Interactive VR and Desktop Environment for HI Data Cube Exploration – A First Step

David Bernhardt, Anton Kindsvater
Tom David Eibich, André Hinkenjann

Publisher: Dean Prof. Dr. Sascha Alda

Hochschule Bonn-Rhein-Sieg – University of Applied Sciences,
Department of Computer Science

Sankt Augustin, Germany

Dezember 2025

Technical Report 02-2025

Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

ISSN 1869-5272

ISBN 978-3-96043-124-4

Copyright © 2025, by the author(s). All rights reserved. Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

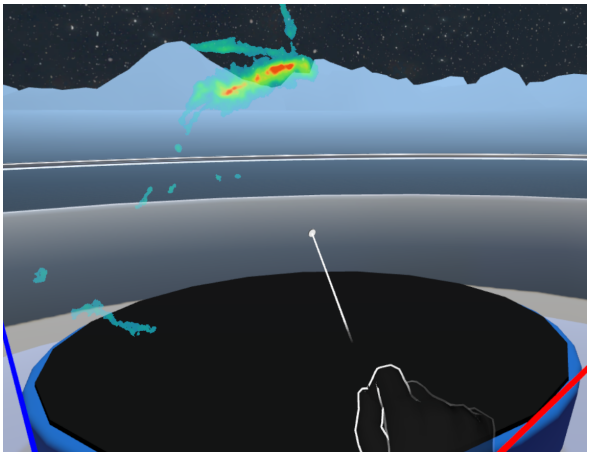
Das Urheberrecht des Autors bzw. der Autoren ist unveräußerlich. Das Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Das Werk kann innerhalb der engen Grenzen des Urheberrechtsgesetzes (UrhG), *German copyright law*, genutzt werden. Jede weitergehende Nutzung regelt obiger englischsprachiger Copyright-Vermerk. Die Nutzung des Werkes außerhalb des UrhG und des obigen Copyright-Vermerks ist unzulässig und strafbar.

Digital Object Identifier https://doi.org/10.18418/978-3-96043-124-4

Interactive VR and Desktop Environment for HI Data Cube Exploration - A First Step

David Bernhardt, Anton Kindsvater, Tom David Eibich, and André Hinkenjann

Institute of Visual Computing, University of Applied Sciences Bonn-Rhein-Sieg, Sankt Augustin, Germany
https://www.h-brs.de/de/ivc



Abstract. We present an immersive multi-user virtual reality (VR) system aimed at enabling collaborative exploration of large-scale neutral hydrogen (HI) spectral-line data cubes in radio astronomy. By integrating an external native rendering plugin with Unity and a partially GPU-accelerated source finding pipeline based on SoFiA-2, our system facilitates interactive filtering and visualization of radio data cubes beyond traditional size constraints. While real-time filtering remains limited to a standalone desktop variant, the VR system allows users to inspect pre-filtered data in high fidelity jointly. Our work addresses key challenges in scaling volume rendering for large data sets, reducing filtering latency through GPU acceleration, and enabling co-located scientific discussion in virtual space. We outline the system architecture, describe the GPU porting efforts of SoFiA, and position our work in the context of current astronomical visualization tools and workflows.

Keywords: VR · AR · Radio Astronomy · Ray Tracing · GPU · Volume Rendering · Scientific Visualization.

1 Introduction

Neutral hydrogen (HI) plays a central role in the study of galaxy evolution, the cosmic web, and large-scale structure. Through its characteristic emission at the 21cm line (1420MHz) [4,19,14,6], astronomers can trace the distribution and kinematics of gas in galaxies over cosmological timescales. Mapping this emission yields three-dimensional data products, so-called HI spectral-line cubes, with two spatial axes and one spectral axis. These cubes can grow exceptionally large, particularly with next-generation radio interferometers such as ASKAP, MeerKAT, and the Square Kilometre Array (SKA) [3,15,16].

For example, a single observation by the Australian SKA Pathfinder (ASKAP) as part of the WALLABY survey can result in data cubes exceeding 2 TB in size [17]. Throughout an observing campaign, many such cubes are generated, leading to total data volumes in the petabyte regime. These cubes are typically stored in FITS format, the de facto standard in astronomy, and include not only spectral data but also extensive metadata about coordinate systems and observational parameters [36].

Practical scientific use of this data requires robust pipelines for noise filtering, automated source detection, and volume rendering for visualisation. Traditional desktop workflows strain under the weight of data at this scale, particularly when immersive exploration and collaborative analysis are desired. Virtual reality (VR) offers spatial immersion and co-presence but poses technical challenges when integrating with large datasets. Popular engines like Unity enforce per-object GPU memory caps (e.g., 2GB), often requiring substantial downsampling that obscures subtle HI features [5].

Our work addresses these challenges by coupling GPU-accelerated source finding with native high-resolution rendering within a distributed, multi-user VR environment. While real-time filtering in VR remains an ongoing process, our system already demonstrates interactive, high-fidelity visualisation of pre-filtered HI cubes, with synchronised multi-user support and latency-minimised data pipelines.

2 Related Work

2.1 Telescope Arrays and HI Data Cubes

Modern radio astronomy surveys rely on large interferometric telescope arrays to detect faint neutral hydrogen signals across vast areas of the sky. Instruments like the Australian SKA Pathfinder ASKAP, MeerKAT in South Africa, and the upcoming SKA1-MID and SKA1-LOW arrays consist of multiple dishes whose signals are combined to simulate a much larger effective aperture, achieving high angular resolution and sensitivity [16,3,15]. These facilities are designed for large-scale, blind surveys of the HI sky, producing data suitable for both cosmological and galactic-scale studies.

