Server-Side Rendering of Large Astronomical Data Cubes

Jonathan Weideman
Department of Computer Science
University of Cape Town
South Africa

April 2020

Abstract

In this literature review, the current astronomical visualisation software applications are analysed to determine their efficacy and efficiency in visualising the large volumes of data being produced by modern radio telescopes. Volumetric rendering is then introduced as a potential technique for rendering multi-dimensional astronomical data. We conclude that a future application, in order to meet the requirements of radio astronomers for visualisation of large data sets, would require a client-server architecture, use volumteric instead of texture-based rendering, perform rendering on the server side before streaming the compressed rendered images to the user, support various features which allow interaction, and be built opensource with high modularity to allow the development of plugins which could be shared between users.

CCS CONCEPTS

• **Computing methodologies** ~ Computer graphics
• **Human-centered computing** ~ Visualisation
• **Human-centered computing** ~ Interaction design

KEYWORDS

Volumetric Rendering, Texture-based Rendering, Client-server Architecture, Volume Ray Casting

1 Introduction

The development of radio telescopes (such as the Square Kilometer Array [15] currently under construction in South Africa) has presented new opportunities in the field of radio astronomy. The quantity and quality of the data collected by these telescopes requires novel techniques for analysis and interpretation in order to extract the maximum amount of useful information. This data comes in the form of data “cubes”, meaning it is three dimensional (two positional and one spectral) and can be visualised as a stack of 2D images captured along the radiofrequency spectrum [4]. Analysis of astronomical data at different frequencies allows astronomers to detect certain astronomical phenomena which would otherwise be undetectable. For example, neutral hydrogen (H1) surveys at the 21-cm sub infra-red emission-line can be used to infer important information such as the star formation rate [11] and cold gas accretion [14] of galaxies.

It has recently been pointed out that there is a lack of an application that can deal with large astronomical data cubes [4][9]. Currently, there are only a few tools which provide 3D rendering and only for a small subset of a larger data cube (see section 2). When working with a large data set, these tools have required the user to step through the data cube viewing one rendered 2D image at a time until the desired frequency or collection of frequencies is found. This process is inefficient since it requires the user to perform many interactions with the data and attempt to construct a mental 3D visualisation from a collection of 2D images. Although this technique is useful in certain cases, the addition of a 3D representation of this data using volumetric rendering along with tools which would provide interaction and flexibility for cubes larger than what can be stored on a personal computer (PC) would be of great use to the next and current generation of radio astronomers [4].

In this literature review, the current astronomical visualisation tools are analysed as well as their shortcomings in terms of their efficacy and efficiency in visualising 3D data, 3D volumetric rendering is then introduced as potential technique to overcome these shortcomings and this technique is explored in detail along with a review of its current implementation in astronomy and other scientific fields. Finally, in the conclusions section, a potential tool or plugin is outlined which would utilize this volumetric rendering technique and draw upon the tools analysed in order to satisfy the needs of radio astronomers for 3D visualisation of large data sets.

2. Review of contemporary 3D visualisation tools

Many 3D data visualisation tools exist, however in this review only those which are open source, publicly available and continuously maintained by developers are analysed. This review is required in order to avoid duplication of software and features which have already been developed and are in use.

2.1 KARMA

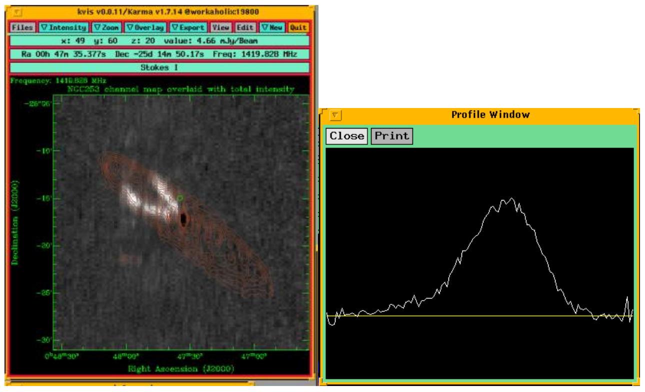
kvis v0.0.11/Karma v1.7.14 @workaholix:19800
Files
▽ Intensity
▽ Zoom
▽ Overlay
▽ Export
View
Edit
▽ New
Quit
x: 49 y: 60 z: 20 value: 4.66 mJy/Beam
Ra 00h 47m 35.377s Dec -25d 14m 50.17s Freq: 1419.828 MHz
Stokes I
Frequency: 1419.828 MHz
NGC253 channel map overlaid with total intensity
Declination (J2000)
Right Ascension (J2000)
Profile Window
Close
Print

Figure 1: The user interface of Karma. Left: A 2D rendered image superimposed with a contour lines for intensity values. Right: A position velocity profile corresponding to a slice specified by a user.

