3D visualization of large astronomical data cubes, towards a client-server based solution

A literature review

Shuaib Parker
University of Cape Town
prkshu001@myuct.ac.za

ABSTRACT

Astronomy has experienced large technological advancements in the last few years. Modern telescopes collect data at a previously unprecedented rate. This new wealth of data presents a great opportunity for scientific advancement in astronomy.

Three-dimensional (3D) visualization becomes a key feature in allowing astronomers to reason about these large sets of complex data. However, this data boom also presents astronomers and software developers with a new set of challenges, as visualization tools need to be adequately equipped to deal with the ever increasing data size.

We find that existing visualization tools focus mainly on 2D visualization and simply cannot process large data cubes efficiently. Good software architecture, such as a client-server approach, becomes key in mitigating this issue since complex data processing can be offloaded to a remote server. We also highlight the importance of harnessing the power of the graphics processing unit (GPU) to speed up rendering and allow the user to properly interact with the 3D model.

CCS CONCEPTS

• **Applied computing → Physical sciences and engineering** .

KEYWORDS

Visualization, Visualization Tools, Astronomic visualization

1 INTRODUCTION

Astronomy has always been a visual science. Astronomers often need to analyze large sets of complex data and visualization helps immensely in this regard [16]. The need for visualization has only increased since then, as modern telescopes collect data in greater detail and magnitude than ever before [9]. The data collected by these telescopes are often multi-dimensional, and are commonly referred to as astronomical data cubes. For the rest of this paper we will simply refer to them as *data cubes*.

Despite the multi-dimensional nature of these data cubes, most current software packages only offer 2D visualizations [2]. While 2D visualizations are useful in their own right, it is clear that 3D visualizations can provide further insight during data analysis [16, 24]. It's also important to note that these 3D visualizations need to be rendered efficiently enough to allow for real-time user interactivity.

Although there have been several attempts at 3D visualization [14], there seems to be a severe lack of modern 3D visualization tools for astronomy.

The large size of these modern data cubes also poses a serious problem, since most existing software was not designed to deal with data cubes many gigabytes large, resulting in poor performance on a typical workstation.

With these issues in mind it becomes clear that the ideal modern visualization tool would need to:

- Be capable of good 3D visualization
- Have the ability to run on a modestly powered workstation at adequate speeds
- Allow for a high level of interactivity

In this paper we will review the various tools and techniques used for 3D visualization in astronomy and determine whether it satisfies the criteria above. This will allow us to learn from their strengths and avoid their weaknesses when designing a visualization tool of our own.

2 WHY 3D VISUALIZATION?

Why the need for 3D visualizations at all? The first argument can be found in the nature of the data cubes themselves. They are inherently three-dimensional, containing two spatial axes and a third axis (usually frequency). Figure 1 displays the typical structure of the data cube

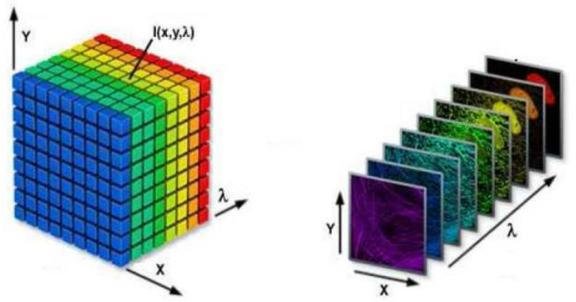


Figure 1: Typical structure of an astronomical data cube. X and Y represent the spatial axes, while lambda represents the frequency axis. Retrieved from https://www.researchgate.net/figure/the-spectral-data-cube_fig10_274710096

It's natural to assume that visualizing this data cube in 3D could aid analysis and provide insight not readily apparent in 2D. 3D models also aid users in evaluating the data qualitatively [24]. This is essential when dealing with large amounts of data, since it is near impossible to fully comprehend every detail of such large amounts of data.