The fundamental observable is the 21cm hyperfine transition of HI, emitted when the spins of the proton and electron in a hydrogen atom flip from parallel

to antiparallel alignment. This emission line corresponds to a rest frequency of 1420.4MHz [4,19,14,6]. As this radiation travels through the expanding universe, its frequency is redshifted in proportion to the source's velocity away from the observer. The degree of redshift (or blueshift) thus encodes information about the line-of-sight velocity of the emitting gas. For distant galaxies, this is interpreted as a proxy for cosmological distance via Hubble's law [13,8,2,12]. Within galaxies, differential redshift due to rotation reveals internal kinematics, such as rotation curves and disk warps [1].

Radio interferometers measure the brightness temperature of the sky as a function of both position and frequency. The resulting data cube has two spatial axes (right ascension and declination) and one spectral axis (corresponding to frequency or radial velocity). Each voxel in the cube contains an intensity value, usually given in units of Jy/beam, representing the flux density at a given spatial position and frequency [33]. In practical terms, this means that astronomers can reconstruct both where HI is located on the sky and how fast it is moving relative to Earth. Extended structures such as tidal tails or gas bridges may span multiple velocity channels, while rotating disks exhibit systematic gradients across the frequency axis. These characteristics make spectral-line cubes indispensable for understanding galaxy dynamics and environment.

The volume and complexity of this data pose significant challenges for storage, analysis, and visualization. Instruments like ASKAP generate dozens of such cubes per night of observation, requiring scalable tools for detection, inspection, and cataloging of sources [17].

2.2 Current Visualization Tools

There exist many different visualization tools for viewing HI data cubes. Here we list a selection of tools able to visualize these cubes in 3D space and iterate over their abilities and drawbacks.

SlicerAstro extends the medical imaging platform 3DSlicer to support astronomical FITS cubes. It provides interactive linked 2D/3D views, masking, filtering, and model overlays (e.g., via 3DBarolo) for rotation curve analysis. SlicerAstro supports loading of full-resolution data and allows interactive inspection of complex galaxy morphologies such as warps or tidal features [28]. However, it is single-user and desktop-bound, and performance degrades in low signal-to-noise scenarios without prior masking or preprocessing.

iDaVIE-v, developed using Unity, enables immersive inspection of HI cubes with basic editing and visualisation features on head-mounted displays. Due to Unity's 2GB per-asset GPU buffer cap, large FITS cubes must be downsampled, reducing spatial and spectral resolution [5]. Moreover, it does not support dynamic source filtering; instead, data must be preprocessed offline.

Encube is a visual analytics framework designed for large-format tiled display walls and CAVE environments. It supports collaborative inspection of FITS

cubes and comparative viewing across datasets [11]. However, it relies on expensive, fixed infrastructure and does not support lightweight VR headsets or dynamic data filtering workflows.

2.3 High-Performance Volume Rendering

High-performance distributed rendering systems have been explored for HI data visualisation. For instance, Hassan et al. implemented a real-time distributed GPU volume rendering system that handled 204GB cubes at 30 frames per second using 128 GPUs [10]. Earlier work by the same group achieved interactive rates for HIPASS and GASS survey cubes ranging from 12–26GB in size [9]. These systems prioritise rendering throughput and data partitioning but lack support for immersive VR, co-located collaboration, or integrated source finding.

2.4 Source Finding Pipelines

SoFiA-2 (Source Finding Application 2) is a widely used modular pipeline developed for HI surveys such as WALLABY. It operates on FITS cubes and includes key steps such as noise normalisation, continuum subtraction, and the core smooth-and-clip (S+C) algorithm. The S+C algorithm performs spatial smoothing using multi-scale Gaussian kernels and spectral smoothing via boxcar filters, followed by threshold clipping. This enhances faint, extended emission and suppresses background noise [29,38].

Detected regions are grouped via connected-component labelling and assigned reliability scores based on a positive/negative voxel comparison. Red- and blue-shifted emission, due to galaxy rotation or cosmological redshift, is treated as signal when it coherently matches the expected line profile structure. Output products include source catalogues, mask cubes, and moment maps for further analysis. Although optimised for batch processing, SoFiA-2 is CPU-bound and not well suited for iterative workflows. Parallel variants like SoFiA-X exist, but even these require hours to process large data cubes [37].

Recent work has begun porting parts of SoFiA to the GPU to accelerate key components such as the S+C finder. In our system, we integrate such GPU-based acceleration to reduce latency by over an order of magnitude, enabling the potential for near real-time feedback in desktop environments and fast data preparation for VR rendering.

3 System Overview

To cover the defined use case, we identified several main components that the system must have. Between those components exists a clearly defined architectural border. The components are as follows: Server-side components for Data storage and distribution via network, data processing by using a source finding algorithm, and finally, the client-side frontend component for VR/AR using a Unity-based application, and the standalone application for desktop users, both

of which share the same component for visualization of HI-Datacubes. Figure 1 shows the architectural diagram and the dependencies between them. The users can access the same data independently of the client application. The server can also run on a client device. Still, the shared memory resource must be considered, especially when the user requests a data processing query on the server. This can lead to frame freezes because the source finding algorithm also runs on the GPU. It might be memory hungry. It strongly depends on the underlying dataset.

Server-related components are responsible for storage and processing. We were against splitting them further and distributing the components to multiple server devices for simplicity. Additionally, the data processing requires rapid access to the data set; otherwise, the distributed solution would necessitate implementing a data caching system to minimize processing request latency for frequently accessed data. For interaction, we made multiple REST API entry points accessible for the frontend applications. As mentioned earlier, we use a FITS file as a source for the volumetric dataset. And each data manipulation request produces, on its success, a new FITS mask file and a filtered FITS data set. The visualization is further transformed into a NanoVDB format. The NanoVDB format is annotated with a unique ID, and the ID to file mapping is stored in a simple data catalogue Database as a JSON file. The user application can use the unique ID to target specific processed data sets.

