Visualisation of Galaxy Simulations - Daniel Beard

Abstract

The distribution of neutral hydrogen is a useful tracer of the underlying dark matter across cosmological distances and the key to understanding star formation and galactic growth. In this paper we present a method of converting HDF5 files containing hydrogen masses from cosmological simulations into a true 3D voxel volumetric representation. The results, conducted in a High Performance Computing situation, show that calculating a 256 cubed volume from an HDF5 file takes approximately 20 minutes to complete. The algorithm is embarrassingly parallel so each volume from a time series can be calculated on individual processors, greatly reducing the overall compute time. A number of different methods of visualizing the volumetric data are explored including spherical and fisheye projection.

Introduction

The distribution of neutral hydrogen in the Universe is of fundamental importance as both a tracer of the underlying Dark Matter across cosmological distances and the basic fuel for star formation and galactic growth. A ground-breaking new radio telescope in Western Australia, the Australian Square Kilometer Array Pathfinder, will revolutionize our view of this key quantity. To both predict and understand the results from this facility we need high resolution simulations of significant volumes of the Universe. Just such a series of simulations, with the accompanying data-cubes, have been created in an international collaboration between researchers in the Netherlands, UK and ICRAR. This project involves the visualization of these simulation in a manner that is able to cope with the large spatial dynamic ranges together with the orders of magnitude variations in the signal of the neutral hydrogen. The outcome will be a series of high resolution images and videos that explore different visualisation techniques which will allow researchers to better comprehend the simulation and better present the simulation output to an audience.

Program Structure

- Create and allocate data structures to hold data values
- Read data files
- Scale Points and Smoothing arrays to dimensions of volume
- Compute Density
    - Loop through point data
    - Loop through voxels within a range
- Compute HI Density
    - Loop through point data
    - Loop through voxels within a range
- Write Density Raw file
- Write HI Density Raw file
- Write POVray df3 file (optional)

For each point, I loop over the voxels and calculate the density. The quantity of interest is the neutral hydrogen however the total gas density must be calculated first.

HDF5 File Structure

The simulation data is stored in a binary file format called hdf5. HDF5 is a flexible and efficient method of storing and operating on extremely large data sets. It is faster to access the information from the hdf5 file directly using the API provided. The version of hdf5 used for this project was the 1.8.2/gcc-4.2.0. The description below shows the abstract structure of the hdf5 files that were used in this project, however my project only used the HImass, Mass, Position and Smooth values. Reading the HDF5 files directly as opposed to using H5dump then extracting the variables into separate text files reduces read time from 30 minutes to less than 2 minutes for a typical HDF5 file with 2400000 particles. The HDF5 file has the following format:

HDF5 Data Format

```
Particles
     {
          Gas
          {
              HIImass     - (Ionised Hydrogen Mass)    - Units (Msol)
              HImass      - (Neutral Hydrogen Mass)    - Units (Msol)
              Htwomass    - (Molecular Hydrogen Mass)  - Units (Msol)
              Mass        - (Total gas mass)           - Units (Msol)
              Position    - (Coordinates)              - Units (Mpc - Megaparsec)
              Smooth      - (SPH Smoothing Length)     - Units (Mpc - Megaparsec)
              Velocity    - (Velocity in km/s)         - Units (Km/s)
          }
     }
```

Smoothed Particle Hydrodynamics (SPH)

Smoothed Particle hydrodynamics is a computational method for simulating fluid flows and works by calculating the weighted average of neighbouring particles. Smoothed Particle Hydrodynamics is used in this project to determine the true 3D density of HI hydrogen gas based on a set of point coordinates and mass data. Smoothed Particle Hydrodynamics has the effect of taking a series of discrete points and 'smoothing' them to form a cloud like representation. It is used extensively in astronomy because its nature allows it to simulate fields with orders of magnitude difference such as the Hydrogen gas density.

