Immersive and Interactive 3D Visualization of Large-Scale Geo-Scientific Data

Markus Flatken *
German Aerospace Center (DLR)
Institute for Software Technology

Simon Schneegans †
University of Bremen
High-Performance Visualization

Riccardo Fellegara ‡
German Aerospace Center (DLR)
Institute for Software Technology

Andreas Gerndt §
German Aerospace Center (DLR)
Institute for Software Technology
and
University of Bremen
High-Performance Visualization

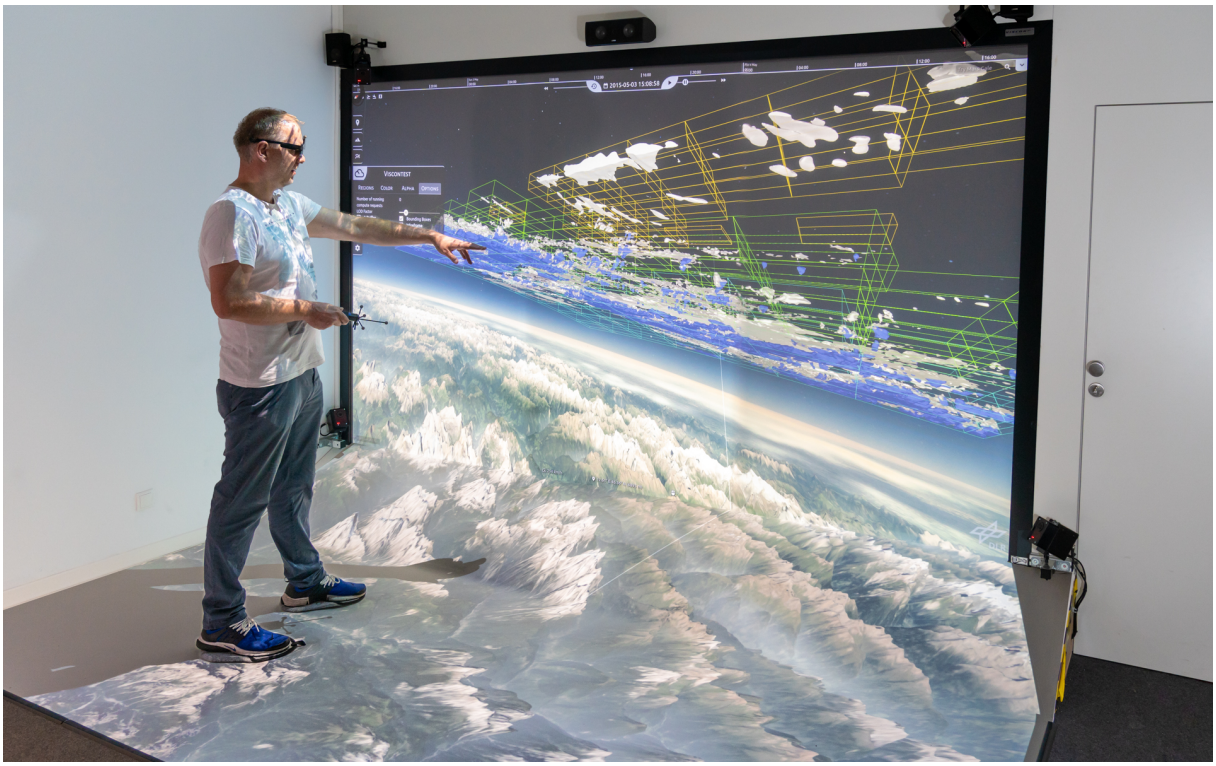

Figure 1: The image above displays an interactive visualization of a 2.6 TB weather simulation in combination with high-resolution satellite data. Interactive exploration of the time-dependent weather-simulatation data set is achieved through view-dependent, distributed feature extraction on a high-performance data-analytics cluster.