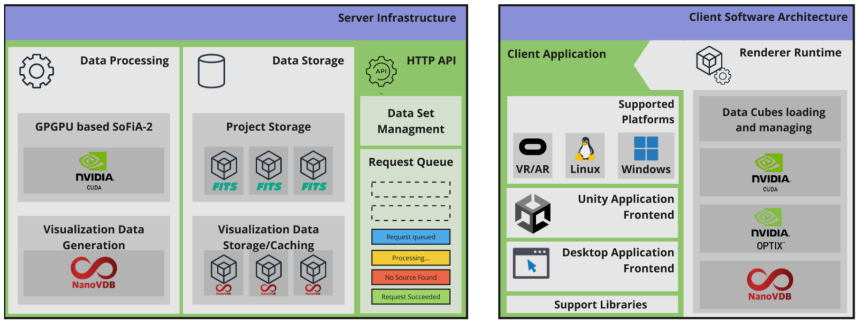


Fig. 1. Simplified component overview. The left part of an image represents the server-side technology stack and shows the on-server request management concept. The right site represents different client applications and their technology stacks.

3.1 Data Management and Processing

Data Storage and Distribution is the simplest component we have, its only purpose is to make the data set and its processed versions available for download over REST API via a unique identifier. Data processing is divided into separate steps. The server receives the source search request and queues a SoFiA data

processing process. If the SoFiA search is completed successfully, the mask FITS file and the source FITS file are passed to the NanoVDB generator. Data is extracted from the source using the mask data and fitted to the NanoVDB Grid builder. The resulting NanoVDB volume is dumped to persistent storage with additional metadata, and the data gets associated with the ongoing request. Finally, the user receives an acknowledgement response, and the user application can react to it by fetching the data via a download request, for example.

3.2 Visualization Component

This component is only responsible for the rendering. By design, it should be implemented as a shared library and provide an easy integration interface for use in a standalone desktop application and the Unity project. For interoperability between Unity's C# run-time, we exported all necessary interfaces as C language functions in a separate layer. So, the design can be described by the layered architecture. The rendering frequency should fit the VR Device requirements of 90 to 120 Hertz. As a starting point, we tested multiple approaches to scale the data for a consistent time frame. The approaches included scaling the volume data, resampling the spatial volume space, or reducing the amount of data. We focus on data reduction for efficient data traversal while rendering. This approach is not a general solution, as it may not be sufficient in some use-case scenarios where it is also essential to consider the surrounding noise in the near field of the found sources. However, we focused on a special case and decided to remove the noise as much as possible in search of a fast iteration over the dataset for the user. Technically, we implemented a direct volume rendering and sampled the volume data from the in-GPU memory stored in NanoVDB's data structure, as mentioned in the Data Processing 3. While integrating along the ray, we resample according to the normalisation parameters and apply the transfer function for colour mapping. Internally, we use multiple integration methods, but for the coverage of the use case, we needed to have only the maximum projection. For the ray traversal, the HDDA algorithm is used for efficient hierarchical space skipping [20].

As one of the renderers, the client application runs on the VR device. We also investigated foveated rendering to scale further rendering performance. We tested and implemented the techniques mentioned in the paper [18]. Since our tested data were stable at native resolution, foveated rendering is not enabled by default in the current version of the software. This is also a topic where further work can be extended; for example, we can precisely use VR's gaze sensor to control foveated resolution.

3.3 Collaborative AR/VR Client

The collaborative Virtual Reality (VR) client is the key interface for immersive exploration of filtered HI data cubes in a multi-user setting. Built in Unity and deployed on PC-tethered Meta Quest 3 HMDs, the system supports natural data

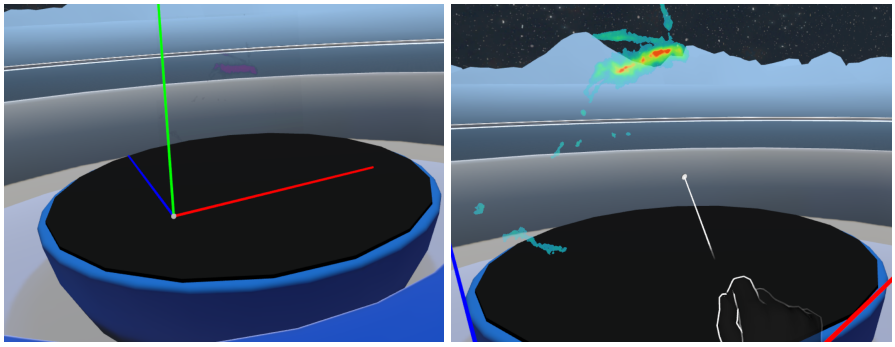


Fig. 2. The left picture shows a data cube at resting position with an unaltered transfer function. On the right side, user interaction can be observed with the user pulling the cube closer with a pinching gesture, and with an altered transfer function for better visibility.

interaction and spatial reasoning.

Interaction Design and Transfer Function Control
Using Meta Quest hand tracking, users can directly manipulate the data cube: grabbing it with one hand enables free rotation and repositioning, while holding it with both hands allows scaling (see Fig. 2). This gesture-based interface aligns with natural spatial interaction behaviors; empirical research shows that users prefer hand-based interactions that reflect real-world manipulation - particularly one-handed gestures, with multi-handed control relaxed or alternated for comfort and intuitiveness [22].

Visualization is fine-tuned via a transfer function editor rendered as a grabbable 2D UI window in VR (see Fig. 3). Users adjust the intensity mapping by drawing directly within VR, with changes synchronized across all participants, ensuring consistent visualization for the collaborative group.

Collaboration Model and User Representation
When a session begins, the first connecting client becomes the host and has control over which cube is displayed and how it is positioned. Control can be dynamically shared: participants take authority by interacting—only one user may hold control at a time to prevent conflict. A floating avatar head represents each user in the scene with accurate gaze direction, enabling participants to infer where others are focusing and support collaborative exploration.