The Points and Smooth array values are scaled to the dimensions of the volume before the density calculation is performed. This is accomplished by finding the minimum and maximum values of the Points values in all dimensions (x,y,z). The scale factors are then calculated as the minimum point subtracted from the maximum point in all dimensions. The final scale factor is then the volume size divided by the largest of these three values. The points are then multiplied by the scaling factor to give values within the range 0 to volume size.
Each point in the data set has a smoothing length associated with it. When calculating the density of a single stored point, the value from the Mass array is weighted by the smoothing kernel, W, which is shown in Figure 1. As the radius, r, approaches the smoothing length, ξ , the contribution from that particle decreases. Once the radius is greater than the smoothing length the contribution from that particle is 0. The density of a voxel can then be thought of as a summation of contributions from points within a range. Figure 2 shows the contributions of points to a voxel.

$$W(r,\xi)=\frac{8}{\pi\xi^3}\begin{cases}1-6\left(\frac{r}{\xi}\right)^2+6\left(\frac{r}{\xi}\right)^3 & 0\le\frac{r}{\xi}\le 0.5\\ 2\left(1-\frac{r}{\xi}\right)^3 & 0.5<\frac{r}{\xi}\le 1\\ 0 & \frac{r}{\xi}>1\end{cases}$$

Figure 1.

The smoothing function shown below is a SPH softening spline kernel [1]. "Is used to calculate the physical quantities of interest, with decreasing contributions from particles as a function of distance" [2]. The weighting function uses the distance, r, between the current voxel and the current point taken from the Points array contained in the hdf5 file. The smoothing length, ξ, is defined in the Smooth array from the hdf5 file.

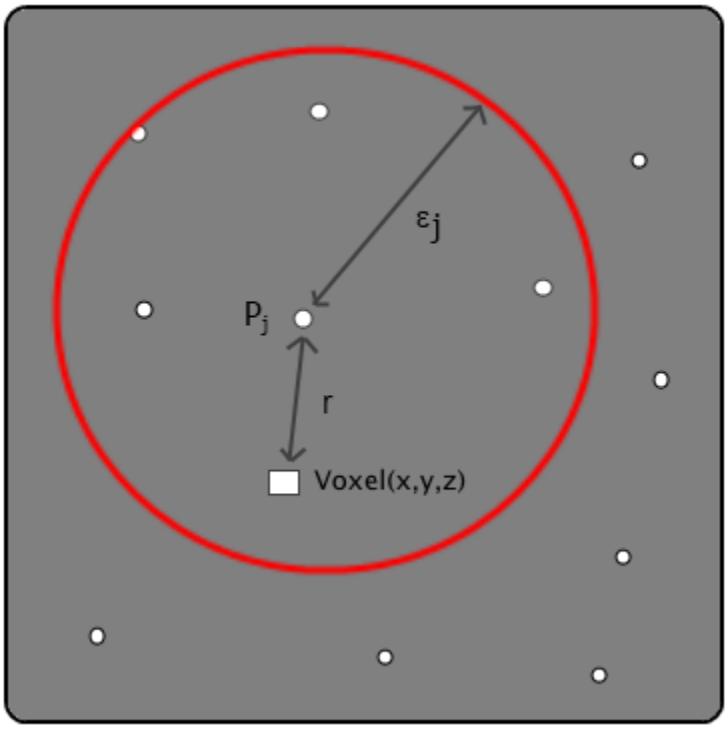

Figure 2.

$$Voxel\,(x,y,z)=m_j\,W(r,\xi_j)$$

Figure 3.

For each particle in the points array, we only loop over the voxels that are within a maximum radius. This reduces the total iterations significantly. The maximum radius is calculated as: 1+ ceil(smooth[ip]). This algorithm scales by a factor of 8 as the voxel volume increases. For example a volume resolution of 512^3 takes 8 times as long as a volume resolution of 256^3. This algorithm also scales linearly by number of points. For example twice as many points takes twice as long to execute.