*e-mail: markus.flatken@dlr.de
†e-mail: sschneeg@uni-bremen.de
‡e-mail: riccardo.fellegara@dlr.de
§e-mail: andreas.gerndt@dlr.de

Virtual reality (VR) technology holds immense potential to revolutionize the field of geo-sciences by offering immersive and interactive experiences for data exploration and analysis. This report aims to address the challenges and opportunities associated with integrating VR into the geo-sciences domain. Our focus is on highlighting the numerous benefits that VR brings, including stereo vision, head tracking, and collaborative capabilities. These features not only facilitate knowledge exchange but also foster interdisciplinary research within the field. Nevertheless, the adoption of VR presents certain challenges that need to be addressed. These include maintaining high refresh rates, handling large datasets, and striking a balance between visualization fidelity and performance. Fortunately, significant advancements have been made in high-performance data analysis, progressive data streaming, and real-time visualization, enabling interactive exploration of large-scale datasets within VR environments. To encourage wider adoption among domain experts, we propose scaling down VR approaches to desktop setups while ensuring compatibility with VR environments. This approach allows for seamless transitions between desktop and VR experiences, leveraging immersive environments for collaboration and outreach activities. In this report, we present an architectural framework, hardware environments, use-cases, lessons learned, and the future potential of VR in the geo-sciences field. By bridging the gap between VR and geo-sciences, we anticipate transformative advancements in scientific exploration, collaboration, and knowledge dissemination.

1 Introduction

Virtual reality (VR) technology has experienced remarkable advancements in recent years, offering immersive and interactive experiences with the potential to revolutionize various domains. In the field of geo-sciences, the integration of VR holds great promise, enabling scientists to explore and analyze complex data in natural and intuitive ways, including satellite data, numerical simulations, and sensor data. In this report, we aim to bridge the gap between geo-science domains and VR, facilitating the adoption of this modern technology by domain experts.

The use of VR in geo-sciences presents numerous benefits. Firstly, VR provides scientists with stereo vision and head tracking, creating a sense of depth and spatial awareness that enhances their understanding of geological phenomena. Then, the collaborative nature of VR enables researchers to work together virtually, simulating the experience of ”being in the field” even when physically far away. Such a collaboration fosters knowledge exchange, encourages diverse perspectives, and facilitates interdisciplinary research.

However, the adoption of VR poses several challenges for visualization software. In the context of VR, maintaining a high refresh rate and low latency is crucial to avoid motion sickness and ensure a seamless experience. The use of head-mounted displays (HMDs) can be physically demanding and lead to fatigue and motion sickness. While motion sickness may be less of a problem in large-scale VR labs, setting up and maintaining these facilities can be complex, since they require sophisticated distributed software architectures to support multi-pipe display systems.

Adding to the complexity, the geo-sciences often deal with large datasets that are challenging to visualize and explore interactively. Nevertheless, achieving such visualization in VR extends the benefits beyond the immersive experience itself. It opens up possibilities for uncovering unexpected findings, facilitating rapid feedback loops, and enabling faster iterations in research.

Additionally, visualizing large data in VR serves as a powerful tool for outreach, allowing scientists to communicate their findings to the public through compelling visualizations. However, exploring large datasets in VR presents its own set of challenges, including data size, data locality, and striking a balance between visualization fidelity and performance.

In this report, we will demonstrate how significant progress has been made in addressing many of these challenges over the past few decades. We now have interactive software that runs efficiently on visualization clusters, high-performance data analysis clusters capable of extracting relevant data in near-real time, and advanced technology for low-latency progressive data streaming and level-of-detail rendering on the VR frontend. Despite these advancements, we have observed a slow adoption rate of this infrastructure among domain scientists.

Several factors contribute to this reluctance. Limited availability of facilities on-site, the absence of a comprehensive data analysis suite in VR (usually only selected tools are implemented), limitations of collaboration within single-user VR environments, challenges associated with interaction in VR (such as numeric input and the absence of physical keyboards), and the learning curve associated with navigation and interaction in VR all play a role. Furthermore, even when the system is used for outreach purposes, it typically accommodates only a few individuals at a time.

To address this problem and promote wider adoption of VR technology in the geo-sciences domain, we have opted for a scale-down approach. By adapting our software from large-scale labs to head-mounted displays (HMD) and even desktop setups, we aim to directly bring our visualization approaches to domain experts in their office environments. This approach allows scientists to leverage VR technology when needed, such as for collaborative discussions or outreach activities, while primarily working on their desktops. By maintaining familiarity with the interaction techniques developed for VR, scientists can seamlessly transition between desktop and VR environments, even preparing visualizations on the desktop and subsequently exploring

them in VR. Importantly, designing software for VR not only enables immersive experiences but also enhances desktop experiences by providing interactivity that aids exploratory data analysis tasks.

In this report, we present the hardware environments currently used in our research and the architectural framework that facilitates the development of distributed visualization systems compatible with large-scale VR, HMDs, and desktop setups. We showcase several use cases highlighting the benefits of integrating VR into geo-sciences research. Furthermore, we share the valuable lessons we have learned from our experiences and provide insights into the future potential and developments in this exciting field.