Rendering Pipeline and Performance Considerations
High-resolution volume rendering is executed outside of Unity using a custom native plugin, which interfaces with Unity via shared GPU memory and render textures. This approach was essential to bypass Unity's internal limits on large asset sizes. While Unity supports 3D textures with dimensions up to $2048 \times 2048 \times 2048$

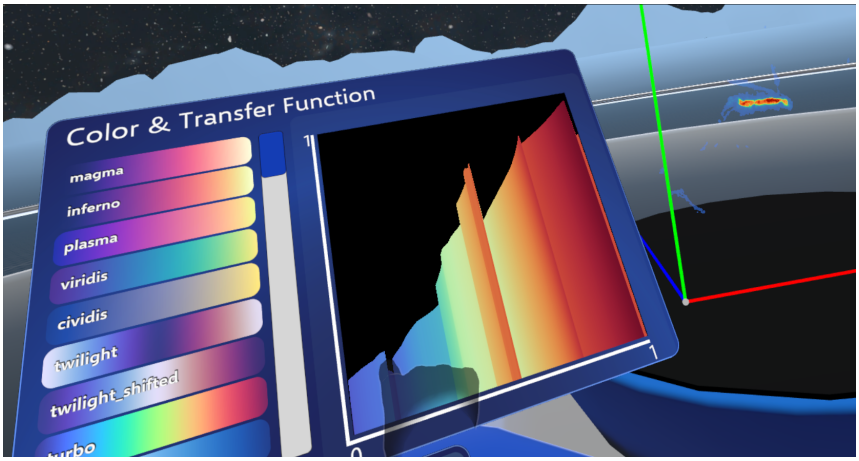


Fig. 3. VR Interface for altering the transfer function. The user may "paint" a new transfer function using a finger to assign color values to certain intensities. On the left side, a color spectrum can be selected to accommodate user preferences regarding color schemes. On the right side, the impact on the data cube can be observed.

voxels [34], the actual memory allocation limit per texture is not officially defined. However, a practical limit of around 2GB per texture is widely recognized among developers, with larger textures frequently causing load failures or crashes. This limitation becomes critical when working with full-resolution astronomical data cubes, which can easily exceed this size threshold when using high-precision formats like floating-point.

By outsourcing rendering to a dedicated native module, our system avoids Unity's memory restrictions and supports real-time visualization of large, filtered data cubes in their entirety.

The rendered volume is streamed into Unity as a render texture. Because of the inter-process communication required and the complexity of synchronizing rendering outside of Unity's rendering loop, this process introduces a constant latency of approximately 100ms between user interaction (e.g., movement) and the cube's visual response. While this delay is perceptible during rapid motion, it remains acceptable for typical slow-paced inspection of the data.

Technical Platform and Constraints

The VR client requires a PC with a CUDA-capable GPU to run required workloads via OptiX [24] with hardware raytracing support and CUDA [26], and is not designed for standalone execution on a Meta Quest 3, due to the high computational and memory requirements of the external renderer. This constraint limits the application's portability but enables full-resolution rendering that aligns with the scientific needs of radio astronomers.

Future developments may include distributed synchronization of parameter changes and real-time source finding feedback within the VR environment, enabling an even more integrated collaborative workflow.

3.4 Desktop Client

The desktop client is a full-featured tool for single-user data exploration, built around the data analysis workflow and utilizing almost all the previously mentioned components of the system. Additionally, the tool was used in development as a benchmark baseline for the renderer. The software architecture is designed flat to facilitate quick integration of testing and benchmarking tools. Initially, the application primarily served as a glue and test sandbox during development. Figure 4 shows the work in progress of the desktop application UI.

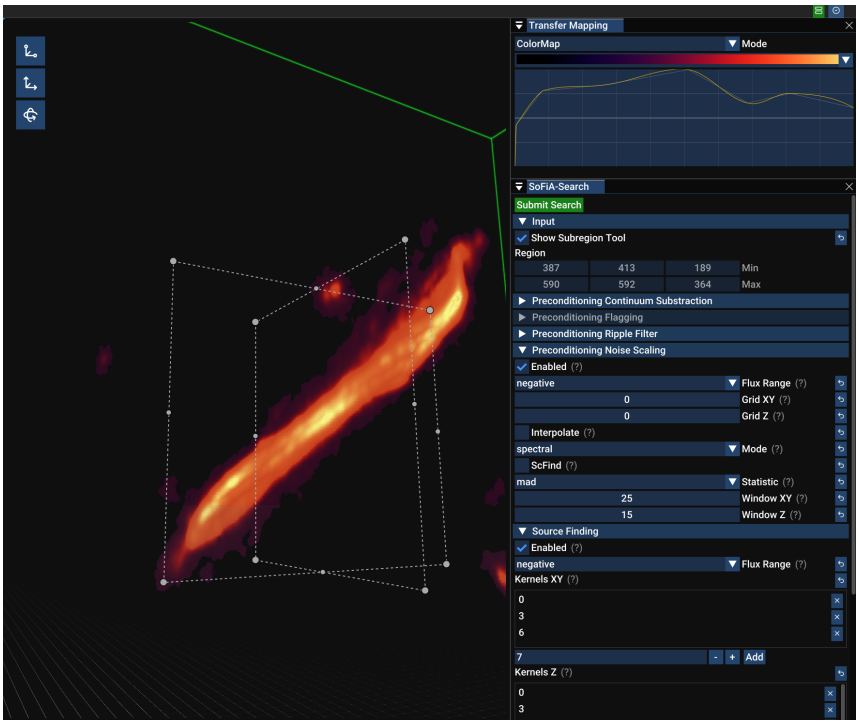


Fig. 4. View of the desktop client. To the left, a selected source inside a data cube is visible. To the right, different settings related to data visualization, like a changeable transfer function, are shown.

3.5 SoFiA GPU Acceleration

To enable efficient source detection in large HI data cubes, the system integrates a GPU-accelerated version of the Smooth and Clip (S+C) finder from SoFiA-2 [38]. The most computationally intensive part of the pipeline—smoothing the data cube in spectral and spatial dimensions—has been parallelized using NVIDIA CUDA [26]. Spectral (z-axis) and one spatial (x-axis) filtering are performed in a combined step, followed by a second spatial filter in the y-direction. Thresholding of the smoothed result is also parallelized on the GPU.