Calculating HIDensity values is dependent on total gas density, so the total gas density has to be calculated first. Then using the total density calculation results, the HI gas density volume can be calculated. The HI gas density is just the HI mass, contained in the array 'HImass', divided by the voxel volume. One therefore multiplies this HI density by the initial mass array, dividing by total density at that point and weighting by the smoothing kernel [2]. The HI density is then weighted by the voxel volume but because the simulation data has been scaled to the voxel volume size, the volume of a single voxel will always be 1^3 so we can effectively ignore this in the equation.
This method gives the true HI 3D density. The equation is shown in Figure 4.

$$HIdensity = \frac{HImass_i}{Voxel\ Volume} \frac{Mass_i}{P_i} W(r, \xi)$$

Figure 4.

Saving data to volumetric raw files

A raw file has a simple structure and can be imported into a range of volume visualization tools. I am using an open source, cross platform volume rendering tool called Drishti. This raw file format is compatible with Drishti. The header consists of 1 byte and specifies the voxel type. The value that is used for this project is 8 which denotes a float – 4 bytes per voxel [3].

The three values after the header denote the size of the grid dimensions of the voxel data. These are written as 4 byte integers. The voxel data is then stored with the voxel data size depending on the header. In this case floating point values are stored meaning that each voxel consists of 4 bytes. Figure 5 shows the internal structure of a raw file.

The native file format of Drishti is now based on the raw file structure. Further work on this project includes writing directly to Drishti's native pvl.nc files to avoid having to convert the raw files. The only difference between the raw files currently and Drishti's native file format is that the voxel data is stored as a single byte per voxel. For the program to write directly to pvl.nc files the floating point voxel values need to be converted to unsigned char values and an xml file produced.
This is one inherent disadvantage of using Drishti, the range of different voxel values has to be within 0-255. As an attempt to get around this limitation, the program takes the logarithmic voxel values and stores these instead of the true values. This allows for easier visualisation of the volume.

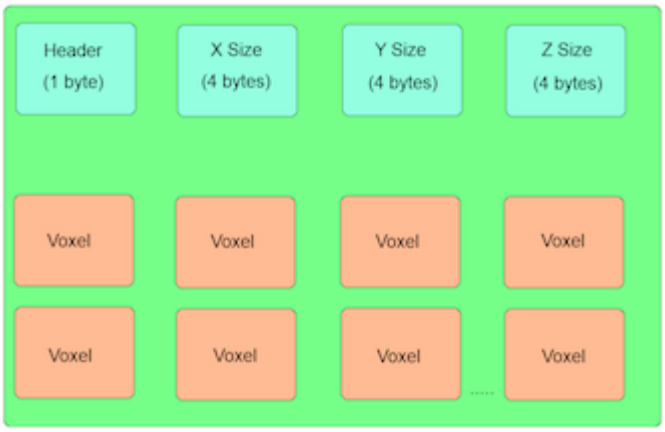

Figure 5.

Periodic Boundary Conditions

When calculating the density values of each voxel, the voxels on the outer edge of the volume will have less influences acting upon them because there are less particles neighboring them. The solution to this is to create a volume that has toroidal bounds also termed periodic boundary conditions. The distance is calculated for the x, y and z values separately, then using the absolute value of the result we compare this to the volume size. If any of the x,y,z distances are greater than half of the volume size, the distance is calculated as distance – nvol / 2. This results in the shortest distance between two points and allows seamless tiling of the volume. The periodic boundary conditions are only applied to the OWLS data set, the GIMIC data set does not have periodic bounds.
Figure 6 illustrates how this calculation works.

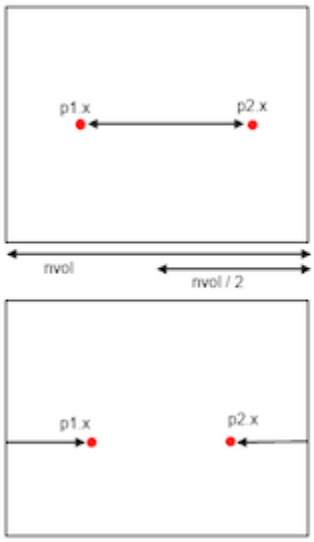

Figure 6.