3D models also offer a more holistic overview of the data. Instead of stepping through a 3D data cube piece by piece, the user can get a more global view to aid analysis. This global view could also be key when performing quality control on the data cube [6].

3 CURRENT 2D AND 3D VISUALIZATION TECHNIQUES

Most of the current astronomical visualization tools focuses on 2D visualization. Typically these 2D visualizations display a 'slice' of the data cube to the user. The user can then typically adjust the third axis with a slider and step through the cube in that way. While adequate for certain types of analysis, these types of 2D visualizations put a large cognitive strain on the user, as they have to remember the information displayed in the previous frames and don't ever get a full overview of the data [10].

There are quite a number of established 3D visualization methods employed by fields such as medicine, gaming and indeed astronomy. One particularly useful set of techniques for astronomy is *volume rendering* [5].

Volume rendering is a set of techniques that are used to display a 2D projection of a 3D data set. Volume rendering can be broken up into two distinct types: indirect volume rendering and direct volume rendering.

3.1 Indirect volume rendering

Indirect volume rendering (otherwise known as surface rendering) tries to transform a 3D data set into a set of geometric primitives representing the data, and then attempts to render this new image. After an initial pre-processing step, each subsequent render can be done quickly. Indirect volume rendering is often used in fields such as gaming [12], in order to create 3D objects such as terrain and characters.

However, not all 3D objects and data can be sufficiently approximated by geometric primitives. Astronomical data in particular falls into this category, since they typically do not have well defined surfaces [10]. This can result in significant noise errors and rendering artifacts.

3.2 Direct volume rendering

Direct volume rendering methods on the other hand, attempts to directly render the data on a 3D grid. The grid can be broken down to individual elements called voxels (which can be thought of as the 3D equivalent of pixels). Direct volume rendering is computationally expensive, but it does not suffer from the type of noise errors found in indirect volume rendering methods. Examples of direct volume rendering methods are ray-tracing [20] and ray-casting [22].

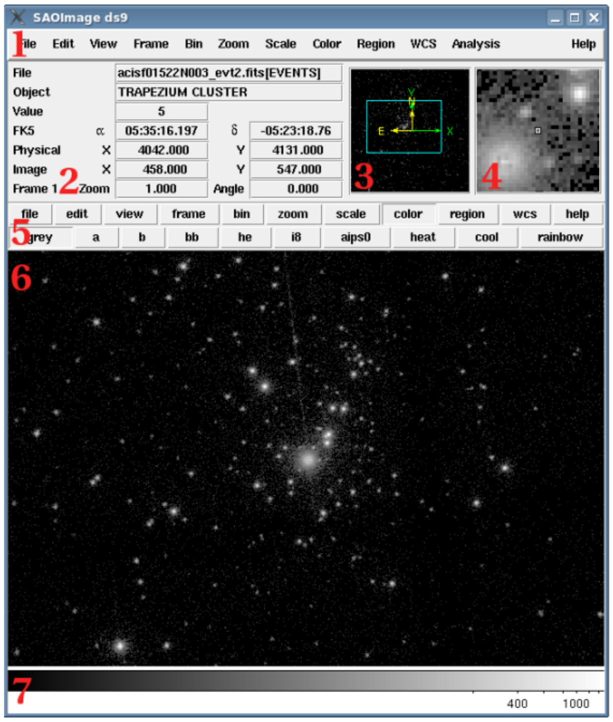
4 EXISTING VISUALIZATION TOOLS

There are quite a number of existing visualization tools for astronomical data. This section will analyze some of the most popular tools in the field, and highlight their strengths and weaknesses.

4.1 SAOImage DS9

One of the more popular pieces of visualization software for astronomy is SAOImage DS9. Initially developed in 1990, it still enjoys widespread use and is actively maintained to this day. DS9 supports the Flexible Image Transport System (FITS) [25] format, which is the most commonly used file format in astronomy, as well as binary tables and multiple other file formats [11].