SoFiA is run as a standalone process on a dedicated server and can be triggered on demand by external applications. While the system architecture allows both the VR and the desktop client to initiate a filtering request, this functionality is currently only exposed in the desktop version.

4 Implementation

4.1 Renderer

The renderer was designed with a data-driven approach, recognizing two fundamentally distinct types of datasets encountered in our workflow. The first type consists of highly noisy data, whereas the second, typically obtained after filtering, contains mostly empty space interspersed with compact, high-intensity sources. Our primary focus was on optimizing rendering for the latter, as the sparsity of data provides opportunities for significant computational efficiency gains.

For rapid prototyping, we adopted the OptiX Wrapper Library (OWL) [25], which builds upon NVIDIA's CUDA and OptiX frameworks. This choice allowed us to leverage hardware-accelerated ray tracing capabilities available on modern GPUs, while enabling flexible experimentation with different rendering strategies.

Our initial implementation served as a baseline and consisted of a naive direct volume renderer. The 3D volume data was stored in GPU-accessible memory, and for each pixel on the screen, a ray was generated. Rays were traversed through the volume using a digital differential analyzer (DDA), stepping sequentially through each voxel and integrating intensity values along the ray path.

In a subsequent iteration, we introduced spatial acceleration by identifying axis-aligned bounding boxes (AABBs) corresponding to regions containing data after filtering. Rays were then traversed independently within each AABB, allowing large empty regions to be skipped entirely. This approach yielded notable performance improvements.

In the final prototype iteration, we replaced the custom empty-space skipping mechanism and naive voxel storage with the NanoVDB [23] sparse volumetric data structure. Ray traversal was updated to use the Hierarchical Digital Differential Analyzer (HDDA) [21], enabling more efficient hierarchical space skipping during rendering.

The renderer also supports user-defined transfer functions, allowing flexible mapping from data intensities to visual attributes. Transfer functions are implemented as 1D textures uploaded to GPU memory for efficient sampling during ray traversal. Intensity integration along the ray can be performed in three modes: maximum intensity projection, average intensity accumulation, or translucent material blending. At the same time, alternative modes can provide visually appealing representations. Maximum intensity projection was selected as the default due to its suitability for accurately visualizing compact astronomical sources in our scientific use case.

4.2 SoFiA

To accelerate the Smooth and Clip (S+C) finder in SoFiA-2, we developed a custom GPU implementation using the NVIDIA CUDA toolkit [26]. This version replaces the CPU-based smoothing pipeline with optimized CUDA kernels that exploit the massive parallelism of modern GPUs.

The original filtering process in SoFiA involves three sequential operations: a Boxcar filter along the spectral axis (z), and Gaussian filters along the two spatial axes (x and y). In our GPU version, we optimize this by combining the z and x filters (see Fig. 5) into a single kernel pass, followed by a second pass for the y-direction Gaussian smoothing (see Fig. 6). The Gaussian filters are approximated by applying a Boxcar filter multiple times, which is well-suited for parallel execution.

Each filter kernel is implemented using 2D CUDA blocks, with memory access patterns carefully optimized for performance. In the x and z directions, cube rows are stored in shared memory in a format where threads within the same warp access consecutive memory locations. This allows for coalesced memory access and minimizes latency. For y-direction smoothing, vertical columns of the cube are reordered as they are written to shared memory so that memory access within warps remains sequential despite the original layout of the cube.

All filter operations operate on shared memory buffers, reducing the need for frequent global memory access and improving performance significantly. The original unsmoothed data is preserved in a separate buffer, as it is reused between filtering passes. This duplication results in an overall GPU memory requirement of roughly twice the cube size.

After filtering, the thresholding step is applied directly on the GPU. Instead of using a full-size floating-point mask, we store the binary source mask as a compact bitmask representation. One byte holds the mask status for eight voxels, which dramatically reduces memory usage and allows the entire pipeline to run in-GPU for cubes of up to approximately 10GB in size (depending on available memory).

All filtering and masking steps are fully executed on the GPU before results are returned to the CPU, minimizing data transfer overhead. This design provides dramatic performance improvements up to 36× faster than the original CPU parallel implementation, while producing equivalent results (within 0.1% deviation).

