Generation & Visualisation of Cosmological Data

Kapil Gupta
University of California, Santa Cruz
Santa Cruz, CA, USA

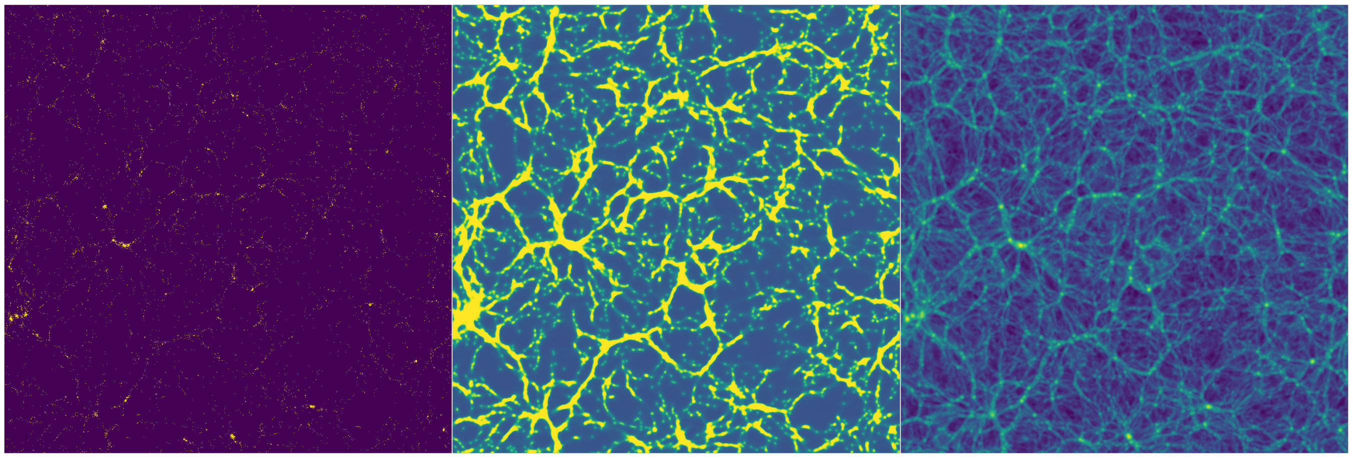

Figure 1: Cosmological data generated using the 2D HDR GAN. The leftmost image here is the Halo Map, that consists of bright spots called the halos and constant color everywhere else, signifying lack of halos. The rightmost image is a slice of the 3D data cube. The middle image is generated using 2D HDR GAN.

ABSTRACT

Cosmological N-body simulations are computationally expensive. Deep Learning techniques can be used in order to reduce the time and resources required, while providing the same statistical properties as the data generated by Monte Carlo Simulations. Understanding the differences between data that has been generated using physical properties such as laws of gravity, properties of dark energy, cosmological constants and the data generated using a Generative Adversarial Network (GAN) can help understand the usefulness of the dataset. We explore the useage of a Generative Adversarial Network for generating the Cosmological data. We further use progressive Direct Volume Rendering(DVR) that accumulates object-space samples over successfully rendered frames.

CCS CONCEPTS

• **Computer Graphics** → **Volume Rendering**; • **Deep Learning**;

KEYWORDS

volume rendering, neural networks, generative adversarial network

ACM Reference Format:
Kapil Gupta. 2018. Generation & Visualisation of Cosmological Data. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Cosmic Web is an intricate multiscale interconnected network composed of filaments of clustered galaxies and other matter such as gases. It is a large-scale structure that is found all over the observable universe and provides great insights into how the dynamics of gravitational structure formation. Topological and morphological information provided from such data allows astrophysicists to study dark matter, cosmological model, and formation and evolution of galaxies. The N-body simulations such as EAGLE and Bolshoi-Planck are generally accepted method for generating such Cosmological data. But the Monte-Carlo simulations required to generate required data are quite expensive computationally. Using Deep Learning algorithms to generate such data with same statistical pattern guarantees can help researchers work more efficiently.