DS9 provides a graphical user interface (GUI) that allows the user to view a 2D slice of the data cube along different axes. The user can then adjust a slider to view different slices of the cube. Figure 2 is an example of the DS9 user-interface, found in their user manual.


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

File acisf01522N003_evt2.fits[EVENTS]
Object TRAPEZIUM CLUSTER
Value 5
FK5 α 05:35:16.197 δ -05:23:18.76
Physical X 4042.000 Y 4131.000
Image X 458.000 Y 547.000
Frame 1 Zoom 1.000 Angle 0.000

Figure 2: A screenshot of the DS9 user interface. Retrieved from http://ds9.si.edu/doc/user/gui/index.html.

More recently, DS9 has added support for 3D visualization by making use of a direct volume rendering technique called ray-tracing [4]. However, this 3D visualization functionality suffers from performance issues, since it does not make use of the graphics processing unit (GPU) to aid rendering. This slows down rendering considerably and makes interacting with the data fairly cumbersome.

DS9 is also requires the entire data cube to be loaded into the main memory of the end user's workstation before processing and rendering it, making it simply impossible to handle very large data cubes [9]. These issues make it less than ideal for modern use.

4.2 KARMA

Another widely used visualization tool is KARMA [7]. First developed in 1996, it is still being used by astronomers today.

KARMA allows the user to view a 2D slice of the data cube along the frequency axis. Typical features for data analysis of the slice, like zooming and panning are also supported. While the 2D visualization features available in KARMA seem adequate, the user-interface itself is not very user-friendly [21]. Simple tasks force the user to navigate through multiple windows and it requires a high degree of technical knowledge to operate.

KARMA later introduced their XRAY package, which introduced 3D volumetric rendering capabilities, but it's performance is substandard [1] and as a result, offers limited interactivity.

KARMA is no longer being actively maintained, and was not initially designed to cater for massive amounts of data, making it fairly cumbersome for modern data analysis.

4.3 CARTA

CARTA [3] or the *Cube Analysis and Rendering Tool for Astronomy*, is one of the more modern visualization tools. The key feature that makes CARTA interesting is it's client-server model. All of the CPU and memory intensive tasks are performed by a remote server, easing the burden on the end user's workstation. The server then sends a compressed subset of the data to the client, which renders the image using GPU-accelerated techniques [15]. This architecture allows CARTA to run at impressive speeds, loading terabytes worth of data in a couple of seconds [17]. In addition, the client-side rendering allows visual changes to reflect on the users end rapidly, without the need to fetch the model from the server again. However, this does mean that the end user's workstation needs to be equipped with decent on-board GPU.

CARTA is also built to be extensible through a plug-in framework. Developers who want to modify or extend CARTA's capabilities can make use of existing API's and write their own plugins in Python or C++. CARTA is also open-source, allowing multiple developers to update and maintain it's code. A desktop version of CARTA also exists where the 'server' code can be loaded on the end user's workstation. This can prove useful when working in an area with limited internet connectivity, but this version is not suitable for large data sets.

Unfortunately, CARTA only offers 2D visualization, allowing the user to view a slice of the data cube along different axes. No 3D visualization features are currently available. Figure 3 shows an example screenshot of the CARTA interface

While CARTA's client-server architecture yields promising results for handling large data sets, it's lack of 3D rendering capabilities are a significant drawback. However, CARTA is highly extensible, which means that it should be possible to integrate a 3D visualization into it.

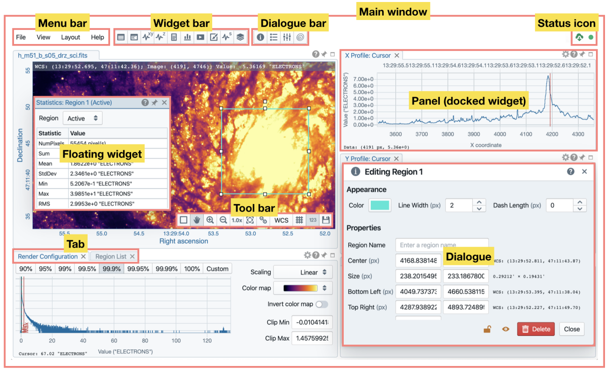


Figure 3: A screenshot of the CARTA GUI. Retrieved from https://carta.readthedocs.io/en/latest/about_gui.html