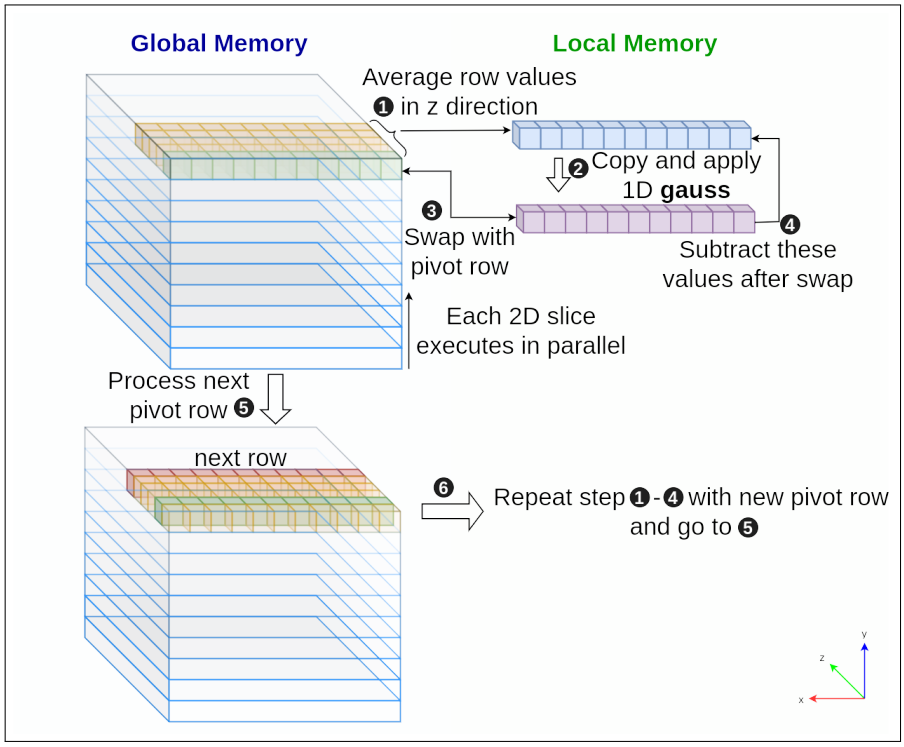


Fig. 5. Application of Z- and X-filtering. All rows in spatial direction contributing to the filtering of pivot row (green) are read from global memory (GM) and averaged in z-direction (1). 1D-gauss is applied to result (2). Result (purple) is swapped with the original pivot row in GM (3). The original data is subtracted from the averaged (blue) row (4). The next row (red) is added to the averaged row (5). Steps get repeated for every row as the pivot row.

The GPU acceleration is integrated as a backend module that can be invoked on demand. It operates independently of how the filtering request is triggered, whether from a desktop application or future VR interfaces. Currently, only the desktop version supports requesting filter operations in this way, but the system design would equally support interactive filtering via VR.

5 Discussion

This work explores how immersive and real-time technologies can reshape traditional radio astronomy workflows - particularly those involving the inspection and classification of HI sources in 3D spectral data cubes. In contrast to conventional 2D tools such as KARMA, or even more advanced 3D desktop applications like SlicerAstro, our system aims to unify source finding and volumetric visu-

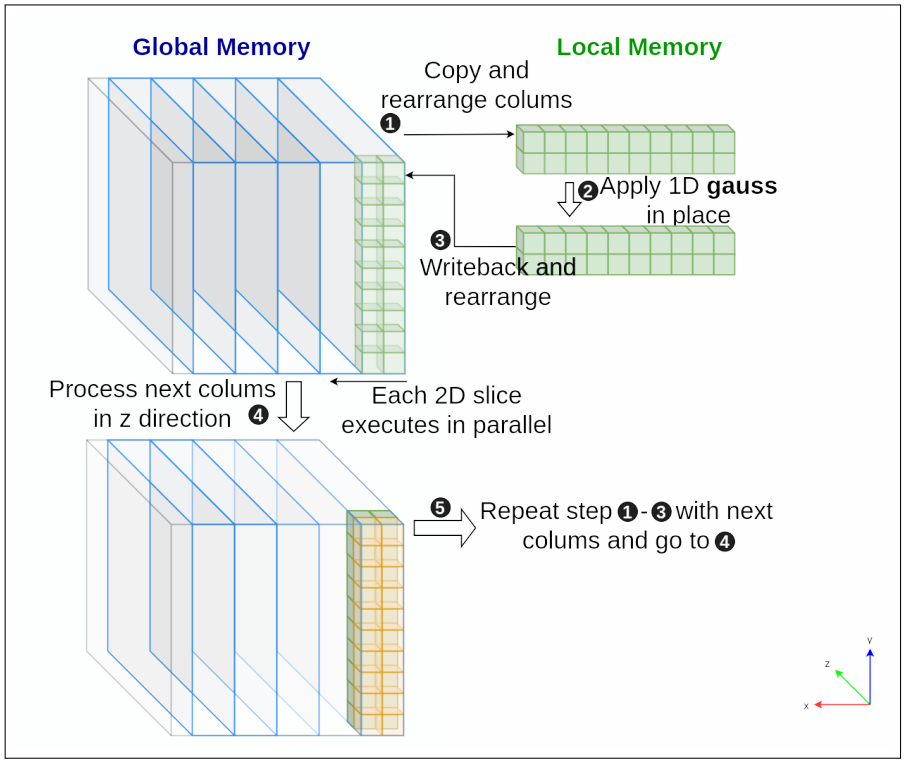


Fig. 6. Application of Y-filtering. Multiple columns of the same spatial slice are copied and rearranged into local memory (1). Rearranging is done so that pixels that are directly adjacent in the y-direction are neighbors in memory. A 1-D Gaussian filter is applied to each row (2). The result is written back to global memory to the original memory locations (3). The following columns in the z-direction are processed in the same way (4).

alization in both interactive desktop and immersive VR environments. This is a key differentiator compared to related tools such as iDaVIE-v and Encube, which emphasize collaborative and immersive data viewing but rely on external pipelines for source extraction.

Unlike those systems, our platform supports direct invocation of the SoFiA source finder from within the visualization environment, and thanks to GPU acceleration, the response time is reduced from minutes to seconds even for large data cubes. This enables a novel kind of iterative exploration, where users can repeatedly tweak filtering thresholds and immediately see the impact in 3D. This loop of "filter → visualize → refine" is impractical in traditional workflows, where filtering is slow and visualization is detached from source extraction.

User feedback from radio astronomers supports these advantages. They particularly appreciated the ability to explore the whole 3D structure of a data

cube, instead of flipping through spectral slices or analyzing voxel tables. This perspective was reported to enhance intuition about ambiguous features and assist in identifying potential source candidates more confidently. The possibility of invoking SoFiA multiple times in rapid succession was described as a powerful enhancement to their existing tools.

However, while immersive VR visualization was seen as valuable, users also pointed out ergonomic limitations of current-generation VR hardware. Headsets were considered too heavy and thermally uncomfortable for prolonged use, leading most users to prefer the desktop version. In this sense, our system straddles the boundary between future-facing immersive workflows and practical, real-world usage. Multi-user VR sessions are already supported, offering promising directions for remote collaborative analysis - especially as lighter, AR-based headsets emerge in the coming years.