By bridging the gap between VR technology and geo-sciences, our objective is to unlock new opportunities for scientific exploration, collaboration, and knowledge dissemination. Through the convergence of immersive experiences and data visualization, we anticipate transformative advancements in the way geo-scientists interact with their data and gain insights into complex systems.

2 Related Work

Over the past few decades, the geo-sciences have witnessed a rapid increase in the volume and the complexity of data. This has led to the development of numerous visualization techniques and tools to support scientists in their data analysis tasks. In this section, we first provide, in section 2.1, an overview of the most relevant visualization techniques and tools for large-scale data analysis, and then, in section 2.2 existing approaches for integrating VR into scientific visualization workflows.

2.1 Distributed Visualization Pipeline

In large-scale data post-processing, distributed visualization systems have been used successfully for decades. The main key to achieve high interactive frame-rates is how components of the visualization pipeline are distributed. In (Bethel et al., 2013, Chap. 3), different aspects of the so-called Remote and Distributed Visualization (RDV) research have been surveyed. They specify three partitioning strategies with respect to the data types transferred between components:

- Send Data
- Send Geometry
- Send Images

Everything what can be processed remotely (on the *backend*) relieves the local visualization system (denoted as *frontend*) and fosters interactive, immersive data exploration. The *Send Data* strategies hides the storage and I/O of raw scientific data. The datasets remain on the High-Performance Computing (HPC) cluster that is linked to highly efficient and parallel file servers. The used interconnect relies on network adapters with high bandwidth and low latency. In (Bethel et al., 2000), high-speed networks are exploited in the combustion exploration framework Visapult.

In general, the frontend just requests needed data chunks of the dataset from the backend. There, the respective dataset is loaded and the requested data chunk is extracted which is then sent to the frontend. Out-of-core approaches allow a more efficient access to the data chunks without loading the entire dataset from file server. This reduces the response time of the backend. An even more reduced response time can be achieved when partial data is streamed one-by-one to the frontend. Those streaming approaches can also help when the network shows high latency. Streaming often relies on multi-resolution data structures. The backend is then processing the

data from coarse to fine resolutions which are streamed to the frontend step-by-step. One of the frameworks supporting such optimization approaches is the ViSUS system (Bethel et al., 2013, Chap. 19). In (Cohen-Or and Zadicario, 1998) the transfer load is reduced by considering only visible objects. View-dependent data extraction can also be exploited for progressive streaming and visualization (Flatken et al., 2015).

Send Data requires additional feature extraction steps on the frontend. This can be avoided when extracted geometries are directly streamed from the backend. *Send Geometry* is used by most of the key-turn applications like VisIt (Childs et al., 2011) and ParaView (Ayachit, 2015). The efficiency does not depend only on how a hardware infrastructure is distributed. It also depends on the way a post-processing algorithm is parallelized. Ahrens et al. distinguish between task, pipeline, and data parallelism (Ahrens et al., 2000). Data parallelism shows often best results on massive parallel computing resources. But this depends on the actual task. For parallel particle tracing, for instance, one can distinguish between Parallelize-over-Data (POD) and Parallelize-over-Particles (POV) (Camp et al., 2012). The last one can be considered as an out-of-core approach as it loads data when needed (load on demand).

Other applications intercept OpenGL (Shreiner et al., 2013) calls. A prominent representative of this family of approaches is Chromium (Paul et al., 2008). It routes OpenGL commands to remote (parallel) machines where the data is loaded and the visualization is performed. However, the final draw commands are transferred back to the local rendering nodes. A typical application of Chromium is rendering support for large tiled displays.

The final rendering step can also be distributed to remote render clusters. This leads to the *Send Images* strategy which send finally rendered images to the local visualization environment where they are copied to the local frame buffer(s). Parallel rendering requires sorting and redistribution of geometries and pixel images. This has been investigated in (Molnar et al., 1994). To avoid network congestion, work-load balancing for image compositing have to be

implemented. Quite generic algorithms are 2-3 Swap (Yu et al., 2008) and Radix-k (Peterka et al., 2009). Further optimizations are pixel encoding and compression. With those enhancements, Kendall et al. could demonstrate volume rendering of a supernova simulation in real-time on a 64 mega-pixel display (Kendall et al., 2010). A popular framework for parallel rendering is IceT (Image Composition Engine for Tiles), developed at Sandia national labs [1]. Parallel rendering pays off whenever datasets are huge or extracted geometry shows large amount of polygons (Shalf and Bethel, 2003). A more general performance model for 3D visualization pipeline distribution in heterogeneous computing environments is present by Bowman in (Bowman et al., 2004).