4.4 3D slicer and SlicerAstro

3D slicer [13] is an open-source 3D visualization package. It is free to use and has a large number of active developers. It boasts powerful tools for 3D visualization such as both CPU and GPU powered rendering using the *Visualization Toolkit* (VTK) [8] interface. 3D slicer itself is not an astronomical tool. It was initially designed for use in the medical field for visualizations such as MRI scans. However, the package has a plug-in mechanism that allows extensions to be developed for usage in other fields. One such extension is SlicerAstro [19].

The SlicerAstro extension is geared towards astronomic visualization, particularly that of HI galaxies. It uses a direct volume rendering method called ray-casting [22] to render 3D views of the data. Ray-casting produces high quality 3D models, but is computationally expensive. The GPU acceleration techniques inherent to 3D slicer mitigates this drawback somewhat and allows the visualizations to render efficiently, enabling user interactivity. Figure 3 displays a 3D model of an HI emission that has been rendered by SlicerAstro.

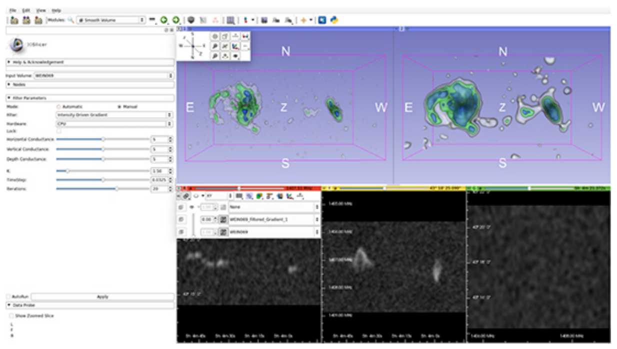


Figure 4: A screenshot of the SlicerAstro interface when visualizing an HI emission [19]

Despite it's impressive 3D visualization capabilities, SlicerAstro and 3D slicer in general struggles to deal with very large data cubes. The entire cube has to be loaded onto the end user's workstation for processing and then needs to be rendered, limiting the size of data cubes that can be processed.

5 COMMON ISSUES IN VISUALIZATION SOFTWARE

While reviewing the existing software tools, we noted some common issues that many of these tools suffer from.

5.1 The *large-than-memory problem*

Most current tools require the entire data cube to be loaded into the main memory before they could perform any processing or visualization of the data. However, the sheer size of modern data cubes can often exceed a typical desktop computer's memory, making this task practically impossible. This problem was dubbed the *larger-than-memory problem* by Hassan et al. [9]. This is a serious issue that occurs even in frequently used packages like the aforementioned DS9 and KARMA.

One way to mitigate this problem is to employ a client-server architecture such as the aforementioned CARTA. This model delegates the memory intensive tasks to a remote server, and allows the visualization itself to be viewed on the end user's browser. This is a promising approach, since it enables very large data cubes to be viewed by even modestly powered workstations.

Another approach to the *larger-than-memory problem* is the use of algorithms that leverage secondary storage such as the hard disk in order to supplement the main system memory [18]. Naturally, the hard disk has a much slower read/write speed than the main memory, so these algorithms need to be highly optimized in order to create a satisfactory user experience.

5.2 Lack of GPU-assisted rendering

Another common issue found in older visualization tools such as DS9 and KARMA is the lack of GPU-assisted rendering. Modern GPU's are increasing in power at a much faster rate than CPU's [15] and their architecture allows for highly parallel algorithms to be run on them. As the volume and complexity of data keeps increasing, tools that don't harness the GPU's power during rendering run a serious risk of being left behind due to severe performance issues.