Karma [3] is a general-purpose programmer's toolkit and is made up of KarmaLib, a structured library and application programming interface (API), and several modules or applications to perform specific tasks. Karma is a serverless and non-network application. Hence, all input data cubes must be stored on the host machine. This will often prevent users from being able to visualise large data cubes when running the application on their PCs due to lack of storage capacity. For example, the typical survey conducted by MeerKAT will have 8K × 8K bits spatial resolution and up to 32K spectral resolution [6], which would give a typical size of 8K × 8K × 32K × 4 (assuming 32-bit floats) ≈ 8 TB at full spectral and spatial resolution. These data cubes are clearly too large to be stored on most PCs without access to an external hard drive, most of which have far less than 8 TB of storage capacity.

Karma's rendering is done by the CPU as opposed to a graphical processing unit (GPU). This, together with the lack of memory capacity of PCs, imposes significant constraints on the size and resolution of data which can be visualised using Karma.

There is also the requirement that data cubes used for volumetric rendering in Karma must have bit values in the range of -127 to 127. This is a severe limitation for astronomers as images often contain very bright sources which would be stored as bit values higher than 127. For these data cubes to be rendered, they would have to be scaled to fit into this range which would result in a significant loss of information.

The main visualisation features supported by the library include:

- Volume rendering of data cubes (see section 3).
- Play movies of data cubes - allowing the user to step through frames of a data cube. E.g., a cube where each step in the movie corresponds to a different frequency.
- Inspecting multiple images and cubes at the same time - allowing the user to compare several datasets at the same time and apply different display settings to each.
- Slice a cube - used to display three orthogonal slices through the cube where each slice is along one of the principle planes (X, Y and Z).
- Superimposing images.

- Interactive position-velocity slices - allowing users to use a two-dimensional image to define a slice through a three-dimensional data cube.
- Interactive co-ordinate placement - allowing users to quickly place a co-ordinate system onto images which do not have one.
- Rectangular to polar gridding of images - which allows the user to easily alternate the co-ordinate grid between polar and rectangular.

2.2 SAOImage DS9

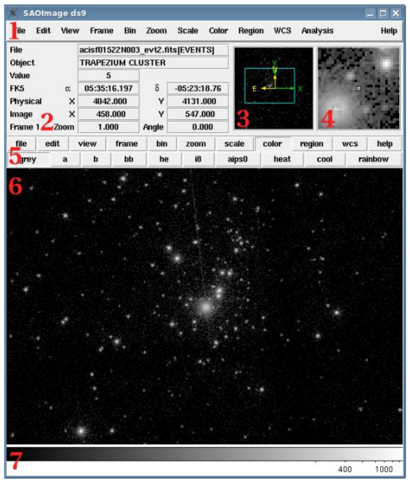
SAOImage ds9
1
File Edit View Frame Bin Zoom Scale Color Region WCS Analysis Help
2
3
4
5
file edit view frame bin zoom scale color region wcs help
grey a b bb he i8 aips0 heat cool rainbow
6
7
400 1000

Figure 2: The user interface of SAOImage DS9.

SAOIMage DS9 [7] is another widely used astronomical data imaging and visualisation application. As with Karma, all rendering is done on the client's CPU, and the input data cubes are required to be stored on the client's PC. Hence it imposes similar constraints on the cube size and resolution depending on the user's CPU(s) performance and storage capacity.

DS9 is a standalone application and requires no installation. It also provides access to web-based archive servers such as the Mikulski Archive for Space Telescopes (MAST), SkyView and many more through FTP and HTTP. All data cubes retrieved in this manner are required to be entirely copied into the memory of the user's machine before rendering can take place.