In (Bethel et al., 2000), it is presented a hybrid approach which first renders multiple images of the data from different angles before these images are composed locally for an interactive volume rendering. Those approaches belong to Image-Based Rendering (IBR) techniques and are often used for volume rendering (Müller et al., 1999). Another approach is presented by Ma et al. that is able to render a tera-scale particle accelerator simulation dataset in real-time by mixing volume rendering and point rendering (Ma et al., 2002). Another sophisticating volume rendering library, which is CPU-based and works on large-scale parallel HPC systems, is OSPRay by Intel (Wald et al., 2017).

As the efficiency of supercomputers grows much faster than the capacities of connected file servers, simulation data can become so huge that it is not possible anymore to write the data to disk in a reasonable time (cf. (Bauer et al., 2016)). Also network bandwidth is much to low to move all produced data to other computing instances for subsequent processing and visualization. This yields the unsatisfying situation that just data snapshots, which are considerably reduced in time and space, are dumped to disk. Thus, post-processing suffers from missing details and might extract artifacts and findings not embedded in the original data. The solution for this issue is co-processing, which processes data in-situ, i.e., the data is not moved but processed

[1] "https://www.sandia.gov/ccr/software/icet/"

on site where it has been produced (Fabian et al., 2011). Just the features extracted in-situ are forwarded to the frontend. One software library for in-situ co-processing is Catalyst which is tightly coupled with ParaView (Ahrens et al., 2014).

2.2 Virtual Environments

The most important benefit of immersive environments is presence. The user is dived into the virtual world and feels like in the real field. This requires full coverage of the field-of-view by computer graphics generated images. Large-scale visualization environments are appropriate for that purpose. In (Cruz-Neira et al., 1992) a multi-wall display system called CAVE has been introduced. They have also compared different Virtual Reality (VR) devices with respect to immersion and visualization issues. Their main statement is that ”one of the most important aspects of visualization is communication. For virtual reality to become an efficient and complete visualization tool, it must permit more than one user in the same environment.” Because of technical limitations, however, only one user can be tracked within a CAVE-like system. Thus, the permanently adapted user-centered projection produces incorrect images for all other CAVE users. In (Kulik et al., 2011), the projection technology to enable collaboration of up to 6 users has been enhanced. If collaboration within one physical space is not required, multiple virtual environments can be linked together to create a joint virtual workspace. In this case, avatars are rendered as user proxies which often results in situations where users feel unpleasant. This phenomenon is also known as *Uncanny Valley* (Mori et al., 2012).

Besides the 3D perception, another main advantage of virtual environments is the possibility to directly interact with the virtual objects. The direct manipulation allows to select and pick up objects directly as in the real world. In contrast, a desktop-based workplace with its desktop mouse offers indirect manipulation. A cursor on the screen is indirectly controlled by moving the mouse. Instead of grabbing an object in 3D, it is often more convenience to operate with

tools which are spatially collocated with the objects. Such tools are called *Visualization Widgets* (Bryson, 2005). Bryson recommends to configure detailed properties of visualization widgets by graphical user interface (GUI) elements well-known from desktop applications. However, they should be embedded into 3D space to allow for consistent virtual reality interaction.

In addition to the visualization widgets for user interaction, Bryson also describes requirements for system response times coming from human factors:

- *Graphics Update Rate*: This addresses the render frame rate of the scene. To preserve the sense of presence, it must be greater than 10 frames per second. In (Kreylos et al., 2003) it is discussed that 30 frames per second are the minimum to reduce discomfort in virtual environments. The graphics update rate is independent of but limited by the projector refresh rate. Cruz-Neira et al. detected flickering at 30 Hz refresh rate when using active stereo shutter glasses. Therefore, they used 60 Hz per eye which requires 120 Hz in total when using an active stereo projector (Cruz-Neira et al., 1992).

- *Interaction Responsiveness*: To give the user a good sense of e.g. selecting and manipulating objects in 3D, the latency between triggering an action and the visual response should be below 0.1 s.

- *Data Display Responsiveness*: The latency between the time data is requested and data is presented should be below 1/3 s. Data display responsiveness is less restrictive as the other two response time requirements as it is an observing task. The other two are linked to manual control and thus are much more sensitive from a human factors point of view.

