School of Physics and Astronomy

Senior Honours Project

Visualising the Cosmic Web and Dark Matter Halos

Adair Wood 2021

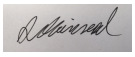
Abstract

The aim of this project was to create 3D movies of cosmic structure forming. This was done with data from the Legacy Project simulations, N-Body dark matter simulations created by the TMoX group at the Institute for Astronomy. Movies were made of a 100Mpc/h simulation box containing 2048^3 particles as well as a movie of the evolution of an individual dark matter halo in the simulation. The visualisations were made with Python using open source `yt` volume rendering functions and the `ytree` package [1] . The animations show the evolution of structure both forwards and backwards in time from different angles, as well as the mass of objects on a log colour scale. Giving the observer a visual understanding of the structures formed by dark matter. Links to the movies and codes with documentation can be found in appendices A and B.

Declaration

I declare that this project and report is my own work.

Signatures: AW

Date: 5/3/2021 

Supervisors: Dr. S. Kochfar, Dr B. Smith

Contents

1 Introduction

The concepts of dark matter and the cosmic web are among the most significant developments in cosmology in the past century. Computer simulations of dark matter are essential to modern cosmology, due the vast size of the systems being modeled and our limited to capacity to experiment with it.

"Now, normally, when scientists want to get to grips with how something works, rather like a small child armed with a screwdriver and a toy, we try and take it to pieces. But the problem with dark matter is that, although we can see its effects, we can't actually detect it or manipulate it. So how can we understand it? One way is to work out what it must be doing by using computer simulations. You tell the computer what dark matter does in some particular way and then ask it to predict how the Universe would look if that were true."[2]

This area is being investigated by the TMoX Group's Legacy Project, led by Dr. Sadegh Kochfar. The Legacy project is a suite of cosmological N-body simulations made with Gadget-4 code, an N-body Smoothed Particle Hydrodynamics code [3].

In order to fully utilise the results of these simulations, the data needs to be visualised. Visualisation enables researchers to 'see' dark matter that otherwise we would not be able to observe and presents the results in an accessible and easily analysed format. Seeing how dark matter is distributed contributes towards understanding its nature and how it fits within our current knowledge of physics.

The simulations consist of two primary volumes of 1600 Mpc/h and 100 Mpc/h boxes run from $z = 99$ to $z = 0$ with up to 2048^3 resolution elements, as well as 83 Mpc/h zoom-in simulations on the larger box with an effective particle number resolution of 32768^3.

Name	L_{box} (Mpc/h)	L_{box}^{hr} (Mpc/h)	N^{hr}	$N_{eff.}$	m_p (M_o)	Comment
	box size	box size high resolution	high-resolution particle number	effective high-resolution particle number	particle mass	
1600	1600.0	1600.0	2048^3	2048^3	5.43E10	full-box
the expanse	1600.0	703.1	1800^3	4096^3	6.78E+9	zoom-in box
mean	1600.0	83.0	1700^3	32768^3	1.32E+7	zoom-in box
m1s	1600.0	83.0	1700^3	32768^3	1.32E+7	zoom-in box
m2s	1600.0	83.0	1700^3	32768^3	1.32E+7	zoom-in box
p1s	1600.0	83.0	1700^3	32768^3	1.32E+7	zoom-in box
p2s	1600.0	83.0	1700^3	32768^3	1.32E+7	zoom-in box
cluster	1600.0	83.0	1700^3	32768^3	1.32E+7	zoom-in box
void	1600.0	83.0	1700^3	32768^3	1.32E+7	zoom-in box
100	100.0	100.0	2048^3	2048^3	1.32E+7	full-box, same resolution as zoom-in box
10	10.0	10.0	1024^3	1024^3	1.06E+5	full-box, very high time resolution halo catalogs at high-z

Figure 1: Table of high resolution simulation volumes created by the Legacy Project

This project worked with the high resolution 100Mpc/h box. These simulations produce multiple terabytes of data and making detailed informative movies with this volume took roughly 4-5 days to compile even with $>$ 100 GB of RAM on the Cuillin computing cluster. So although rendering the 1600Mpc/h would provide more informative images, it would have added a considerable time constraint.

2 Method

2.1 Making the Fames

The `yt` library was used for this project as it offers several methods for easily visualising astrophysical simulation data, such as `yt.SlicePlot()`, `yt.ParticleProjectionPlot()` & `yt.ParticlePlot()`. The `yt.ParticleProjectionPlot()` function was initially considered as it produced it produces informative images (figures 2&3) with very few lines of code.

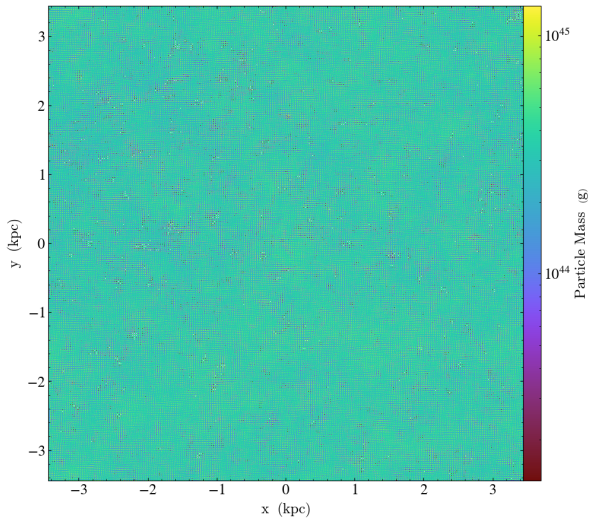


Figure 2: Particle Projection plots of low res. $N = 256^3$ box. $Z = 99$ snapshot

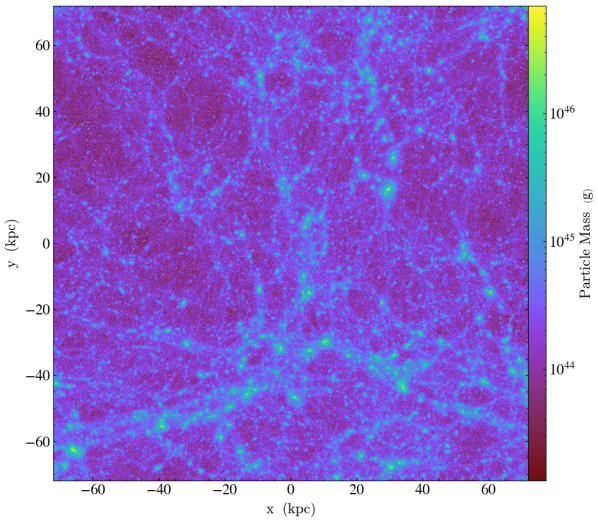


Figure 3: Particle Projection plots of low res. $N = 256^3$ box. $Z = 0$ snapshot

However this approach presented limitations. The aim was to make a movie that showed the observer the structures formed in the 3 dimensional simulation volume, and in order to do this most effectively The viewer should be presented with views from multiple different angles. Moving the point of view to create a dynamic animation using `yt.ParticleProjectionPlot()` proved cumbersome. The solution to this was to use the `yt` volume rendering camera [4]. The virtual camera can be moved, focused and oriented in the scene rendered giving the user an intuitive control over what images are produced. The volume rendering package is also able to automatically calculate a colour transfer function for the field being visualised, which means the field boundaries would not have to be defined by the user every time a new data set is loaded. This method also allows the user to pick different levels of contrast. The contrast is set by a user defined value of σ as the volume rendering uses sigma clipping to determine the contrast. Images were made with $\sigma = 2$ as this showed the filament structure in the best detail.