DS9 does not support volumetric rendering, hence it requires astronomers to visualise data cubes in "movies" - by stepping through the data cube one rendered 2D image at a time. Another notable feature supported by DS9 is inter-process communication using the Simple Application Messaging Protocol (SAMP) [16]. This is a standard for the exchange of data between participating client applications and is used by many applications dealing with astronomical data.

2.3 CARTA

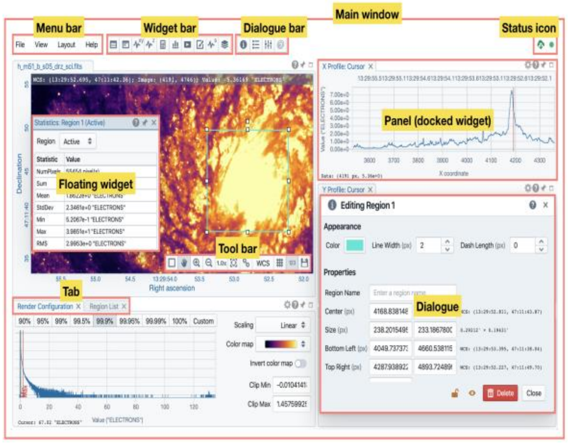

Figure 2: The user interface of CARTA with labels of widgets superimposed.

The Cube Analysis and Rendering Tool for Astronomy (CARTA) [1] is an application used for the analysis and visualisation of astronomical data. Its mission is to provide usability and scalability into the future as the size and detail of radio images increases as more advanced radio telescopes are built. It plans to achieve this through exploiting modern web technologies, computing parallelization and modern GPUs.

CARTA adopts a client-server architecture. It does this to allow users to visualise data cubes too large to be stored on PCs. It comprises of two versions: the desktop (server) version and the remote (client) version. The server is responsible for data cube storage, retrieval of a subset of a data cube which is selected by a client, compression and any other preprocessing. The remote version is used for accessing the CARTA server remotely, GPU-accelerated rendering, displaying the image and interaction from the client. The desktop version allows users to run a CARTA server on their own PC. The main use case for this is when a user has a data cube stored locally which is not stored on the CARTA server.

CARTA currently does not support volumetric rendering. Similarly to SAOImage DS9, it only allows visualisation of 3D data cubes through 2D rendered slices. However, CARTA supports several features which are noteworthy:

- Tiled rendering – the separation of a graphical image into a regular grid so that each section of the grid (or tile) can be rendered independently of the others and at the same time (in parallel).
- Cursor information – displaying information about a pixel where the cursor is positioned at the top of the screen.
- Region of Interest (ROI) – allowing the user to draw or select a specific region in the image which can then be separated for further analysis.

2.4 SlicerAstro (plugin for 3DSlicer)

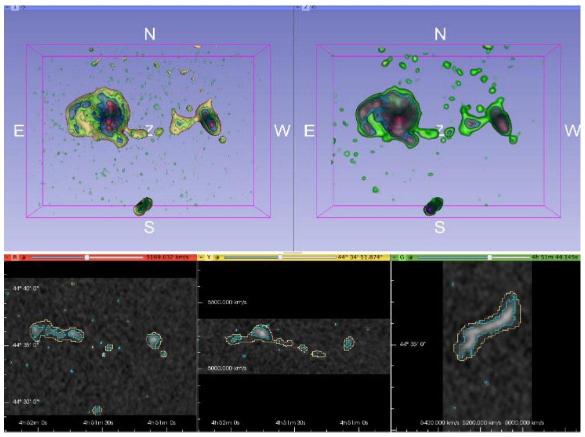

Figure 3: The user interface of SlicerAstro. Top left: A 3D render before smoothing is applied. Top right: A 3D render after smoothing is applied. Bottom: Paired 2D views of the same data with intensity contours superimposed.

3DSlicer [8] is an application for the analysis and visualisation of medical images. It is supported by multiple operating systems including Windows, MacOSX and Linux, supports both CPU and GPU rendering, supports 3D volumetric rendering and is extensible to allow the development and use of plugin applications and algorithms.