2D Occlusion Spectrum

When using Drishti, volumes are visualised with the use of transfer functions. A transfer function is a 2d histogram where the first axis is the voxel value and the second axis is gradient magnitude. Drishti has the option of using a method called 2D Occlusion Spectrum to produce a 'smooth' file that is used to produce different transfer functions for a single volume. Instead of using the gradient magnitude, the filtered value is used for the second parameter of the 2D transfer function.

The method is implemented by performing a low pass filter on the volume. This method is particularly useful for volumes with a high amount of noise and lets the user isolate certain structures. The method is implemented in Drishti although currently it requires the filter size to be set for each volume. This can be extremely time consuming when processing a large number of volumes. A solution to this limitation was to create a program that operates on multiple volumes while only having to specify the filter size once.

Drishti stores voxel data in .pvl.nc files which have the same structure as the .raw files - first byte is 0, three integers that specify the volume size (x size, y size, z size), followed by the actual voxel data. When the 2D Occlusion Spectrum is calculated in Drishti, the .pvl.nc file is read in and a low pass filter is applied to the volume. The result is then stored in a .pvl.nc.smooth file.

I have written a program that mirrors the 2D Occlusion Spectrum functionality in Drishti. My program produces the .smooth file by reading the .pvl.nc file. The low pass filter is a grid based on the filter size that smoothes the volume by taking a mean of a voxel based on its neighbours. The grid size is calculated as 2*filtersize+1, this ensures that the grid is always an odd number. In my program the filter size is always 2, giving a 5x5x5 grid. The structure of the program is as follows.

```
 - Read in original .pvl.nc file
 - Loop through all voxels calculating a mean based on the surrounding voxels
 - For edge cases discard the values that lie outside the volume
 - Take the difference in value of the original voxel and the smoothed value clamping the values
between 0 and 255
      - total[i][j][k] = Clamp(0, 128 + (values[i][j][k] - mean[i][j][k]), 255);
 - Write the result to file using unsigned chars to store the voxels
```

A clear advantage of mirroring this functionality is that smoothing volumes can calculated in parallel, greatly reducing total calculation time. This difference in calculation time is significant for a large number of volumes.

Alternate Visualisation Methods

Cube maps

An interesting feature of Drishti is that it allows the user to export the scene as a cube map. A cube map is a view of a scene in all directions from the camera's position. The ability to export to cube map allows different visualisation techniques to improve the interactivity and extended view of a scene. Cube maps are rarely used to visualise a scene in their raw form. In this case, cube maps were a method of producing alternate visualisation methods such as fisheye and spherical projections. Six different images are saved with the prefixes as shown in figure 7.

Abbrev.	Image Face
f	front
r	right
l	left
t	top
b	back / behind
d	down

Figure 7.

Fisheye projection

Fisheye projections are usually used in planetariums or IMAX for projecting a video onto a curved surface. The benefit of a fisheye projection is that the video has a wider field of view than a normal video. This means that it is easier to visualise more of the volume at any one time. A fisheye projection was created and successfully tested at the upright half dome at WASP at the University of Western Australia. When flying through a data set, a fisheye projection can give a sense of 3D because the image surrounds the viewer. The field of view value used in this project was 180 degrees giving a standard fisheye projection. It is possible to have elliptical fisheye projections by changing the aspect ratio.

When creating a fisheye projection using Drishti, cube maps of each frame must be rendered then converted. For this project the conversion was calculated using Paul Bourke's cube2dome program [4]. When rendering with POVray, the fisheye projections can be calculated directly without the need to render to cube maps first. Rendering with POVray provides faster calculation speed and takes less disk space but the tradeoff is a more difficult visualisation than using Drishti.