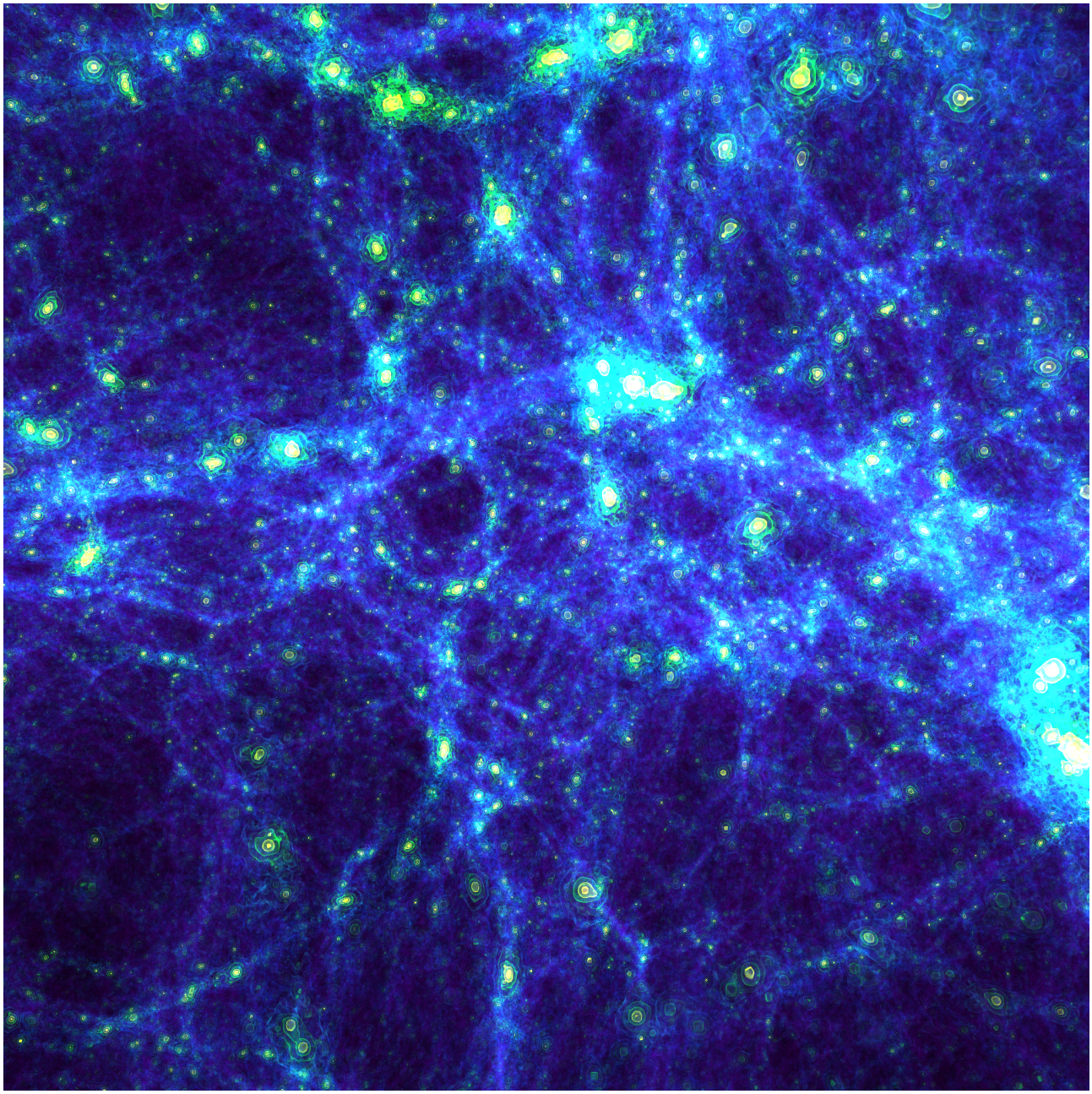


Figure 4: Frame produced using volume rendering camera.

It should be noted that for the volume rendering method to work the snapshot files first needed to put into `YTCoveringGrid`. e.g.

```
ds = yt.load("snapshot.hdf5")
snapshot_covering_grid = ds.covering_grid(level, left_edge, dims,
fields=None, ds=None, num_ghost_zones=0, use_pbar=True,
field_parameters=None)
```

[5]

The covering grid is a data container that covers the simulation volume in a user specified number of grid cells. It smooths each particle in the simulation on to the nearest grid cells (a 2x2x2 cube) via a cloud-in-cell method. This was done by Dr B. Smith, taking the legacy `snapshot_*_.hdf5` files and creating grid covered `snapshot_*_covering_grid.h5` files with 1024^3 grid cells.

Images annotated with the particle mass-colour plot and the simulation time could also be made (figure 5). However the inclusion of colour plot and time stamp take up a considerable portion of the images, and as the movie is more focused on a giving qualitative understanding of the system un-annotated images were produced (e.g. figure 4).

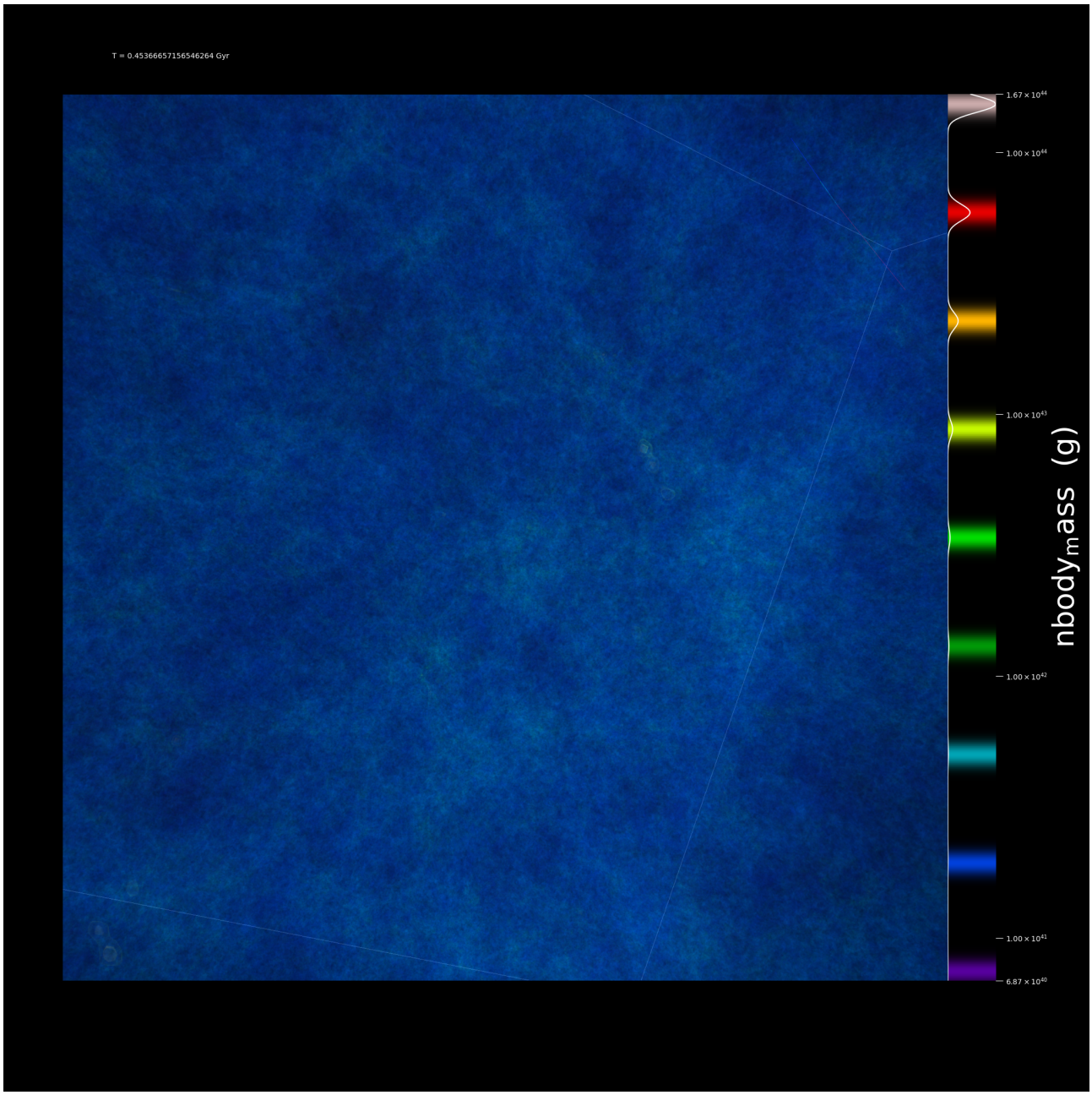


Figure 5: Annotated volume rendering image.