SlicerAstro [12] is one such plugin, which includes several capabilities that are particular to astronomical applications, such as:

- Support for the Flexible Image Transport System (FITS) file format, which is the most widely used file format in astronomy.
- Astronomical co-ordinate systems.
- Interactive 3-D modeling (rotation, zoom, etc.).
- Coupled 1-D/2-D/3-D visualisation with linked views.
- Support for the SAMP protocol.
- Generation of flux density profiles and histograms of the voxel intensities.

3DSlicer (and hence SlicerAstro) is a serverless application, and all rendering is done using the host machine's CPU and GPU. Punzo et al. reported that SlicerAstro provides interactive performance when rendering data-cubes of dimensions up to 10^7 voxels and very fast performance (<3.5 sec) for larger ones (up to 10^8voxels) [12]. SlicerAstro also makes use of a several smoothing functions, most notably of which is the intensity-driven gradient filter. This filter preserves the detailed structure of the signal while smoothing the faint part of it, improving visualisation and feature detection. An example of this shown in figure 4.

3 Volumetric Rendering

Volumetric rendering is a technique used in computer graphics to visualise a 3D discretely sampled data set on a 2D display [2][10]. The data usually comes in the form of a "cube" or stack of 2D images of the same dimensions (i.e. the same amount of pixels), with each image on the stack usually acquired in a regular pattern (e.g. one every unit of time or distance).

Previous approaches to visualising 3D data utilize computer graphics techniques which attempt to reduce the volume array to a set of 2D geometric primitives (usually triangles) linked together in a mesh in order to approximate the surfaces of objects contained in the data. This technique, known as texture mapping (or texture-based rendering) [5], is inefficient when rendering objects which have a branching structure, especially when there is a high density of branching relative to the overall size of the object. This is because only the surfaces of objects are rendered when using this technique and branching structures rapidly increase the surface area of objects, making them more computationally intensive to render. Animators and computer game developers have had to circumnavigate this problem either by using various techniques to reduce the complexity of the object (e.g. multiresolution rendering [13]) or exclude them altogether. Thus, texture-based rendering is undesirable when high detail is required or branching objects are being rendered.

Texture-based rendering also proves inadequate when users require the insides of objects to be rendered, since this technique only renders the surfaces of objects and excludes all detail found within. This is most notably seen in scientific visualisation when users wish to interact with, move inside, cut or disassemble objects with high precision. For example, a doctor may wish to visualise the data acquired from an MRI scan in a 3D interactive environment so they can view the inside of a patient's body.

Volumetric rendering aims to solve these problems by generating images of 3D data without explicitly extracting geometric surfaces from the data [10]. This technique directly renders each volumetric element (or voxel), which can be thought of as a 3D pixel, where the values associated with each voxel are calculated by sampling the immediate area surrounding the it. Each voxel has an opacity and colour value, which is usually calculated by the RGBA transfer function. This function maps a numerical value to a colour and opacity.

3.1 Volume Ray Casting

Volume ray casting, also referred to as ray marching, is the technique utilized by volumetric rendering whereby a 2D image is created from a 3D dataset. In this technique, for each pixel of the final image, a viewing ray is caste through the volume. Along its path, samples of opacity and the colour of voxels are recorded and accumulated at equal intervals. In certain cases, the sampling point might occur between voxels, in which case it is necessary to interpolate values from the surrounding voxels. After all sampling points have been shaded (i.e. coloured and lit), they are composited along the viewing ray, and the resulting opacity and colour value assigned to the corresponding pixel. This process is completed for each pixel on the screen until the final image is produced.

Due to the parallel nature of this technique, since each viewing ray can be computed in parallel, modern GPUs with multiple cores are best suited to the task of volumetric rendering. Scalability into the future is also resolved in this way as more GPUs can simply be added as the size of the data cubes increase.

4 Discussion

The various astronomical data visualisation software discussed in section 2 as well as many others which are currently being used have been designed for visualisation of data with file sizes small enough to be stored on PCs and laptops. The next generation of radio telescopes will produce data cubes several TBs in size [4], and these cubes can be stitched together to form even larger cubes. It was shown that CARTA supports data cubes of this size through its use of a client-server architecture which allows the use of storage capacity as well as powerful CPUs which are usually inaccessible to users. However, CARTA does support 3D rendering of these data cubes or any subsets of them – it only allows the user to view 2D renders of the 3D data cubes, and this rendering is done by the users GPU.