Recently, GANs have been proposed for emulating the matter distributions in two dimensions. These approaches have been successful in generating data of high visual quality, and almost indistinguishable from the real simulations to experts. Moreover, several summary statistics often used in cosmology, such as power spectra and density histograms, also revealed good levels of performance. Some challenges still remain when comparing sets of generated samples. In both works, the properties of sets of generated images

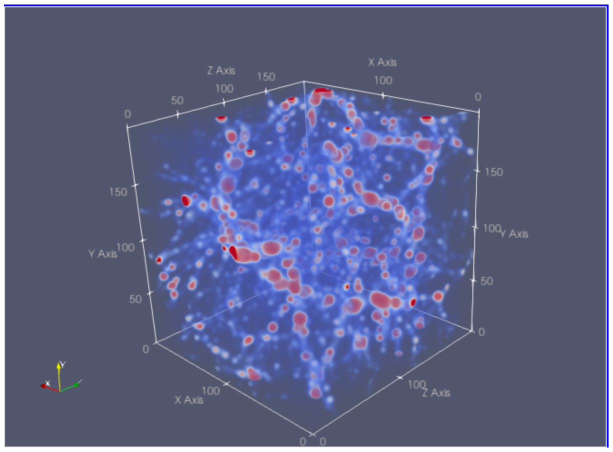

Figure 2: Volume Rendering of the data cube, as visualized using Paraview software.

did not match exactly; the covariance matrix of power spectra of the generated maps differed by order of 10% with the real maps.

While such results signify that GANs are a good approach for generating cosmological data, there is still a significant difficulty in generating high quality data in both 2D and 3D. We explore the idea of using GANs to generate hight quality intensity maps by taking in high mass/density halo positions as input.

2 RELATED WORK

Generative models that can produce novel representative samples from high-dimensional data distributions have become quite popular in different fields such as image translation or image in-painting. Popular Generative models include Variational Autoencoders, Auto regressive models such as Pixel RCNN and Generative Adversarial Models. The most common approaches for generating cosmological data include upsampling using GAN [5], super-resolution of low-resolution simulation data [6] and inpainting, cosmo3.

Direct Volume Rendering has two main approaches that are used widely: Splatting and Ray Casting. In the splatting technique, the data is resampled to a rectilinear grid and then each voxel is drawn from back to front, based on the camera orientation, using a small polygon with a gaussian texture whose color and opacity are based on the voxel's data and the volume transfer function. [8] discuss the application of splatting to neurovascular data. In the raycasting technique, there are a lot of optimizations that can be performed in both image and object space to allow for interactive rendering on unstructured data. Most common methods include reducing sampling rate during interaction, progressive refinement by recomputing required samples or by image space optimization. [7] provide a new method of progressive DVR that is interactive and less computationally intensive as it reuses the previous samples, although this method is memory intensive and requires more resources than other possible methods. [2] is a great framework that utilizes DVR to provide an interactive system.

In order to perform comparison between different 3D volume datasets, geometric approaches can be used as suggested by [4].

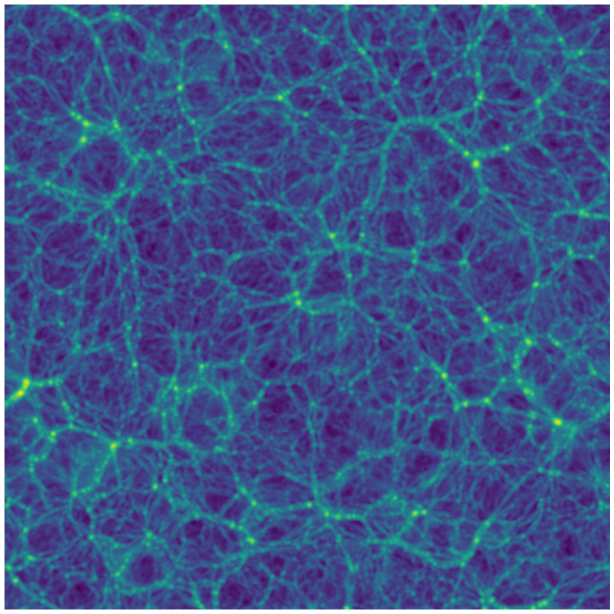

Figure 3: Slice of data cube, the bright spots here are thegalaxy clusters, also called halos and the connecting strand like structures are filaments. Filaments carry gases to galaxy clusters and are instrumental in galaxy formation and evolution.