2.2 Full Volume Movie

Movies of similar cosmological simulations often rotate as the simulation evolves [6]. This was achieved using `for _ in cam.iter_rotate(angle, steps, cam.north_vector, cam.focus)` [7] where the focus point was the center of the box. This function moves the camera like so (figure 6):

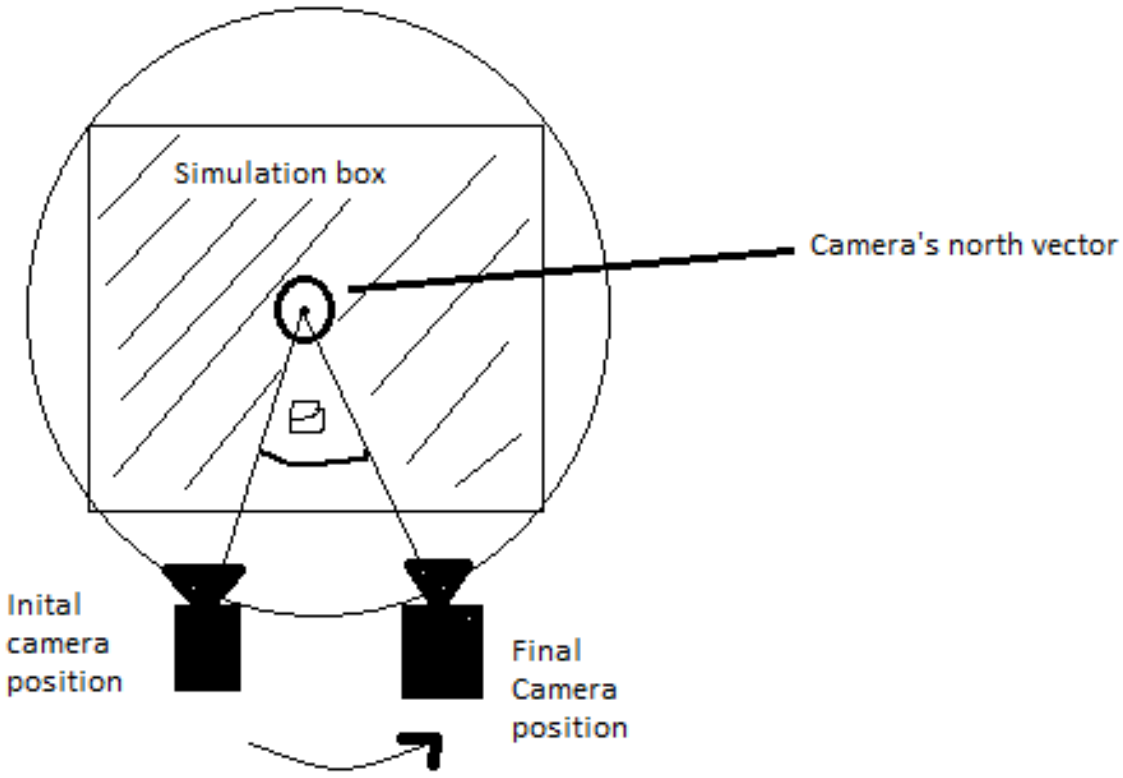


Figure 6: Camera rotation through an angle in one step.

With this function the main code to make the movies broadly followed the steps laid out in figure 7.

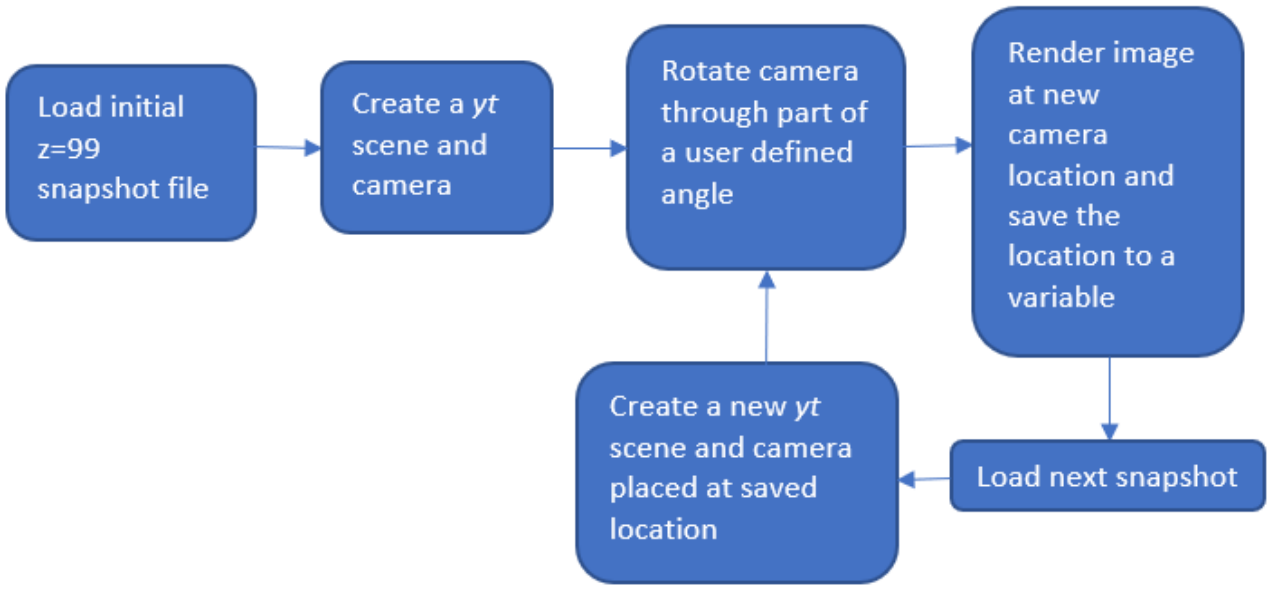


Figure 7: Basic algorithm describing making a moving and evolving animation.