SlicerAstro (section 2.4) was shown to support 3D rendering, however this rendering is also done on client GPUs. SlicerAstro does provide several features which are of great utility to astronomers for visualising galaxies and for tasks such as feature spotting and real time interaction (rotation, zoom, etc.). However, this can only be done for data cubes which are small enough to be stored on the host machine. It was also shown that the interactivity and performance of AstroSlicer begins to decrease when rendering > 10^7 voxels. The typical survey conducted by MeerKAT contains approximately 2×10^{12} voxels [6], i.e. AstroSlicer can provide high performance for data cubes 2×10^5 times smaller than a single data cube captured by MeerKAT. Thus, AstroSlicer would require significantly greater computational power in order to render a single data cube produced by MeerKAT whilst maintaining an acceptable interactive performance. Punzo et al. did not state the specifications of the machine used during these performance tests which makes it is impossible to determine which out of random-access memory (RAM), the CPU or GPU was the bottleneck of the performance for SlicerAstro. More research is needed to be done on this, and projections made on the performance that can be expected if rendering is done using advanced computational hardware.

The features, architecture and customer reviews of SlicerAstro can be drawn upon when developing an application capable of volumetrically rendering and visualising large data cubes. The rendering equation as well as the transfer function could directly be used in more advanced software capable of visualising much larger cubes, or they could prove to be inefficient but capable of being refactored and optimized in order to be used in future applications. Interviews could be conducted and customer reviews pooled to determine which features of SlicerAstro provide the most utility and which are regarded as non-essential for future iterations.

Karma is a good example of a successful astronomical visualisation tool since it was developed in the 1990's and is still in wide use today. Its robustness and wide variety of features is what makes it the tool of choice for many astronomers and should be emulated in future applications. The limitations of Karma which makes it unsuitable for large data sets were its serverless architecture and reliance on user's machine as well as the limitations it sets on bit values during volumetric rendering, leading to a loss of information.

SAOImage DS9, likewise with Karma, requires all rendering to be done on the user's machine and does not support 3D rendering. Hence, it suffers from similar constraints in terms of data cube size. Its feature which distinguish it from the rest in this review is its remote server access of data cubes. This can be considered a quality of life feature since this can be done via a web browser, however it should be considered in future applications in order to improve the overall user experience and ease of use.

The astronomical data captured by radio telescopes is usually in a branching structure or is cloud-like. Galaxies, solar systems and gas clouds are permeated throughout space and are comprised of many smaller, disjointed features. This makes texture-based volumetric rendering inefficient when rendering astronomical data, since the algorithm either has to form surfaces which enwrap multiple smaller objects, leading to a loss of information, or form countless smaller surfaces surrounding individual objects, leading to an exponential increase in computational complexity. Another drawback from wrapping surfaces around multiple objects, for example an H1 gas cloud, is that if the user were to position the view of the visualisation inside the cloud after the render had been completed, nothing would be visible (i.e. it would appear empty inside). Since radio astronomy is a scientific field and interactivity of the 3D virtualization is regarded as essential, high precision and accuracy is required during data analysis and information loss must be avoided at all costs.

This leaves volumetric rendering (section 3) as the rendering technique of choice for 3D visualisation of astronomical data. This technique does not attempt to explicitly extract 3D geometric surfaces and instead directly renders each voxel. This maintains the high detail of cloud-like and branching structures found in astronomical data and allows them to be visualised and interacted with in 3D. Since the volumetric ray casting algorithm can be executed in parallel, rendering can be conducted on modern GPUs or clusters of GPUs. A client-server architecture would likely be a requirement to grant access to such hardware which the majority of users would not have access to. This would allow users to visualise data cubes too large to be stored on their machine, and grant access to a cluster of GPUs as opposed to a single one running on their machine, potentially in an interactive environment with acceptable performance.

5 Conclusions

In this review, various astronomical visualisation software applications were analysed considering recent developments in radio astronomy telescopes and the magnitude of data they are producing. A new application will be required in order to visualise these large data sets in 3D. This application will require a client-server architecture which would allow the exploitation of larger data storage hardware unavailable to most users on their PCs. Volumetric rendering would be the technique of choice over texture-based rendering due to the nature of the data produced by radio telescopes and the types of interaction required by astronomers with the data. This rendering must be done on the server side which would allow for the use of clusters of modern GPUs, after which the rendered images are compressed and streamed to users to be displayed. Several features common to many current astronomy visualisation applications were reviewed in order to consider which should be included in future applications. The software should be modular and opensource to allow development of further plugins and tools which might be required in specific use cases.