Another method of comparison can be performed by linearizing volumes along Hilbert space-filling curves and then using a loss function to calculate the variations as explored by [9]. Proper layout design allows for users to understand the visualization better and hence for the tool to be easier to use, which has been explored by [3]. Visualizing Cosmological data generated using procedural generation methods has been explored before by [1], although no comparative visualization has been done yet.

3 N-BODY DATA

The 3D data cube was generated using N-body Monte Carlo Simulation algorithm. It contained 94795 data points as initialization to the algorithm(which were listed along with their mass and density values, and their positions were tracked to create a halos' catalog), with about 10 million agents running in parallel on a supercomputer to generate it. The simulation grid had a resolution of 1200^3 voxels, which was processed and reduced to 1024^3 grid data. The grid's center is (0.00131607, 0.00131989, 84.6443) mpc while the size is (186.282, 186.282, 186.282) mpc. The data contains 1024^3 particles' intensity values which are directly correlated to the mass and density values as provided in the halos catalog. Given data has high dynamic range, with 64-bit floating point value and intensity value range being [0.017891133176195093, 448.9362993707059].

4 DATA PREPROCESSING

We trained both 2D and 3D networks for which the data was processed separately. For the 2D normalized network, we needed slices(as shown in Figure ??) of the 1024^3 cube data as well as the halo map(as shown in Figure **??**). A Halo map is defined as map of white pixels in places where halos are present in the slice, and a constant color where the halos are not present. As discussed later, the constant color value was used in order to avoid collapsing the convolutions. The slices were also converted into images by first taking a log of every value, normalizing all the values and then applying a color map on them. The slices were taken from all three axes. The data was augmented by performing rotations, reflections of data and sampling the slices across 3 different axes.

For the 2D HDR network, although the input and output values were the same, the data was not normalized and was input as a 2D vector. As mentioned in the methods section, the precision of data was reduced from 64-bit floating point to 16-bit floating point to allow training with lower GPU resource consumption. The data was augmented in same manner as the 2d normalized approach.

For the 3D network, the input cube (of size (1024 x 1024 x 1024 vox)) was subdivided into cubes of size (256 x 256 x 256 vox) with a stride of 128 vox.

The data was augmented through axis-aligned rotations, taking reflections of data, and varying the initial slicing position for the cube.

5 METHOD

The training method is divided into three experiments, based on the data pre-processing required. For the first experiment, data was first cut into slices and then normalized to avoid the HDR property. For the second experiment, the slices were directly input to the model without without normalization. For the third experiment, voxels of constant shape were cut from the 3D data cube, which were then input into the model.

5.1 2D Training with Normalization

In this method, the data was normalized into the RGB range (0-255 integer values), which reduced the HDR property of the data. The Generator model used in this training is the Resnet-6 model, which consists of 6 resnet layers with alternate downsampling and upsampling layers. Each of the resnet layers employ skip connections. The Discriminator is the Pixel Discriminator, which provides a 0/1 value according to fake/real classification of each pixel.

5.2 2D HDR Training

In this method, the data was not normalized before being entered into the Generator network. The Generator network used here is the Unet-256 model, which is a fully-convolutional model with upsampling and pooling layers. Although Unet-256 was initially proposed for segmentation tasks, it has been successfully used in image generation tasks as well as the usage of upsampling layers in the later layers allows for increased resolution of the output. The Discriminator is the Pixel discriminator.

Although the data was not pre-processed, the precision of data was reduced from its original 64-bit floating point to 16-bit floating point value. There was no apparent difference in the output that could have been caused by reduced precision.