The movie has sections where it shows the simulation evolving forwards and backwards in

time. and sections where where the volume rotates at $z = 0$ so the viewer can see the final structure from different perspectives. This scheme is illustrated in figure 8

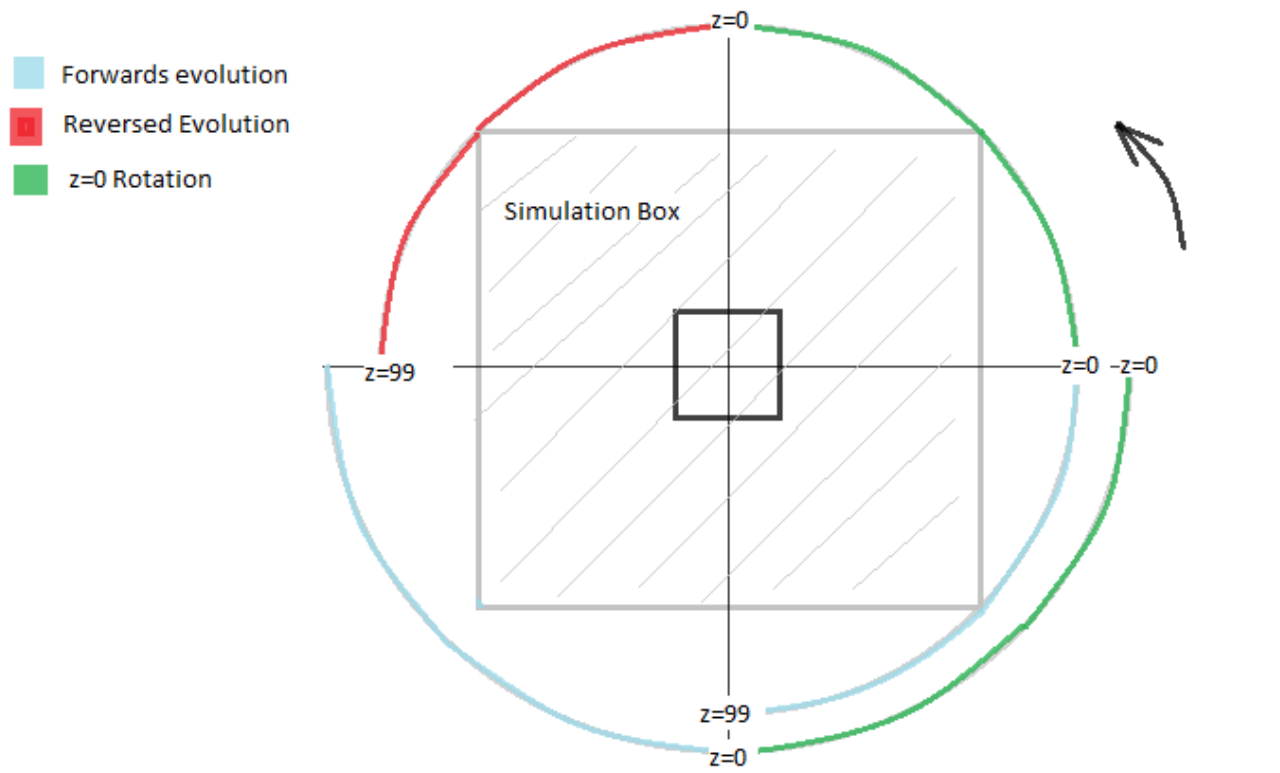


Figure 8: Path the camera describes in the full volume movie, with colours corresponding to the time evolution.

The `frame_*.png` files were then compiled into a movie using linux terminal command `ffmpeg`.

2.3 Dark Matter Halo Movie

Creating the dark matter halo evolution frames involved many more steps than the full simulation movie. For the latter, the camera was simply set to focus on the center of the simulation and then rotated around the volume as it evolved, with the frame width of the camera being the box width. The halo movie however, required filming a smaller object that moves and changes size over time. To achieve this, not only did a halo have to be chosen but information about it's evolution had to be accessed as well. This was done with data loaded from `rockstar mergertrees` using `ytree` [1]. `ytree` allows the user to access the history of halos in the simulation. Halo "progenitors" refer to the halo at earlier times, with each merger creating another progenitor.

```
#load merger tree data
a = ytree.load("../mergertree_h5/rockstar_groups/rockstar_groups.h5")
k = a['mass'].argmax() #select largest halo to focus on
#relevant values of progenitor halos
prog_pos = mytree['prog', 'position']
prog_rad = mytree['prog', 'virial_radius']
prog_time = mytree['prog','time']
```

This progenitor data is used to adress the problems discussed sections 2.3.1 and 2.3.2.

The first question to address was which halo to film. The halo with the largest mass at $z = 0$ was chosen for this as it would show a large amount halo mergers, giving the viewer a more dynamic picture of halo formation.

2.3.1 Focus Point

The next problem is that the position of the halo changes over time, so the camera's focus point would need to track this. Simply pointing the camera at the new progenitor position in each new snapshot would not be sufficient, as this would create a very "shaky" movie, due to the discontinuous nature of the position change between snapshots and the fact that the snapshot times do not exactly match progenitor times. This was solved by taking the original halo coordinates and creating a smoothed cubic position function using:
`scipy.interpolate.UnivariateSpline(time, x, k=3)` [8]. This process is illustrated for the x-component of the halo in figure 9.

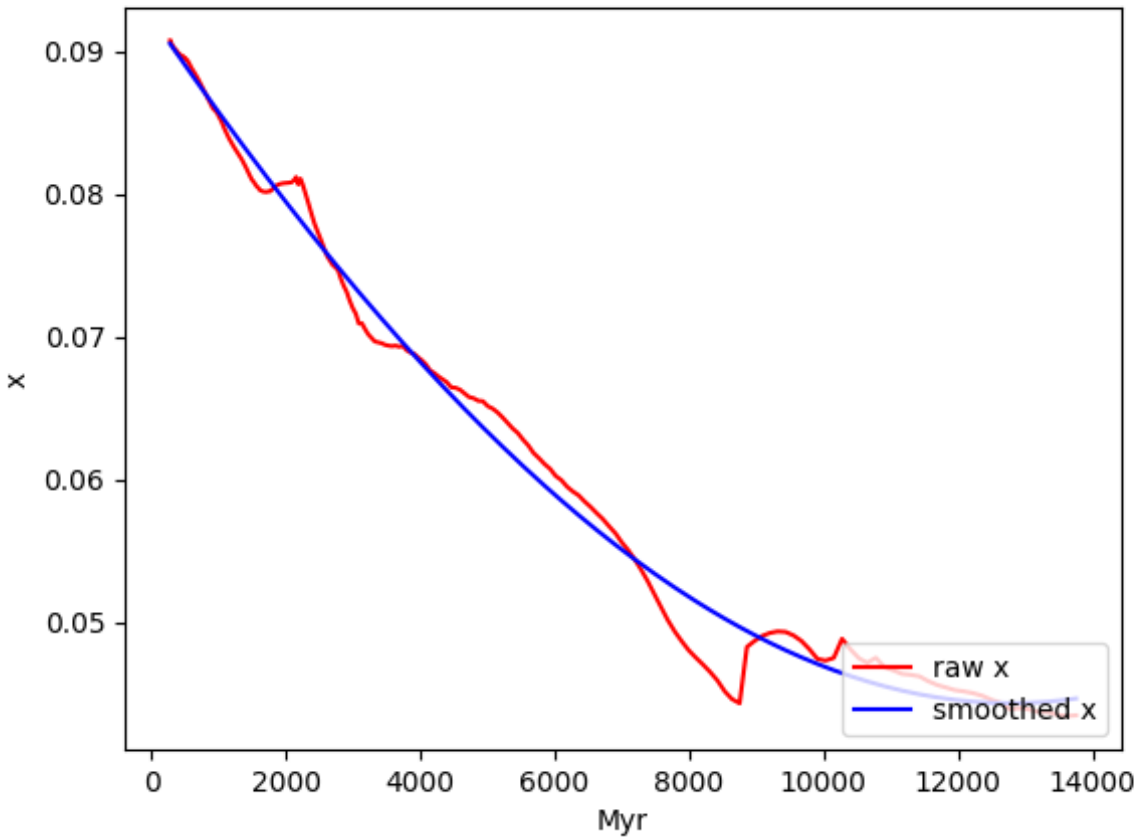


Figure 9: X component of the halo position from `ytree` and subsequent smoothed position function

2.3.2 Zoom factor

Focusing the camera on the desired halo while having a frame width as large as the domain for example, makes it very hard for the viewer to discern the object and any details about it's evolution (see figure 10).