REFERENCES

[1] COMRIE, A., WANG, K., FORD, P., MORAGHAN, A., HSU, S., PIŃSKA, A., CHIANG, C., JAN, H., SIMMONDS, R., CHANG, T., LIN, M., CARTA: The Cube Analysis and Rendering Tool for Astronomy, (Dec 2018). Retrieved May 10, 2020 from https://doi.org/10.5281/zenodo.3377984

[2] DREBIN, R.A., CARPENTER, L., AND HANRAHAN, P., Volume rendering. *In Proceedings of the 15th annual conference on Computer graphics and interactive techniques (SIGGRAPH '88)* 22, 4 (Aug 1988), pp. 65-74.

[3] GOOCH, R.E., JACOBY, G.H., AND BARNES, J., Karma: a Visualisation Test-Bed. *Astronomical Data Analysis Software and Systems V* 101 (1996), pp. 80-83.

[4] HASSAN, A., AND FLUKE, C. J., Scientific Visualization in Astronomy: Towards the Petascale Astronomy Era. *Publications of the Astronomical Society of Australia* 28 (2011), pp. 150-170.

[5] HECKBERT, P.S., Survey of Texture Mapping. *IEEE Computer Graphics and Applications* 6, 11 (Nov 1986), pp. 56-67.

[6] JONAS, J.L. ET AL., The MeerKAT Radio Telescope. *Proceedings of Science* Volume *277 - MeerKAT Science: On the Pathway to the SKA (MeerKAT2016)* (Feb 2018)

[7] JOYE, W. A., GABRIEL, C., ARVISET, C., AND PONZ, D. ET AL., Astronomical Data Analysis Software and Systems XV, *Astronomical Society of the Pacific Conference Series* 351 (July 2006), p. 574.

[8] KIKINIS R, PIEPER SD, AND VOSBURGH K., 3D Slicer: a platform for subject-specific image analysis, visualization, and clinical support. *Intraoperative Imaging Image-Guided Therapy* 3, 19 (2014), pp.277–289.

[9] KORIBALSKI, B. S., Overview on Spectral Line Source Finding and Visualisation. *Publications of the Astronomical Society of Australia* 29 (2012), pp. 359-370.

[10] LEVOY, M. Display of Surfaces from Volume Data. *IEEE Computer Graphics & Applications* 8, 2 (1988), pp. 29–37.

[11] M. SCHMIDT. The Rate of Star Formation. *The Astrophysical Journal* 129 (March 1959), pp. 243.

[12] PUNZO, D., VAN DER HULSTA, J.M., ROERDINKB, J.B.T.M., FILLION-ROBINC, J.C., AND YUDE, L., SlicerAstro: A 3-D interactive visual analytics tool for HI data, *Astronomy and Computing* 19 (April 2007), pp. 45-59.

[13] RIBELLES, J., LÓPEZ, A., BELMONTE, O., REMOLAR, I., AND CHOVER, M. Multiresolution Modeling of Arbitrary Polygonal Surfaces: A Characterization, *Computers & Graphics* 26, 3 (2002), pp. 449-462.

[14] SANCISI, R., FRATERNALI, F., OOSTERLOO, T., AND VAN DER HULST, T. Cold gas accretion in galaxies. *Astronomy and Astrophysics Review* 15, 3 (2008), pp. 189-223.

[15] SCHAUBERT, D.H, BORYSSENKO, A.O. VAN ARDENNE, A., BIJ DE VAATE, J.G., AND CRAEYE, C. The square kilometer array (SKA) antenna. *IEEE International Symposium on Phased Array Systems and Technology 2003* (Oct 2003), pp. 351-358.

[16] TAYLOR, M.B., BOCH, T., AND TAYLOR, J., SAMP, the Simple Application Messaging Protocol: Letting applications talk to each other, *Astronomy and Computing* 11 (June 2015), pp. 81-90.