5.3 Lack of user-friendliness

The lack of user-friendliness is a problem that plagues scientific visualization software in general [21]. Scientific visualization tools are often designed with experts in mind, and can often confuse user's with limited domain knowledge. Employing human computer interaction (HCI) techniques such as participatory design [23] is one of the ways this problem can be mitigated when designing a new tool.

6 DISCUSSION

As we mentioned in the introduction, we are looking for visualization tools that

- Are capable of good 3D visualizations
- Can run on a modestly powered workstation
- Allows for a high level of interactivity

Unfortunately, none of the existing tools seem to fit all of these criteria.

Commonly used tools such as DS9 and KARMA were initially built for 2D visualizations. While current versions of these packages do offer volumetric rendering capabilities, their implementations are dated and do not offer high levels of interactivity. Modern techniques such as GPU accelerated rendering are not employed, resulting in a significant performance hit when dealing with very large data cubes. These older packages also require the entire cube to be loaded into the end user's workstation, severely limiting the size of data cubes they can process.

CARTA offers a promising approach in dealing with the *larger-than-memory problem* with it's client-server model. It also offers a high level of interactivity by leveraging the power of the GPU. Another important aspect of CARTA is the data slicing it performs. It sends only a subset of the data cube to the user, allowing it to efficiently render a view of huge data cubes. Additionally, it employs client-side rendering which enables highly interactive visualizations. However, CARTA does not offer any sort of 3D visualization, making it fall short of our criteria.

SlicerAstro on the other hand has great 3D visualization capabilities, but fails at efficiently dealing with large sets of data due to a lack of the aforementioned client-server approach.

When designing our own tool, it seems quite clear that we need to borrow the client-server approach employed by CARTA. This architecture, coupled it's data slicing capabilities, make it scale incredibly well when dealing with large sets of data. Client-side rendering also seems necessary to enable highly interactive visualizations.

Combining the client-server streaming model of CARTA with the sophisticated 3D visualization of SlicerAstro seems like an interesting angle to explore. It would allow sophisticated 3D visualizations to run smoothly on a modestly powered workstation. When designing a new tool, we would also need to focus on user-friendliness, since this is often a neglected aspect in typical scientific visualization tools.

7 CONCLUSIONS

This paper has reviewed the various techniques and tools used to visualize astronomical data cubes. While many techniques and tools exist, there seems to be a lack of tools that provide good, interactive 3D visualizations of large data cubes. Many of the popular tools are dated and are simply not suitable for the modern era, which points to a growing need for a more modern tool to solve this problem.

Many existing tools have interesting features and architectures that we can gain a lot of insight from. Combining the client-server architecture of CARTA, with the 3D volumetric rendering capabilities of SlicerAstro is an interesting angle to explore if we want to create a suitable visualization tool for the modern era. Additionally, our tool would ideally leverage the power of the client's GPU while performing it's rendering, since this would enable a high level of interactivity.

When designing a new visualization tool, we also need to keep HCI principles in mind in order to make our tool as user-friendly as possible.

REFERENCES

[1] David G Barnes, Christopher J Fluke, Paul D Bourke, and Owen T Parry. An advanced, three-dimensional plotting library for astronomy. *Publications of the Astronomical Society of Australia*, 23(2):82–93, 2006.

[2] Ugo Becciani, Alessandro Costa, V Antonuccio-Delogu, G Caniglia, M Comparato, C Gheller, Z Jin, Mel Krokos, and P Massimino. Visivo–integrated tools and services for large-scale astrophysical visualization. *Publications of the Astronomical Society of the Pacific*, 122(887):119, 2010.

[3] Angus Comrie, Kuo-Song Wang, Pamela Harris, Anthony Moraghan, Shou-Chieh Hsu, Adrianna Pińska, Cheng-Chin Chiang, Hengtai Jan, Rob Simmonds, Tien-Hao Chang, and Ming-Yi Lin. CARTA: The Cube Analysis and Rendering Tool for Astronomy, December 2018.

[4] Robert L Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 137–145, 1984.