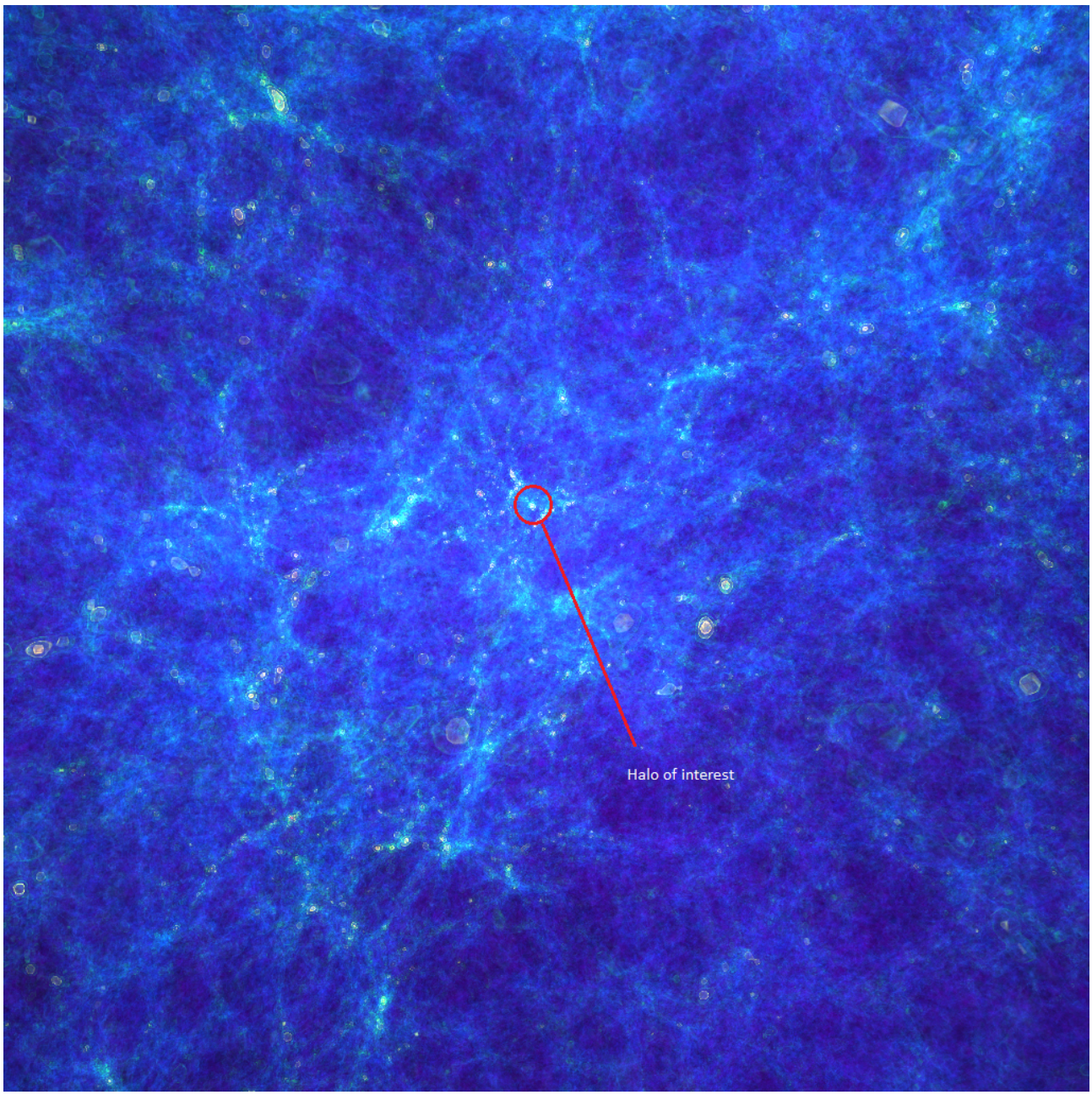


Figure 10: Full width rendering focused on the largest mass halo roughly halfway through it's evolution.

Therefore the frame width needs to be set to a much smaller value to accurately observe the halo. As the halo's radius increases over time as it merges with other halos, setting a constant frame width risks having images where the halo again appears too small or potentially too large and the viewer would not be able to observe the merger process. Therefore the frame width has to change along with the halo size. So in each snapshot the frame width was determined by the Virial radius of the halo. Similar to the problem in 2.3.1 the width does not vary smoothly

and sudden jumps in the frame width make for a jarring viewing experience. The solution was to again create a smoothed radius function from the `mergertree` data using the same method as 2.3.1.

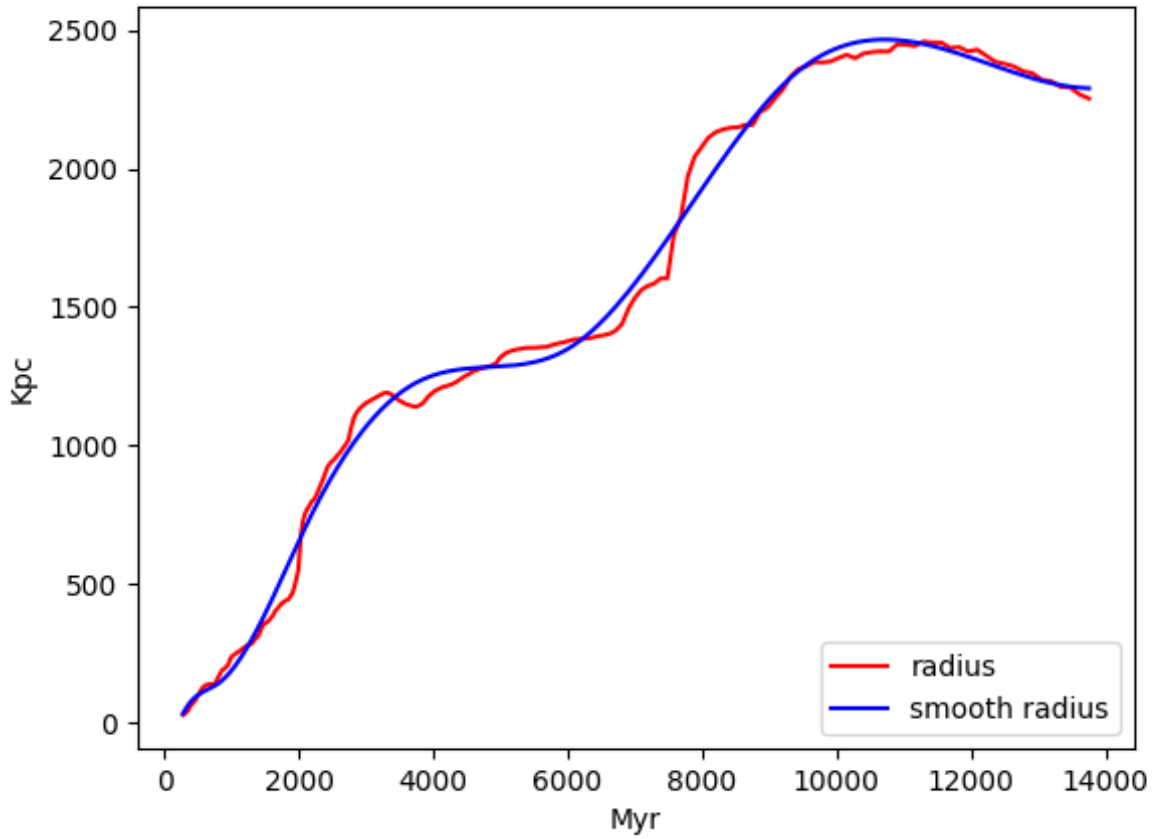


Figure 11: Comparison of the halo radius and the smoothed radius function.