6 Further Work

Several promising directions have emerged for extending the system beyond its current capabilities. One avenue lies in further decoupling the renderer from the client applications by adopting the ANARI rendering standard [31,30,7,32,35]. ANARI provides an abstraction layer that allows different rendering backends to be integrated with minimal changes to application code. This would make it possible to replace our current NanoVDB-based renderer with alternative, ANARI-compatible volume renderers, enabling flexibility in choosing rendering technology for different hardware environments. While this integration was not pursued in the current work due to resource constraints, the standard's active development — including improvements to volume rendering support - suggests that it could become a practical and powerful option in the near future.

Another area for future development is the integration of the GPU-accelerated SoFiA backend directly into the mainline SoFiA codebase. Currently, the acceleration is limited to filtering and masking steps, with linking still handled by the CPU. Extending GPU acceleration to the linking stage would further shorten the iterative analysis cycle, making real-time, in-application filtering more practical for even larger data cubes.

From a user experience perspective, VR ergonomics and accessibility remain central challenges. Although the immersive mode provides unique benefits for spatial reasoning and collaborative exploration, current headsets are often too heavy and warm for prolonged professional use. Future iterations of the system could target lightweight AR glasses or mixed-reality devices, which would allow scientists to interact with volumetric data in their normal working environment without the physical discomfort associated with long VR sessions.

Finally, the collaborative features of the VR client could be extended to the desktop application, enabling mixed-mode sessions where some participants are immersed in VR while others join from traditional workstations. This would support broader adoption and ensure that collaborative analysis is not restricted by hardware availability.

7 Conclusion

We have presented a hybrid system for 3D HI source analysis that tightly integrates GPU-accelerated filtering with immersive and desktop volumetric visualization. Compared to prior tools like SlicerAstro, iDaVIE-v, and Encube, our system is distinguished by its ability to execute filtering operations within the visualization environment itself, in near-real-time. Our system supports a highly interactive, exploratory workflow that blends data processing with spatial intuition.

Our findings suggest that 3D visualization, whether on a desktop or in VR, significantly enhances scientists' ability to evaluate and detect HI sources. While VR-specific benefits such as spatial awareness and collaborative exploration were acknowledged, current hardware limitations (e.g., headset weight and comfort) still present a barrier to adoption. Nonetheless, the potential for VR to evolve into a core component of scientific analysis is clear.

As the hardware matures, we anticipate that immersive environments will increasingly become viable tools for both single-user exploration and distributed collaboration in astronomy. Future work will continue to enhance the GPU backend, particularly by porting SoFiA's linking stage and exploring deeper integration of collaborative features, both in VR and desktop environments.

8 Acknowledges

We want to thank Dr. Peter Kamphuis who helped us iterate over different parts of the system, giving feedback on what tools and functions would be needed in radio astronomy data analysis.

We also want to thank the research grant given by the Ministry of Culture and Science of the German state of North Rhine-Westphalia, that made this research possible in the first place. As such, this research was part of the larger research project B3D (PROFILNRW-2020-038C).

Part of the research was developed, open-sourced, and published under MIT Licence on GitHub [27].

References

1. Begeman, K.G.: Hi rotation curves of spiral galaxies. i—ngc 3198. Astronomy and Astrophysics **223**, 47–60 (1989)
2. Davis, T.M., Lineweaver, C.H.: Superluminal Recession Velocities. arXiv preprint astro-ph/0011070 (2000)
3. Dewdney, P.E., Hall, P.J., Schilizzi, R.T., Lazio, T.J.L.W.: The Square Kilometre Array. Proceedings of the IEEE **97**(8), 1482–1496 (2009). https://doi.org/10.1109/JPROC.2009.2021005
4. Ewen, H.I., Purcell, E.M.: Observation of a Line in the Galactic Radio Spectrum: Radiation from Galactic Hydrogen at 1,420 Mc./sec. Nature **168**, 356–358 (1951)