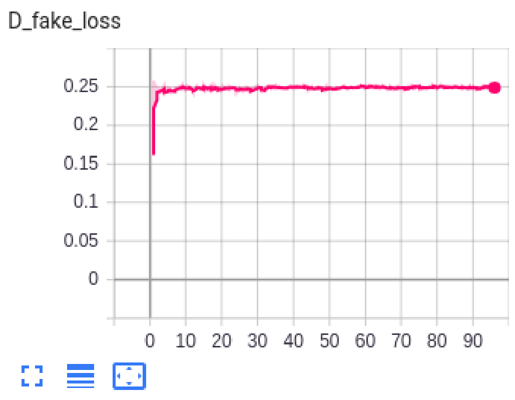

Figure 4: Figure represents the Discriminator loss over the generated image for the 2d normalized gan

5.3 3D Training

For the 3D training, instead of getting slices from the 3D cube, voxels of size (256x256x256) were sampled. A stride of 128 was chosen to allow the network to be able to understand and learn the interconnected nature of data. The Generator model used here is the 3D Unet-256 model, which has the same architecture as the 2D Unet-256 model except with 3D convolutions. The Discriminator used is pixel discriminator.

Data precision was dropped from 64-bit floating point to 16 -bit floating point value for this model in order to fit the model into available GPUs. Since the model was still too large to fit into a single GPU, the discriminator and generator models had to be trained on separate gpus. This lead to longer training time, as the discriminator model had to wait for the generator model to finish training and vice-versa.

6 RESULTS

6.1 2D Normalized GAN

This part of the experiment worked well and the trained gan shows stable loss function output for the discriminator as well as the generator. The output images have very low difference values when subtracted from the real input, showing that the generated output matches the input in the statistical distribution.

The major inconsistency shown by the outputs tends to be in the areas where the filaments are present without any halos at the beginning or end of given filament. Such results are incorrect as filaments cannot exist if not connected by halos. The reasoning for such results can be that since the filaments are 3D structures, a 2D slice is incapable of containing all the information about them. The experiment was repeated with the slices having a small depth value (by concatenating and then averaging multiple slices), but that didn't remove this problem.

6.2 2D HDR GAN

Although both the generator and discriminator remain stable upto the 30th epoch, mode collapse was observed after 30th epoch. The 2D HDR GAN model was able to understand the major filament

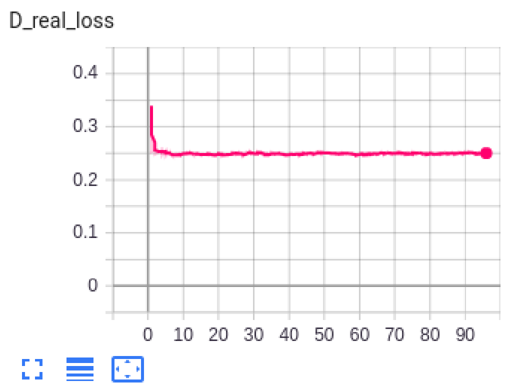

Figure 5: Figure represents the Discriminator loss over the generated image for the 2d normalized gan

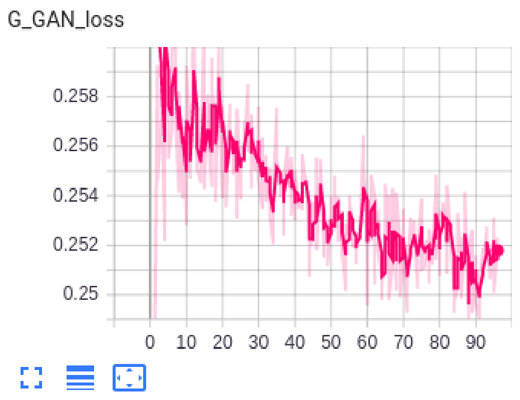

Figure 6: Figure represents the Generator loss as calculated using MSE Loss function for the 2d normalized gan