Looking a figure 12 it can been that in several regions the halo radius is greater than the smoothed radius function. This would produce images where the halo fills the field of view. To address this the the frame width is set to be $F \times SmoothedRadius$ where F is some user defined factor > 1.

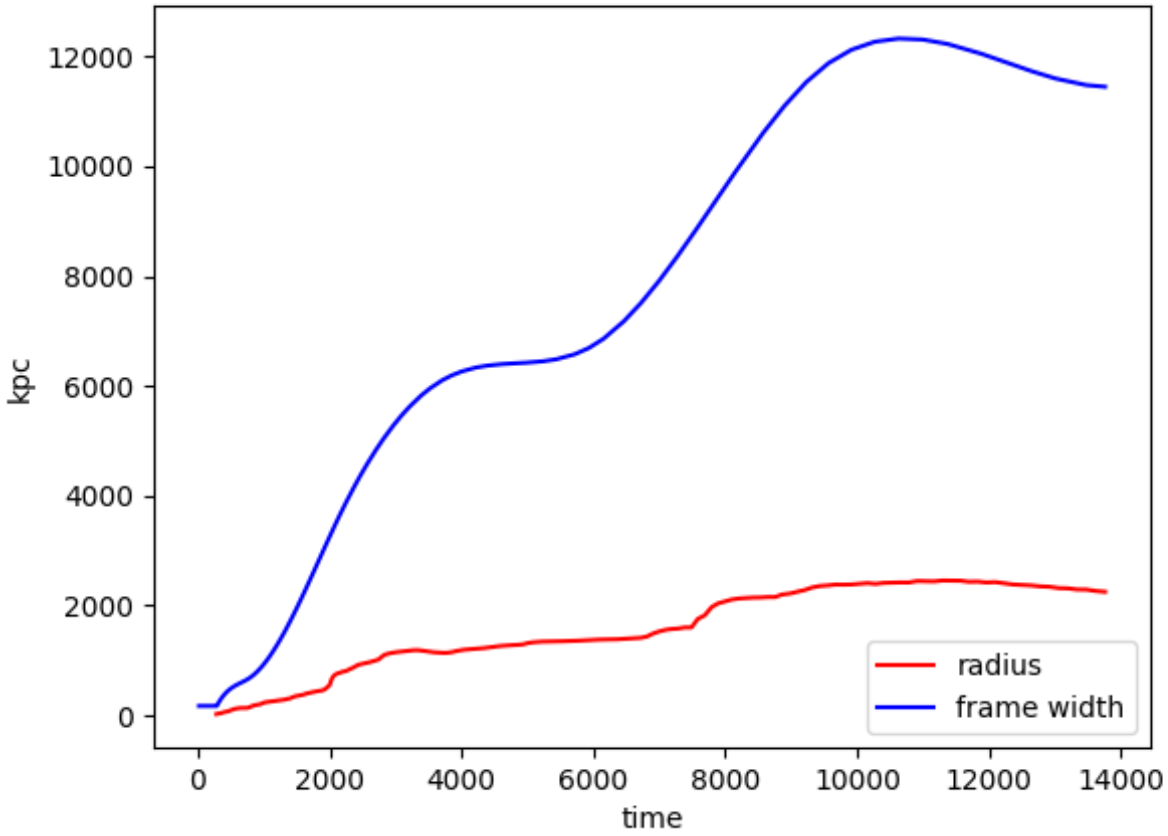


Figure 12: Comparison of the halo radius and the frame width where $frame\ width(t) = 5 \times smoothed\ radius(t)$. time is in Myr.

For the published movie a frame width of $100 \times Smoothed\ Radius$ was chosen to see smaller halos come from the edge of the frame to merge with the central halo.

The frame width also always needs to be greater than the difference between the halos smoothed position function and it's actual position to avoid having frames where the halo is not present. The problem is also solved by this approach as the Virial radius is always greater than the smoothed position residual (figure 13).

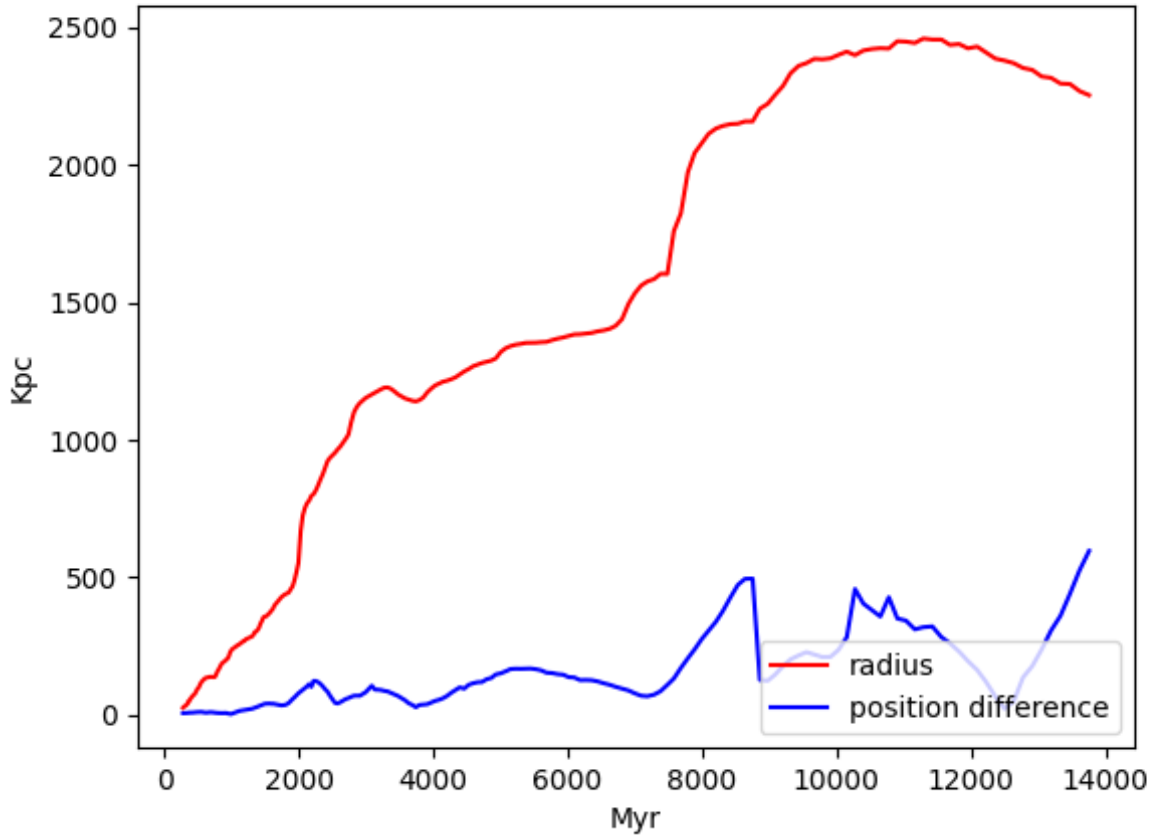


Figure 13: Plot comparing the Virial radius and the magnitude of the smoothed position function residuals

2.3.3 Optical Depth

If, for the sake of argument, the camera was placed in the domain center and focused on on a halo close to the domain edge along with the appropriate frame width. There could still be a great deal of foreground objects obscuring the the details of the halo. To avoid this, in each snapshot the halo was placed in a `YTSphere` [9], centered on the halo position/focus point with a radius of 100 frame widths.

```
ds = yt.load("snapshot_*_covering_grid.h5")
sp = ds.sphere(focus_point, 100*frame_widths)
```

`YTSphere` is a data container that creates a sphere of points defined by user specified center and radius. The camera is then placed at the edge of this region to render images. As the volume of this sphere (order kpc^3) is smaller than the simulation volume (order Mpc^3) this removes unwanted regions from the field of view. It also reduces the amount of grid cells that need to be rendered, decreasing the code's run time.