5. Fluke, C.J., Hassan, A.H., Lahert, C.A., Barnes, D.G.: iDaVIE-v: Immersive Data Visualisation in Virtual Environments for HI Surveys. Astronomy and Computing **30**, 100335 (2020). https://doi.org/10.1016/j.ascom.2019.100335
6. Furlanetto, S.R., Oh, S.P., Briggs, F.H.: Cosmology at low frequencies: The 21 cm transition and the high-redshift universe. Physics Reports **433**(4–6), 181–301 (2006). https://doi.org/10.1016/j.physrep.2006.08.002
7. Group, K.: ANARI - Cross-Platform 3D Rendering Engine API (Feb 2020), https://www.khronos.org/anari/, section: API
8. Harrison, E.R.: The Redshift–Distance and Velocity–Distance Laws. The Astrophysical Journal **403**, 28–34 (1993). https://doi.org/10.1086/172179
9. Hassan, A.H., Fluke, C.J., Barnes, D.G.: Interactive visualization of the largest radioastronomy cubes. In: ASP Conference Series. vol. 442, p. 97 (2011)
10. Hassan, A.H., Fluke, C.J., Barnes, D.G.: A Distributed GPU-based Framework for real-time 3D Volume Rendering of Large Astronomical Data Cubes. New Astronomy **17**(5), 371–381 (2012). https://doi.org/10.1016/j.newast.2011.11.004
11. Hassan, A.H., Fluke, C.J., Barnes, D.G.: Encube: Exploiting the CAVE2 for Collaborative Visual Analytics of Spectral-Cube Surveys. Publications of the Astronomical Society of Australia **30**, e059 (2013). https://doi.org/10.1017/pas.2013.004
12. Hogg, D.W.: Distance measures in cosmology. arXiv preprint astro-ph/9905116 (1999)
13. Hubble, E.: A relation between distance and radial velocity among extra-galactic nebulae. Proceedings of the National Academy of Sciences of the United States of America **15**(3), 168–173 (1929). https://doi.org/10.1073/pnas.15.3.168
14. van de Hulst, H.C.: Theoretical Prediction of the 21-cm Line of Neutral Hydrogen. Bulletin of the Astronomical Institutes of the Netherlands **10**, 279 (1944)
15. Johnston, S., Bailes, M., Bartel, N., et al.: Science with ASKAP: The Australian Square Kilometre Array Pathfinder. Experimental Astronomy **22**(3), 151–273 (2008). https://doi.org/10.1007/s10686-008-9124-7
16. Jonas, J., Team, M.: The meerkat radio telescope. p. 1 (2016). https://doi.org/10.22323/1.277.0001
17. Koribalski, B.S., et al.: WALLABY Early Science. Publications of the Astronomical Society of Australia **37**, e034 (2020). https://doi.org/10.1017/pasa.2020.26
18. Meng, X., Du, R., Zwicker, M., Varshney, A.: Kernel Foveated Rendering. Proc. ACM Comput. Graph. Interact. Tech. **1**(1) (Jul 2018). https://doi.org/10.1145/3203199, https://doi.org/10.1145/3203199
19. Muller, C.A., Oort, J.H.: The Interstellar Hydrogen Line at 1,420 Mc./sec., and an Estimate of Galactic Rotation. Nature **168**, 357–359 (1951)
20. Museth, K.: Hierarchical digital differential analyzer for efficient ray-marching in OpenVDB. In: ACM SIGGRAPH 2014 Talks. p. 1. SIGGRAPH '14, Association for Computing Machinery, New York, NY, USA (Jul 2014). https://doi.org/10.1145/2614106.2614136, https://dl.acm.org/doi/10.1145/2614106.2614136
21. Museth, K.: Hierarchical digital differential analyzer for efficient ray-marching in openvdb. In: ACM SIGGRAPH 2014 Talks, pp. 1–1 (2014)
22. Nanjappan, V., Liang, H.N., Lu, F., Papangelis, K., Yue, Y., Man, K.L.: User-elicited dual-hand interactions for manipulating 3D objects in virtual reality environments. Human-centric Computing and Information Sciences **8**(1), 1–16 (2018)
23. NVIDIA: NanoVDB: A GPU-Friendly and Portable VDB Data Structure For Real-Time Rendering And Simulation | ACM SIGGRAPH 2021 Talks, https://dl.acm.org/doi/abs/10.1145/3450623.3464653

24. NVIDIA: NVIDIA OptiX™ Ray Tracing Engine, https://developer.nvidia.com/rtx/ray-tracing/optix
25. NVIDIA: OWL - A Productivity Library for OptiX, https://owl-project.github.io/
26. NVIDIA Corporation: Cuda toolkit documentation. https://docs.nvidia.com/cuda/ (2024), accessed: 2025-07-11
27. B3D Visualization Project. https://github.com/Institute-of-Visual-Computing/b3d-vis
28. Punzo, D., Oonk, J.B.H., van der Hulst, J.M., van Diepen, G., Alexov, A., Begeman, A., Kleijn, G.V.: SlicerAstro: a 3-D interactive visual analytics tool for HI data. Astronomy & Astrophysics **603**, A75 (2017). https://doi.org/10.1051/0004-6361/201629594
29. Serra, P., Westmeier, T., Koribalski, B.S., Flöer, L., Moss, V.A., Meyer, M.D., Pisano, D.J., Curran, S.S.: Sofia: A flexible source finding pipeline for 3d spectral line data. Monthly Notices of the Royal Astronomical Society **448**(2), 1922–1936 (2015). https://doi.org/10.1093/mnras/stv044
30. The Khronos Group: Khronos launches anari 1.0 as industry's first cross-platform 3d rendering api. https://www.khronos.org/news/press/khronos-launches-anari-1.0-as-the-industrys-first-cross-platform-3d-rendering-engine-open-standard-api (2023), accessed: 2025-08-08
31. The Khronos Group: Anari: Cross-platform 3d rendering engine api. https://www.khronos.org/anari/ (2025), accessed: 2025-08-08
32. The Khronos Group: VisIt Embraces ANARI for Advanced Scientific Visualization (2025), accessed: 2025-08-08
33. Thompson, A.R., Moran, J.M., Swenson, G.W.: Interferometry and Synthesis in Radio Astronomy. Springer, 3rd edn. (2017). https://doi.org/10.1007/978-3-319-44431-4
34. Unity Technologies: Texture3d (2020), https://docs.unity3d.com/2020.1/Documentation/Manual/class-Texture3D.html, accessed: 2025-07-11
35. Wald, I., Zellmann, S., Amstutz, J., Wu, Q., Griffin, K., Jaros, M., Wesner, S.: Standardized Data-Parallel Rendering Using ANARI. arXiv preprint arXiv:2407.00179 (2024), https://arxiv.org/abs/2407.00179
36. Wells, D.C., Greisen, E.W., Harten, R.H.: Fits: A flexible image transport system. Astronomy and Astrophysics Supplement Series **44**, 363 (1981)
37. Westmeier, T., Kitaeff, S., Pallot, D., Serra, P., Hulst, J.M.v.d., Jurek, R.J., Elagali, A., For, B.Q., Kleiner, D., Koribalski, B.S., Lee-Waddell, K., Mould, J.R., Reynolds, T.N., Rhee, J., Staveley-Smith, L.: SoFiA 2 – An automated, parallel HI source finding pipeline for the WALLABY survey. Monthly Notices of the Royal Astronomical Society **506**(3), 3962–3976 (Jul 2021). https://doi.org/10.1093/mnras/stab1881, http://arxiv.org/abs/2106.15789, arXiv:2106.15789 [astro-ph]
38. Westmeier, T., Serra, P., Koribalski, B.S., et al.: Sofia 2: a modular source-finding package for spectral-line data cubes. Publications of the Astronomical Society of Australia **38**, e034 (2021). https://doi.org/10.1017/pasa.2021.31