[5] Robert A Drebin, Loren Carpenter, and Pat Hanrahan. Volume rendering. *ACM Siggraph Computer Graphics*, 22(4):65–74, 1988.

[6] Christopher J Fluke, David G Barnes, and Amr H Hassan. Visualisation and analysis challenges for wallaby. In *2010 Sixth IEEE International Conference on e-Science Workshops*, pages 15–20. IEEE, 2010.

[7] Richard Gooch. Karma: a visualization test-bed. In *Astronomical Data Analysis Software and Systems V*, volume 101, page 80, 1996.

[8] Marcus D Hanwell, Kenneth M Martin, Aashish Chaudhary, and Lisa S Avila. The visualization toolkit (vtk): Rewriting the rendering code for modern graphics cards. *SoftwareX*, 1:9–12, 2015.

[9] Amr Hassan and Christopher J Fluke. Scientific visualization in astronomy: Towards the petascale astronomy era. *Publications of the Astronomical Society of Australia*, 28(2):150–170, 2011.

[10] Amr H Hassan, Christopher J Fluke, and David Graeme Barnes. Interactive visualization of the largest radioastronomy cubes. *New Astronomy*, 16(2):100–109, 2011.

[11] WA Joye and E Mandel. New features of saoimage ds9. In *Astronomical data analysis software and systems XII*, volume 295, page 489, 2003.

[12] Anton Kaplanyan and Carsten Dachsbacher. Cascaded light propagation volumes for real-time indirect illumination. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, pages 99–107, 2010.

[13] Ron Kikinis, Steve D Pieper, and Kirby G Vosburgh. 3d slicer: a platform for subject-specific image analysis, visualization, and clinical support. In *Intraoperative imaging and image-guided therapy*, pages 277–289. Springer, 2014.

[14] Michiel Kregel et al. *Structure and kinematics of edge-on galaxy disks*. Rijksuniversiteit Groningen, 2003.

[15] Jens Kruger and Rüdiger Westermann. Acceleration techniques for gpu-based volume rendering. In *IEEE Visualization, 2003. VIS 2003.*, pages 287–292. IEEE, 2003.

[16] Ray P Norris. The challenge of astronomical visualisation. In *Astronomical Data Analysis Software and Systems III*, volume 61, page 51, 1994.

[17] JA Ott and Carta Team. Carta: Cube analysis and rendering tool for astronomy. *AAS*, pages 364–11, 2020.

[18] Davide Punzo. *3D visualization and analysis of HI in and around galaxies*. PhD thesis, Rijksuniversiteit Groningen, 2017.

[19] Davide Punzo, JM van der Hulst, JBTM Roerdink, Jean-Christophe Fillion-Robin, and Lingyun Yu. Slicerastro: A 3-d interactive visual analytics tool for hi data. *Astronomy and computing*, 19:45–59, 2017.

[20] Timothy J Purcell, Ian Buck, William R Mark, and Pat Hanrahan. Ray tracing on programmable graphics hardware. In *ACM SIGGRAPH 2005 Courses*, pages 268–es. 2005.

[21] Laurisha Rampersad, Sarah Blyth, Ed Elson, and Michelle M Kuttel. Improving the usability of scientific software with participatory design: a new interface design for radio astronomy visualisation software. In *Proceedings of the South African Institute of Computer Scientists and Information Technologists*, pages 1–9. 2017.

[22] Scott D Roth. Ray casting for modeling solids. *Computer graphics and image processing*, 18(2):109–144, 1982.

[23] Douglas Schuler and Aki Namioka. *Participatory design: Principles and practices*. CRC Press, 1993.

[24] Melanie Tory. Mental registration of 2d and 3d visualizations (an empirical study). In *IEEE Visualization, 2003. VIS 2003.*, pages 371–378. IEEE, 2003.

[25] Donald Carson Wells and Eric W Greisen. Fits-a flexible image transport system. In *Image Processing in Astronomy*, page 445, 1979.