2.3.4 Initial Conditions

Another issue came from the fact that halos don't exist at the start of the simulation, they need time to form. Therefore there is no progenitor halo information to give the camera for

it's frame width and focus point in the early snapshots. To solve this, in snapshots prior to the formation of the first progenitor ($t < t_0$), the camera's frame width's and focus points are set to $frame\ width(t) = frame\ width(t_0)$ and $focus\ point(t) = focus\ point(t_0)$.

As mentioned in 2.3.3 the camera is placed at the edge of a spherical region. This process is illustrated in figures 14 and 3

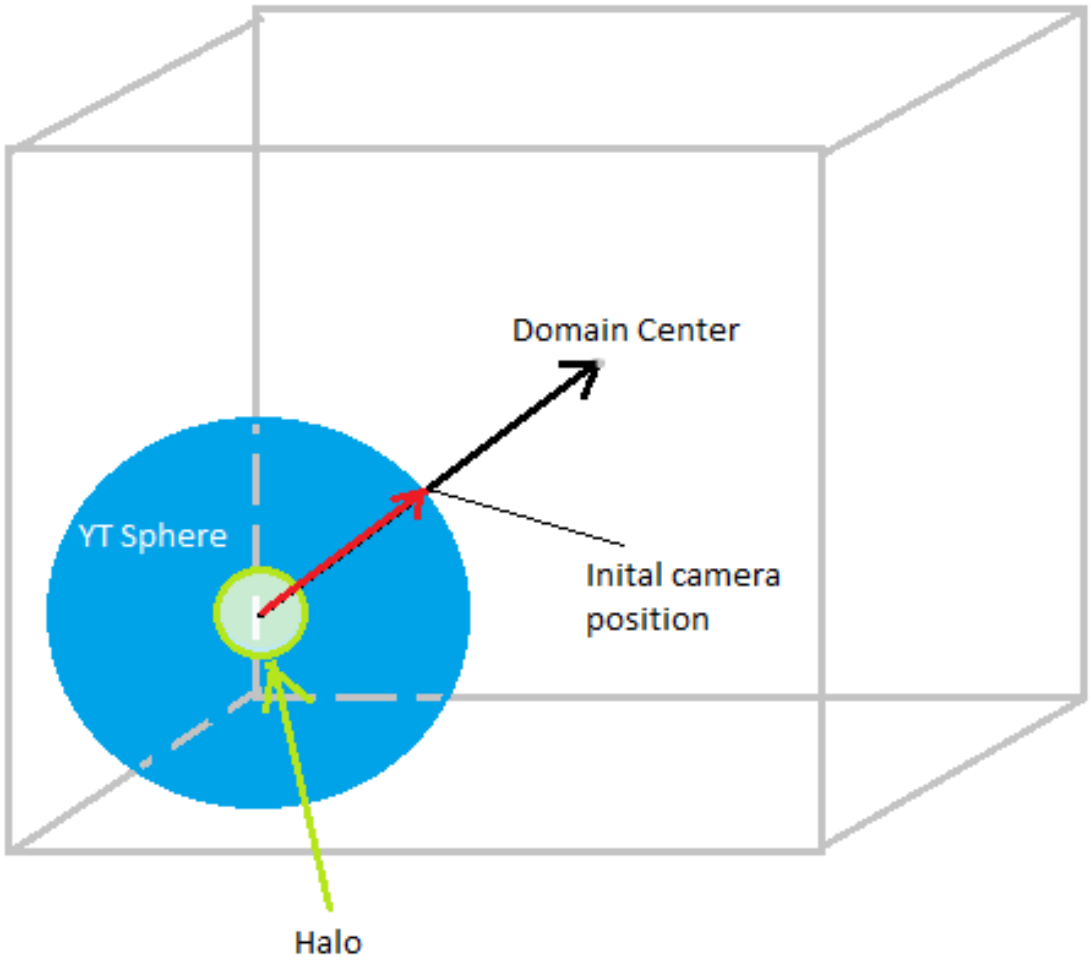


Figure 14: Illustration showing how the camera was initially positioned

S is sphere center position vector, **C** is domain center position vector and r is the radius of the sphere.

$$\mathbf{v} = \mathbf{S} - \mathbf{C} \tag{1}$$

$$\mathbf{r} = r\hat{\mathbf{v}} \tag{2}$$

$$initial\ position = \mathbf{S} - \mathbf{r} \tag{3}$$

This method for setting the initial position was chosen because it keeps the camera away from the domain edge. The necessity of this is apparent in figure 15. Because `yt` volume rendering does not obey periodicity, if the edge of the camera's frame goes beyond the boundaries of the box the subsequent images have empty black regions in them.