These system response requirements are in general fulfilled by powerful visualization clusters. These consist of many visualization workstations each equipped with high-end computer graphics cards (GPUs). This helps to steer large multi-sided immersive environments like CAVE systems. They are also used for huge display walls consisting of many single monitors. For precise

head and input device tracking (or even for finger and body motion tracking), accurate and fast (optical) tracking systems are incorporated. The virtual environment can furthermore be enhanced by multi-modal components e.g. for force feedback or spatial sound.

When it comes to large-scale scientific data processing and distributed visualization pipeline setups, another response time plays an important role. In (Gerndt et al., 2004) it has been shown that the data display response threshold can often not been kept. Nevertheless, the user accepts some waiting time even in interactive environments as long as the graphics update rate and the interaction response times are fulfilled. Gerndt et al. defined following additional time measurement:

- *Secondary Response Time*: After a request for geometrical data, which has to be extracted first, the user accepts some time before the results are depicted. This, however, depends heavily on the processing task. Nevertheless, important is an immediate confirmation of the sent requests (e.g. as hourglass or progress bar presented within the data display response threshold). Intermediate data can bridge the waiting time until the final results are shown. Thus, data streaming and multi-resolution approaches improve the interactivity for large-scale data exploration in immersive environments considerably.

The report we present here focuses on the interactive exploration for large observation and simulation datasets in the geo-science domain. While classical scientific visualization applications such as ParaView also offer VR capabilities (Shetty et al., 2011), they are not designed for that specific use case. In the following sections, we show how the depicted techniques and constraints are incorporated into dedicated software frameworks. Appropriate algorithms support remote extraction and interactive rendering. We also present solutions for suitable GUI approaches embedded in VR which helps to control the distributed visualization pipeline execution.

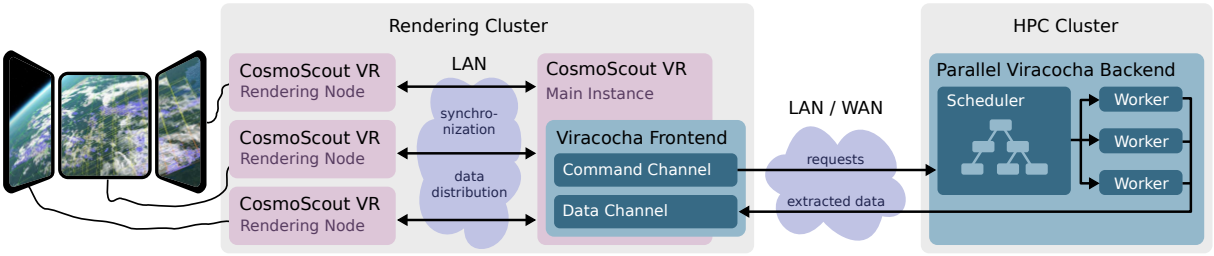

Figure 2: High-level overview of the distributed software architecture. Viracocha is integrated as a CosmoScout VR plugin to enable client/server-based analysis and visualization of large-scale datasets. Heavy workload, such as feature extraction, is outsourced to a parallel backend application running on remote resources, such as an HPC cluster. Compute requests are sent to the backend where they are distributed to a set of worker processes. The extracted features are finally streamed to the main instance of CosmoScout VR or directly to the rendering nodes. The frontend can also run on a cluster of rendering nodes, in order to drive multi-pipe virtual environments or tiled displays.

3 Software Architecture

Our software framework is composed of both a frontend and backend application. The frontend application provides interactive 3D visualization and can be configured to run on a single workstation or on a cluster of multiple machines to operate stereoscopic multi-display setups. Similarly, the backend can be configured either to run on the same machine as the frontend or on the compute nodes of a HPC cluster, allowing for distributed computation and parallel data processing. Figure 2 depicts a high-level overview of the involved components.