Spherical Projection

Spherical projections are similar to fisheye projections in the sense that they are designed to be projected onto a curved surface. The difference between the two is that a spherical projection is designed to give a full view of a scene instead of just a wider field of view. The six faces of a cube map are warped to provide a single flat video that gives a full view of the scene. This flat video is then warped on the fly to be projected. The best advantage of a spherical projection is that when a video is playing on a half dome, the user can turn the camera to look around in the video. Although the user is still constrained to the path that the camera takes originally, this method increases the interactivity of visualisations allowing a user to explore the data. Spherical projections can be created using similar methods as the fisheye projections by rendering cube maps and converting them in Drishti or by rendering spherical projections directly with POVray.

Results

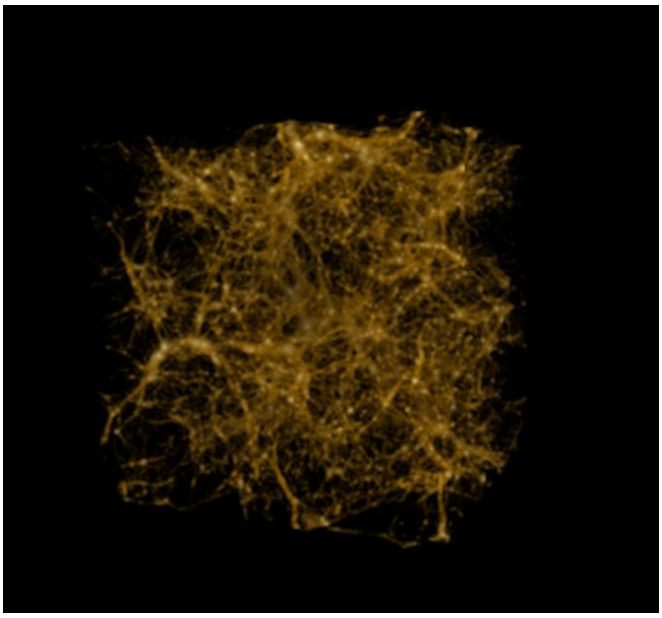

Figure 8.

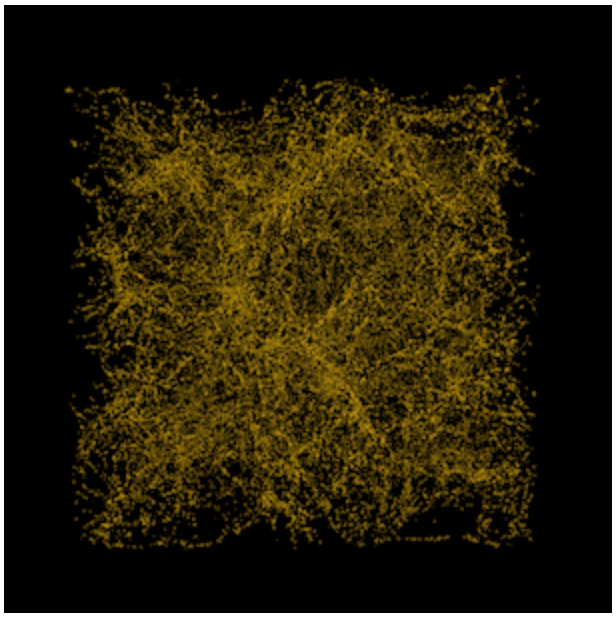

Figure 9.

Figure 8 and Figure 9 are both calculated from the OWLS cosmological simulation data sets. This data set contains about 15 million points and is approximately 300 million lights years in each dimension. The calculation time for these two volumes is about 1 hour and 40 minutes. Figure 8 shows the total gas density. The bright dots are galaxies and the filaments that are visible is the cosmic web. Figure 9 shows the HI gas density for the OWLS data set. The image shows that the gas is clumped together, this is because the gas between the galaxies is ionised.