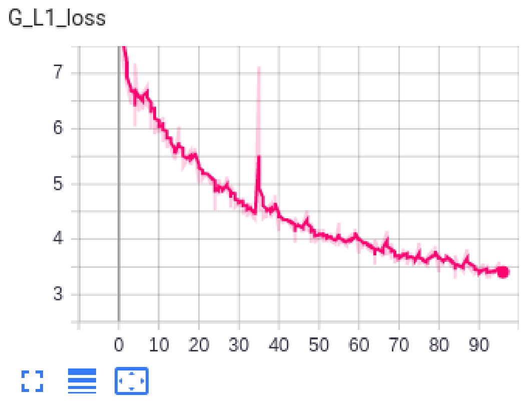

Figure 7: Figure represents the Generator loss as calculated using L1 Loss function for the 2d normalized gan

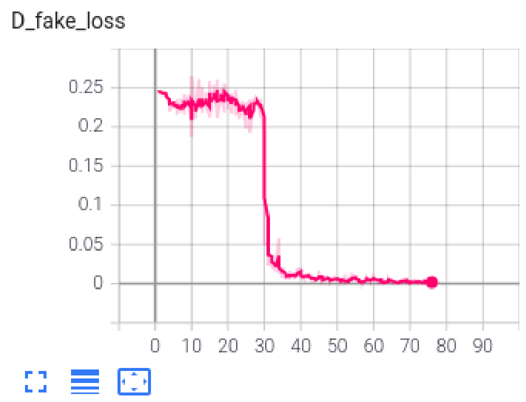

Figure 8: Figure represents the Discriminator loss over the generated image for the 2d hdr gan.

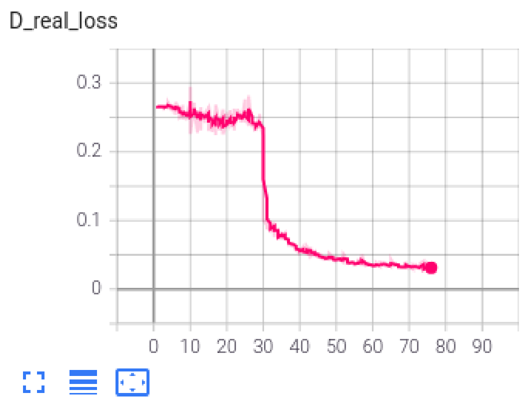

Figure 9: Figure represents the Discriminator loss over the generated image for the 2d hdr gan.

structures present in the data and accurately recreate such structures when working with variable number of halos. The GAN model does have a clamping effect on the High Dynamic Range property of the data, cutting off the extremely high and extremely low values in the data to such effect that the output doesn't need to be normalized to be plotted. Moreover, the intensity values were not consistently generated by the model. When calculating the difference image to observe the difference between input and output 2D vectors, we observed that no multiples of the output lead to a subtracted value close to 0. This shows that the generated intensity values are inconsistent.

When concatenating the output generated using different axes-aligned halo maps (XY vs YZ vs XZ maps), the outputs were seen to be consistent and having maintained the filament structures. Interestingly, even though 2D slices with no depth were considered as data for training, some 3D structure was seen on concatenating contiguous slice outputs and generating a 3D output. The data's inherent 3D structure might be the reason for such output. Running model on non-contiguous halo maps might indicate if the model

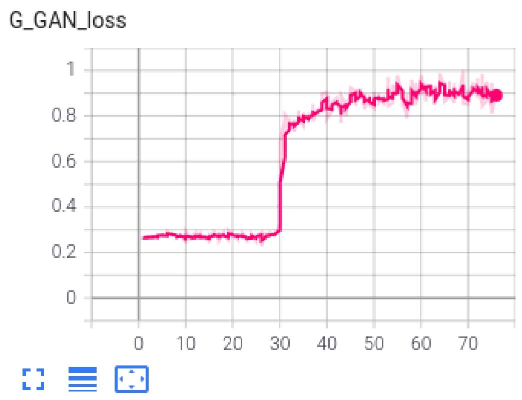

Figure 10: Figure represents the Generator loss as calculated using MSE Loss function for the 2d hdr gan. The sharp increase in the loss at the 30th epoch and sudden decrease in loss for the Discriminator as shown in figures 8 and 9 indicate Mode Collapse.

