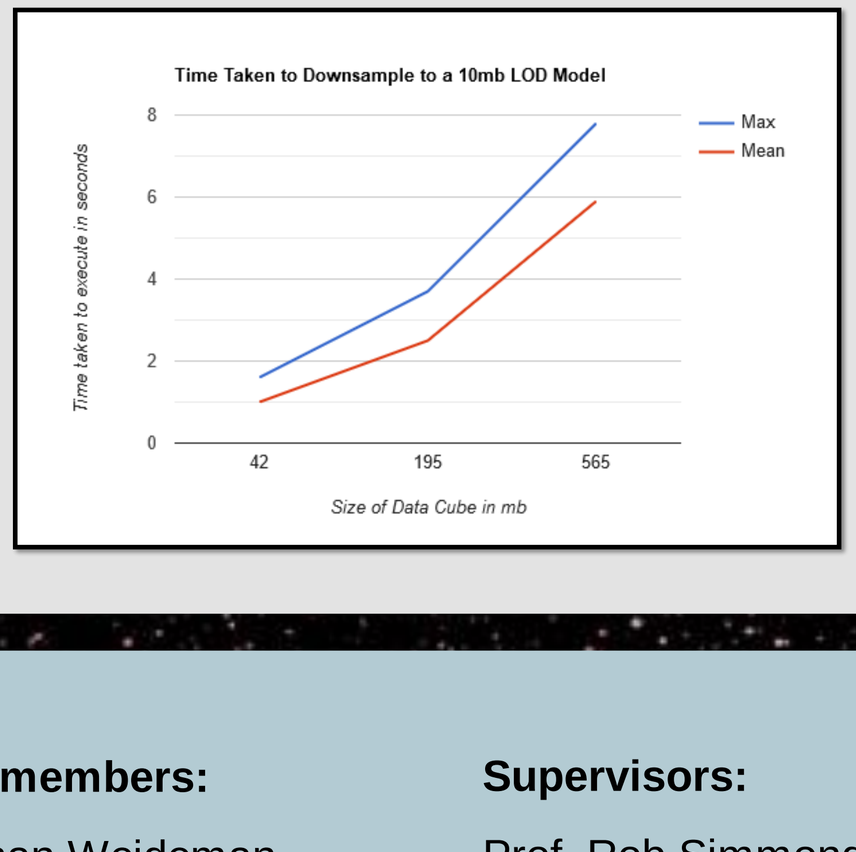
Results and Conclusions

Our prototype achieved our goals relatively well, boasting more than adequate speeds when interacting with the down sampled model. We were able to render and interact with a data cube close to a GB in size at ~40 FPS (frames-per-second).

The graph bottom right shows the time taken to generate a 10mb level-of-detail model from cubes of various sizes. The time taken increased linearly as the size of the cube increased. This function proved to be the bottleneck of our system, and we conclude that future applications should implement the down-sampling in parallel on the CPU or even on the GPU for very large cubes.

Overall we think that our approach provides a good middle ground between performance and detail and warrants further tweaking and exploration.





University of Cape town

Department of Computer Science

Email: dept@cs.uct.ac.za

Team members:

Jonathan Weideman

Shuaib Parker

Supervisors:

Prof. Rob Simmonds

Dr. Angus Comrie