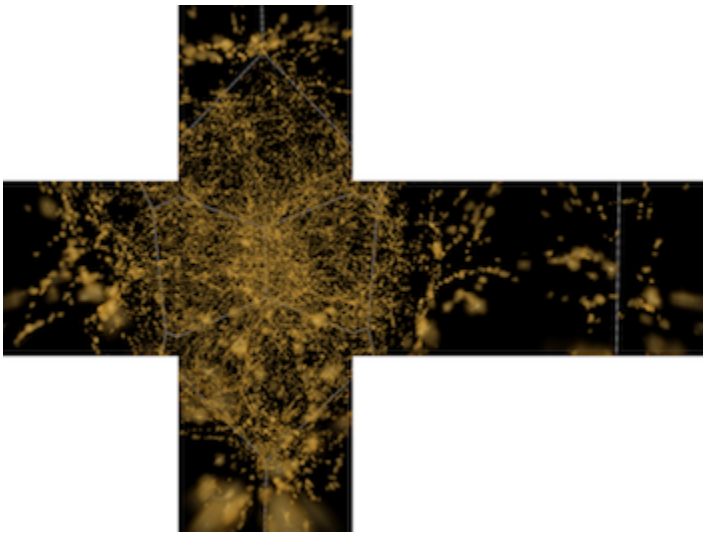

Figure 10.

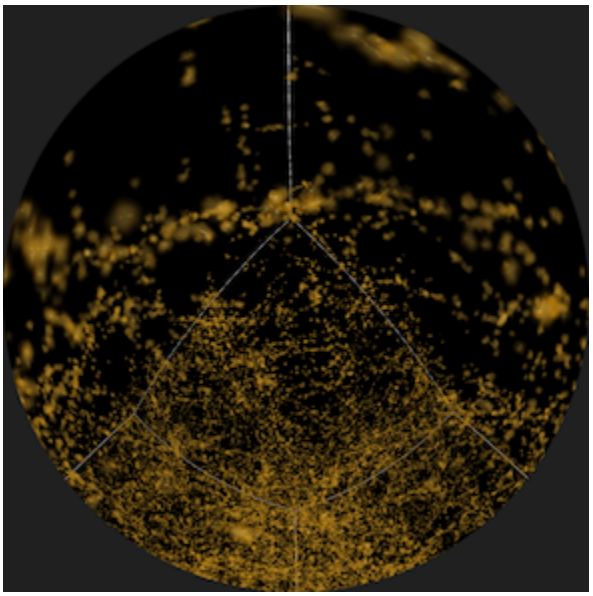

Figure 11.

Figure 10 is a cube map that has been generated by Drishti. A slight draw back of cube maps is the large file sizes which can be a constraint if long videos are being created from cube maps. Figure 11 is a fisheye projection created from the cube map. This shows the total gas density of the OWLS cosmological simulations.