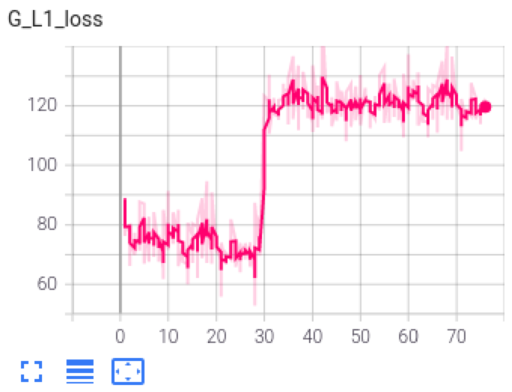

Figure 11: Figure represents the Generator loss as calculated using L1 Loss function for the 2d hdr gan

is biased towards trying to generate 2D vectors that form some structure or not.

6.3 3D HDR GAN

Training using 3D GAN showed better results than the 2D GAN. 3D GAN also didn't show any indication of mode collapse when plotting the loss function, although the gan hadn't converged at the 90th epoch. The output 3D cube is used to perform visualization after slight preprocessing using three.js based code.

7 VOLUME VISUALIZATION

We used GPU-based Ray Casting for Direct VOlume Rendering. Although CPU based approach is also possible, GPU based approach

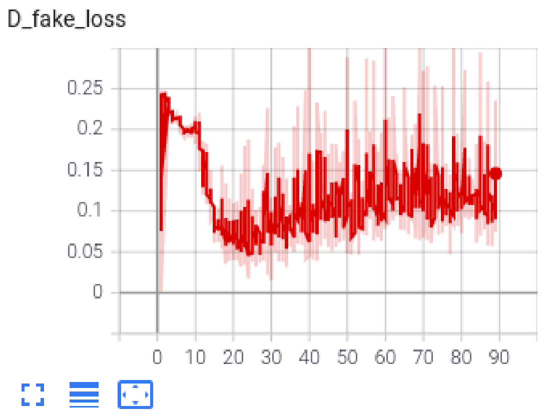

Figure 12: Figure represents the Discriminator loss over the generated image for 3d gan

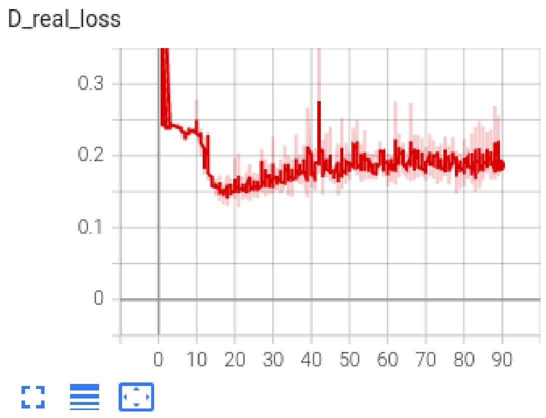

Figure 13: Figure represents the Discriminator loss over the generated image for 3d gan

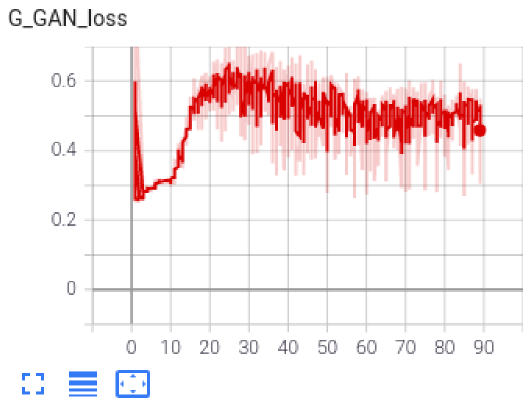

Figure 14: Figure represents the Generator loss as calculated using MSE Loss function for 3d gan