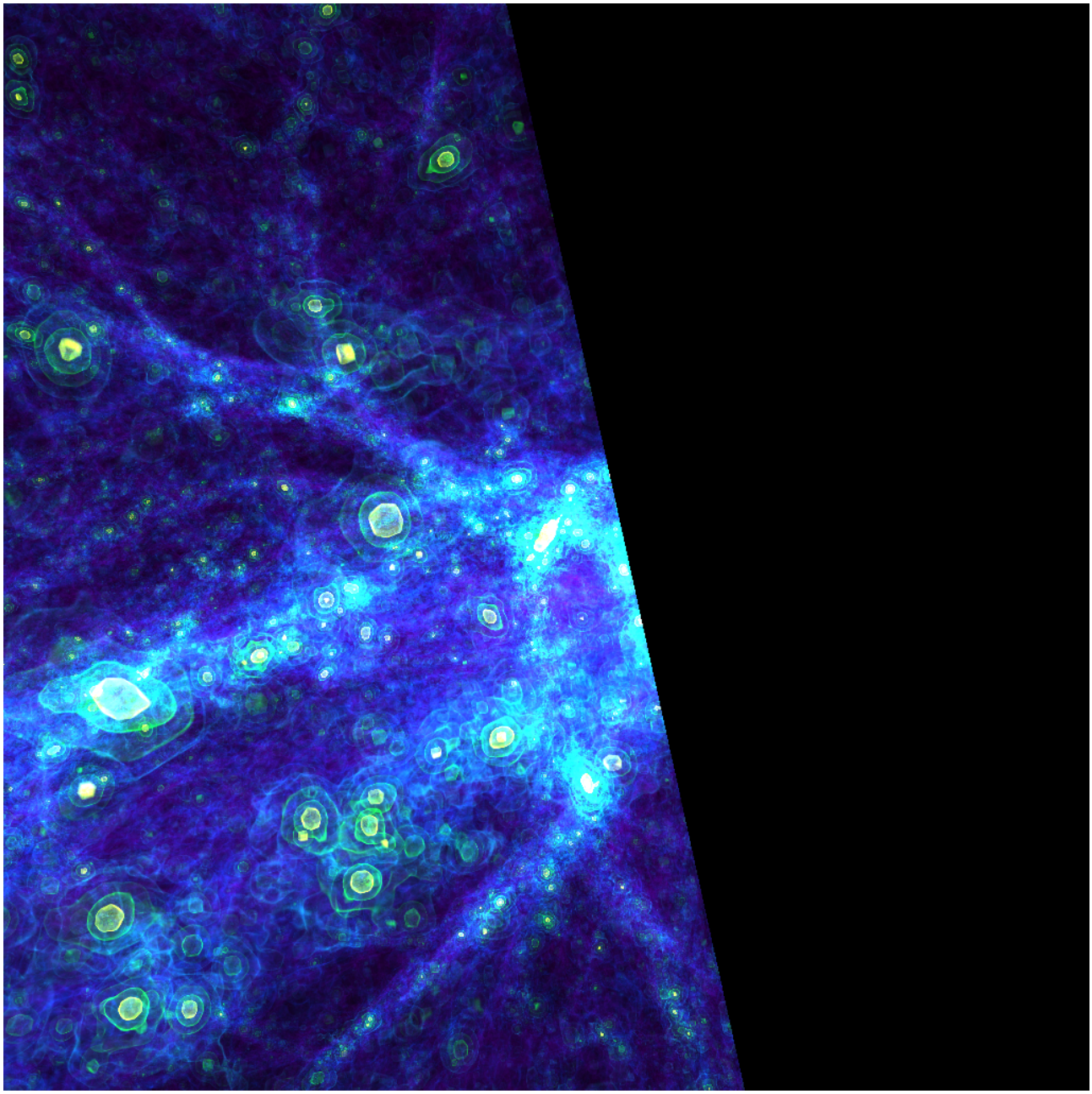


Figure 15: Image of halo produced with camera overlapping the domain edge

2.4 Filming the Halo

With the problems of smoothly adapting the camera's zoom and focus point and the initial camera settings solved, the halo evolution frames can then be produced. The process for this is essentially the same as in figures 7 and 8 (albeit with the addition of new frame widths and focus points and spheres in each snapshot). Also, in figure 8 the camera was rotated through $\frac{\pi}{2}$ radians in each section, this angle was set to a much smaller $\frac{\pi}{12}$ for the halo evolution. This was because the halo is in a corner of the box (approximately (0.1,0.1,0.9) in `unitary` units) very close to edges, keeping the rotation angle small avoided any chance of the frame going over the domain boundary.

3 Discussion

The movies display similar structure to what's seen in other dark matter simulations (figures 16 and 17) such as web like filaments and large voids.

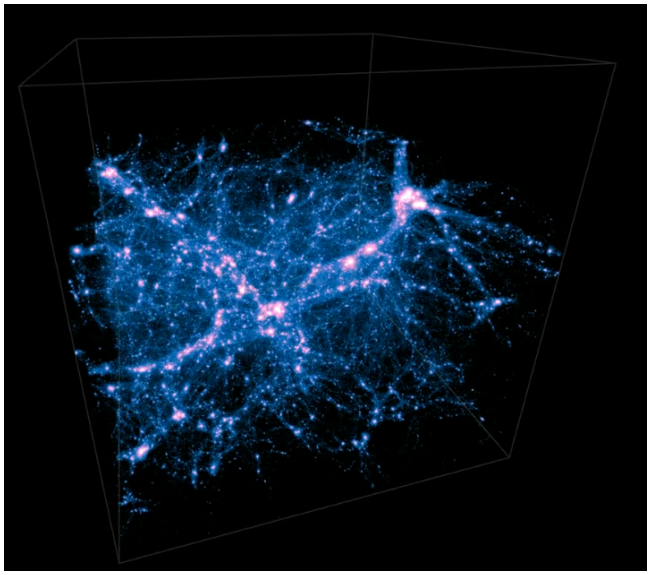


Figure 16: Illustris Project[6].

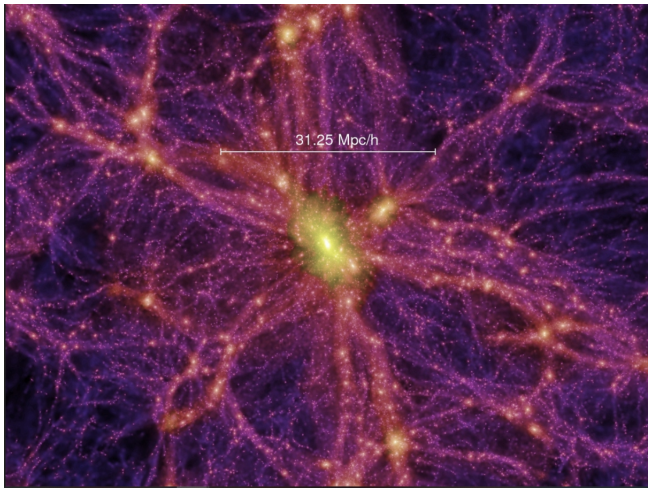


Figure 17: Millennium Project[10].

This means Tmox group members can use these visualisation methods to quickly inspect the results of other Legacy Project simulations and check for any abnormal or interesting aspects. In addition, aside from being pleasant to watch, being able to see the structure dynamically evolve gives the viewer an natural intuition about how the cosmic web forms.

The halo evolution movie shows some interesting features, such as progenitor halos orbiting or passing through each other and in-spiraling before fully merging. Given that spiral galaxies are embedded in dark matter halos [11] and likely have super massive black holes in their center [12], combined with the recent detection of gravitational waves from black hole mergers [13], one can imagine a future scenario where gravitational waves can be used to probe the formation of the cosmic web.

Although this project made movies of the 100Mpc/h box with 2048^3 resolution elements, the codes were written to be suitable for use on other Legacy data sets. So following on from this project, similar images and movies could be made of the 1600 Mpc/h box, these would be very interesting as they would display substantially larger webs and would be better for comparing with the real observations of cosmic structure.

References

[1] B. Smith and M. Lang, "ytree: A python package for analyzing merger trees," Dec. 2019.

[2] C. Smith and D. C. DeGraf, "How do you model dark matter? what can we learn about dark matter from cosmological simulations? colin degraf is here to tell us. interview with colin degraf, cambridge university," *https://www.thenakedscientists.com/articles/interviews/how-do-you-model-dark-matter*, 2019.

[3] V. Springel, "The cosmological simulation codegadget-2," *Mon. Not. R. Astron. Soc*, vol. 364, no. 1, 2005.

[4] YT Project, "Yt volume rendering camera," *https://yt-project.org/docs/dev/visualizing/volume_rendering.html?highlight=camera*.

[5] YT Project, "Ytcoveringgrid documentation," *https://yt-project.org/docs/dev/reference/api/yt.data_objects.construction_data_containers.html#yt.data_objects.construction_data_containers.YTCoveringGrid*.

[6] The Illustris Project, "Time evolution of a 10mpc (comoving) cubic region within illustris," *https://www.illustris-project.org/movies/illustris_movie_cube_sub_frame.mp4*, 2018.

[7] YT Project, "Yt moving camera documentation," *https://yt-project.org/docs/dev/cookbook/complex_plots.html#cookbook-camera-movement*.

[8] SciPy.org, "Scipy spline documentation," *https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.UnivariateSpline.html#scipy.interpolate.UnivariateSpline*.

[9] YT Project, "Yt sphere," *https://yt-project.org/docs/dev/reference/api/yt.data_objects.selection_objects.spheroids.html?highlight=ytsphere#yt.data_objects.selection_objects.spheroids.YTSphere*.

[10] The Millennium Simulation, "The millennium simulation project," *https://wwwmpa.mpa-garching.mpg.de/galform/virgo/millennium/#slices*, 2005.

[11] R. H. Wechsler and J. L. Tinker, "The connection between galaxies and their dark matter halos," *Annual Review of Astronomy and Astrophysics*, vol. 56, no. 1, pp. 435–487, 2018.

[12] L. Ferrarese and D. Merritt, "A fundamental relation between supermassive black holes and their host galaxies," *The Astrophysical Journal*, vol. 539, p. L9–L12, Aug 2000.

[13] B. Abbott, R. Abbott, T. Abbott, M. Abernathy, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R. Adhikari, and et al., "Observation of gravitational waves from a binary black hole merger," *Physical Review Letters*, vol. 116, Feb 2016.

A 1

Both movies can be viewed via this link `https://vimeo.com/showcase/8324326`

B 1

Github repository for codes and documentation `https://github.com/sandonandfreinds/SHP`