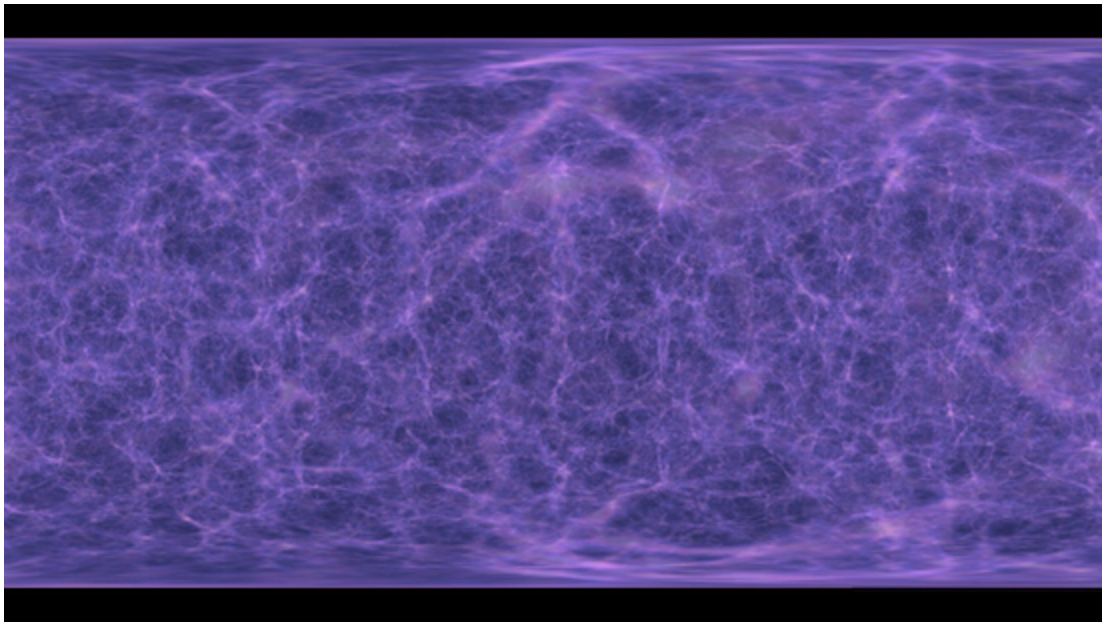

Figure 12.

Figure 12 is taken from a video that was produced using POVray. Using POVray allows for direct exportation of spherical projections without first having to render a cube map. This image also shows the total gas density of the OWLS cosmological simulations. The flat video is converted on the fly to a fisheye projection and using custom video player, it is possible to look in any direction whilst the movie is playing as well as when it is paused.

Figure 13.

Figure 13 shows the spherical projection from above being warped and tested on the half dome at WASP located at the University of Western Australia.

Further work

Further work on this project includes making much higher resolution images and video. Time limitations and hard disk space prevented me from achieving this in the project. A method that would significantly decrease computational time is producing Drishti pvl.nc file directly from my program. This would eliminate the need to import the raw files using the importation tool. Importing volumes took the most amount of time compared to the other steps of the project. Another clear benefit of producing the pvl.nc files directly from my code is smaller file sizes, and the ability to calculate the 2D Occlusion Spectrum files giving me the ability to simplify my processing pipeline. Calculating smoothing files in parallel would decrease calculation time by a factor of 20. My code could be extended to have more support for rendering in POVray as one of the benefits is parallel processing. Interest has been shown from a planetarium to produce a high resolution short presentation to be played at the Horizon planetarium.

Conclusions

I was successfully able to create code that could easily run on different processors in a High Performance Computing environment allowing me to create a series of high resolution videos and images. I created an easily extendable visualisation pipeline that will be used for similar projects in the future. I also explored different visualisation techniques and methods that can be used to extend the audience and interactivity of viewing datasets using Drishti and POVray to render volumes.

References

[1] J. J. Monaghan and J. C. Lattanzio, "A refined particle method for astrophysical problems.", Astronomy and Astrophysics, vol. 149, pp. 135-143, August 1985

[2] A. R. Duffy, "Investigation of large scale structure in the Universe", PhD thesis, University of Manchester, Manchester, United Kingdom, 2009.

[3] A. Limaye, ".pvl.nc file format", Drishti, 22/11/2009, [Online]. Available: http://code.google.com/p/drishti-2/wiki/PvlDotNcFormat [Accessed: 23/02/2010]

[4] P. Bourke, "Geometric Distortion For Dome (Fisheye) Projection", Western Australian Supercomputing Program, November 2004, [Online]. Available: http://local.wasp.uwa.edu.au/~pbourke/miscellaneous/domefisheye/domegeom/ [Accessed: 23/02/2010]

Acknowledgements

- Alan Duffy - International Centre for Radio Astronomy Research
- Paul Bourke - Western Australian Supercomputing Program
- Ajay Limaye - Creator of Drishti