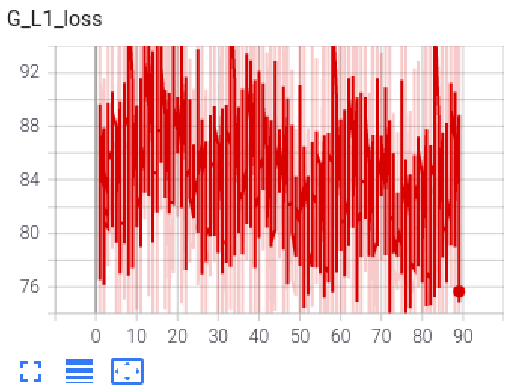

Figure 15: Figure represents the Generator loss as calculated using L1 Loss function for 3d gan

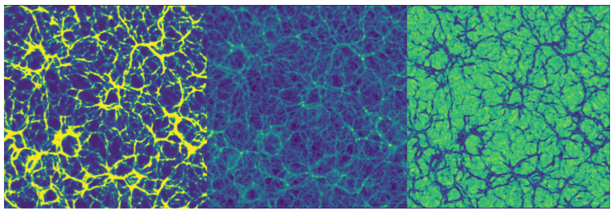

Figure 16: Figure shows the difference between output and input images. On the left is the output image for 2d hdr model, in the middle is the real image and on the right is the difference image. The difference image has a reverted intensity map, indicating that there is a pattern to incorrect predictions of such maps.

is a better fit because of the availability of an optimized graphics pipeline that runs in a loop, thus providing a way to create interactive visualization.

7.1 Using Paraview

Paraview provides a great interface to visualize the volumetric data, providing a fine control over the transfer function. It is possible to clamp the range of values that are mapped by the color range, thus changing the color intensity of each data point. This is really useful for High Dynamic Range, as both high intensity and low intensity values can be easily observed.

Paraview allows users to render data in both Volume and Slice mode. We used the slice mode to show a comparison of the generated cube and the input data. The main observation was the same as seen using image difference as in 16, that the model introduces a clamping to the data, while maintaining the filament structures as shown in 17. We also used the volume mode to visualize the data, varying the color map and the range of color map to observe specific properties of data.

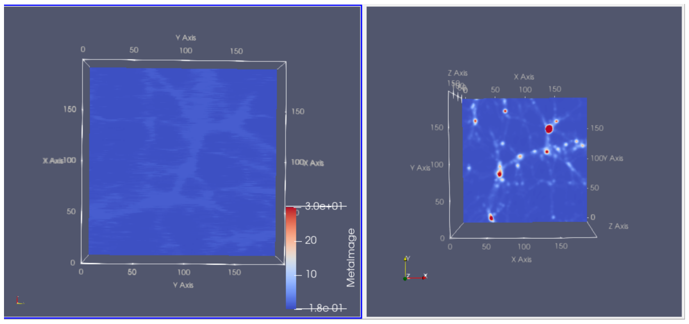

Figure 17: Comparison between slice generated by the 2d model (left) and real image (right). As seen in the difference image, model behaves like a clamping function over the intensity values of halos, but is able to preserve filament structures

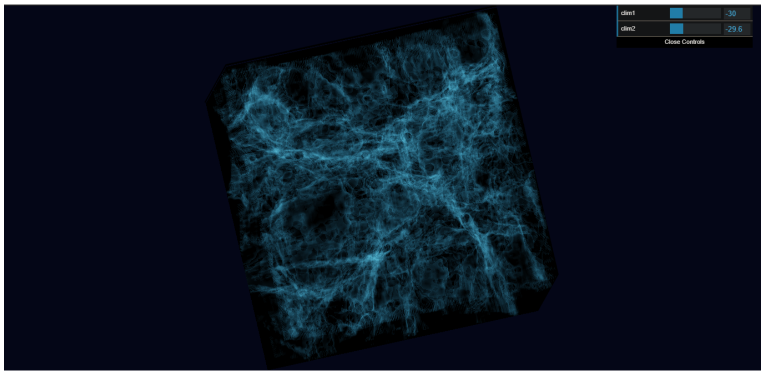

Figure 18: Visualization of the volume, generated using 3D GAN, and then processed to filter out halos with high intensity values. Used raycasting algorithm implemented using three.js

7.2 Using Three.js

Three.js is a library that provides easy access to graphics api for 3D development using WebGL. We used three.js for volume visualization of given data such that the structure of filaments is easily observable as in 18. In order to do that, the data was preprocessed such that the all halos with high intensity values were removed from the given data. This allowed for only filaments to be left visible, providing an interesting visualization of the structure of cosmic web. The current implementation also allows for remapping the scale of transfer function.

8 CONCLUSION

This work represents exploration of Deep Learning methods for Cosmological data generation as well as the visualization of such data. The generation of data was largely successful for 2D & 3D methods, and visualization of various aspects of the system (metrics, loss functions, generated data visualization) helped understand properties of input and generated data well.

9 FUTURE WORK

The idea for the online (three.js based) visualization was for it to act as a server, which could recieve the intermediary 3D results from the model while training, and render those results for easy observation. Moreover, a cell selection mechanism to explore specific part of the visualization was also considered. We would like to work on implementing these capabilities to this system.

The deep learning model has problems with mode collapse. Future work here would be selecting models carefully and preprocessing data to avoid model collapse.

REFERENCES

[1] Oskar Elek, Joseph N. Burchett, J. Xavier Prochaska, and Angus G. Forbes. 2020. Polyphorm: Structural Analysis of Cosmological Datasets via Interactive Physarum Polycephalum Visualization. *IEEE Transactions on Visualization and Computer Graphics* (2020). https://arxiv.org/abs/2009.02441 To appear.

[2] Pratik Kalshetti, Parag Rahangdale, Dinesh Jangra, Manas Bundele, and Chiranjoy Chattopadhyay. 2018. Antara: An Interactive 3D Volume Rendering and Visualization Framework. (December 2018). https://arxiv.org/abs/1812.04233

[3] Sehi L'Yi, Jaemin Jo, and Jinwook Seo. 2020. Comparative Layouts Revisited: Design Space, Guidelines, and Future Directions. *IEEE VIS InfoVis* 1, 2 (June 2020). https://arxiv.org/abs/2009.00192

[4] M. Novotni and R. Clien. 2001. A geometric approach to 3D object comparison. *International Conference on Shape Modeling and Applications* 1, 2. https://doi.org/10.1109/SMA.2001.923387

[5] Nathana¨el Perraudin1, Ankit Srivastava, Aurelien Lucchi, Tomasz Kacprzak, Thomas Hofmann, and Alexandre Refregier. 2019. Cosmological N-body simulations: a challenge for scalable generative models. (2019). https://link.springer.com/article/10.1186/s40668-019-0032-1

[6] Doogesh Kodi Ramanah, Tom Charnock, Francisco Villaescusa-Navarro, and Benjamin D Wandelt. 2020. Super-resolution emulator of cosmological simulations using deep physical models. *Monthly Notices of the Royal Astronomical Society* (2020). https://academic.oup.com/mnras/article-abstract/495/4/4227/5843286

[7] W. Usher, J. Amstutz, C. Brownlee, A. Knoll, and I. Wald. 2017. Progressive CPU Volume Rendering with Sample Accumulation. (2017). http://sci.utah.edu/~will/papers/savr/savr.pdf

[8] F. Vega-Higuera, P. Hastreiter, R. Fahlbusch, and G. Greiner. 2005. High performance volume splatting for visualization of neurovascular data. IEEE Viz. (October 2005). https://ieeexplore.ieee.org/document/1532805

[9] Johannes Weissenbock, Bernhard Frohler, Eduard Groller, Johann Kastner, and Christoph Heinzl. 2019. Dynamic Volume Lines: Visual Comparison of 3D Volumes through Space-filling Curves. *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS* 25, 1 (January 2019). https://www.cg.tuwien.ac.at/research/publications/2019/Weissenboeck_2019/Weissenboeck_2019-Paper.pdf