3.1 The Frontend: CosmoScout VR

As visualization frontend, we developed CosmoScout VR, an open source 3D solar system (Schneegans et al., 2022a, Schneegans et al., 2022b). The software enables scientists to virtually navigate from planet to planet and explore time-dependent data across many magnitudes of spatial scales. A key feature of CosmoScout VR is its precise rendering of planet-scale terrain datasets using the web map service (WMS), a standard protocol in the field of geo-sciences. These datasets in general are 2.5D elevation data, where each pixel encodes the terrain height

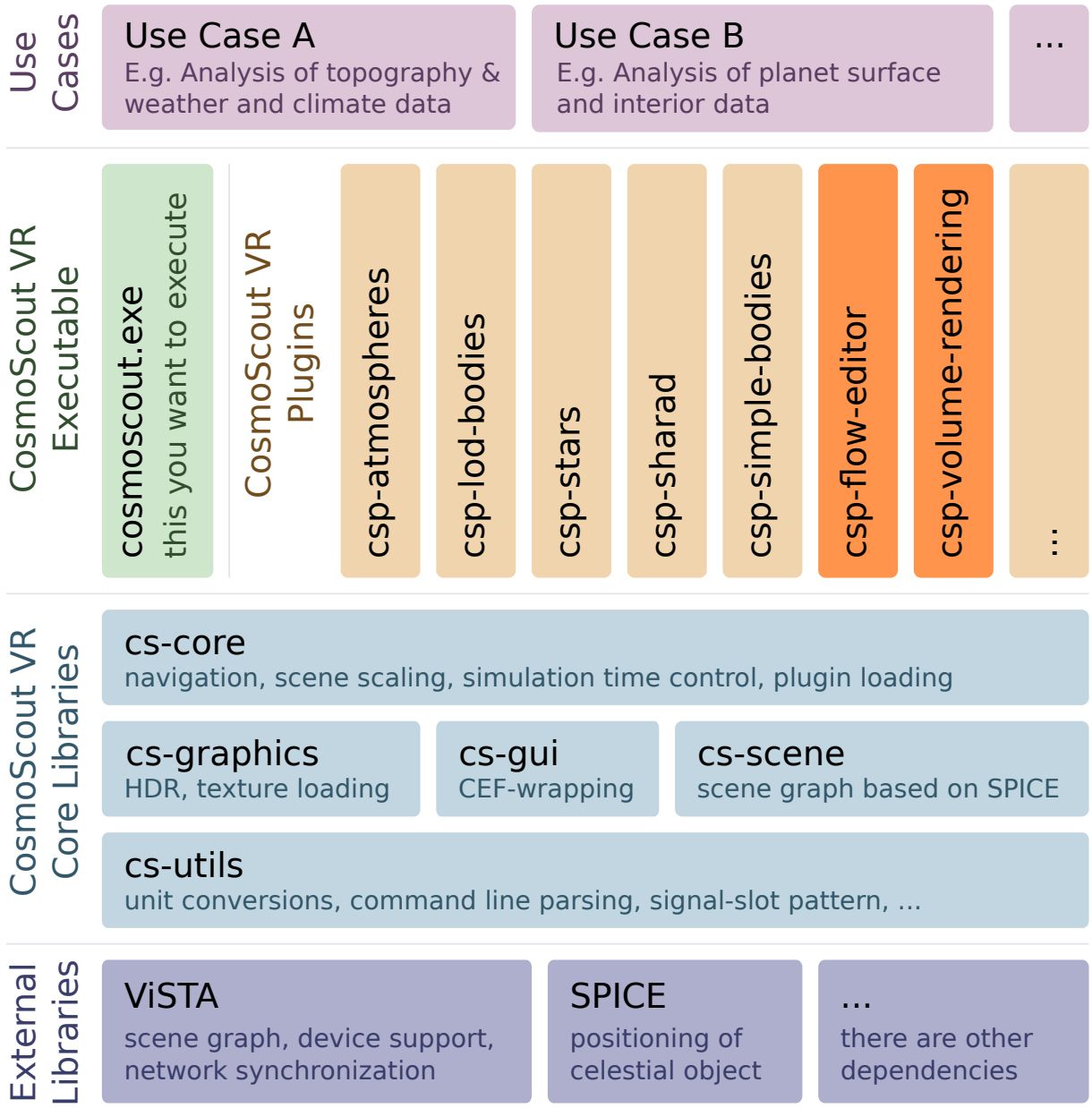

Figure 3: CosmoScout VR consists of five core libraries (cs-utils, cs-graphics, cs-scene, cs-gui, and cs-core) which are based on several third-party libraries, such as ViSTA (Assenmacher and Kuhlen, 2008) and SPICE (Acton, 1996). The actual functionality is implemented on top of these core library in form of plugins which are loaded by the CosmoScout VR executable at runtime. Which plugins are to be loaded and their configurations can be specified on a per use-case basis. This figure is based on a similar image in (Schneegans et al., 2022b).

and an optical image encoding color information into the red, green, and blue channel (RGB) of an image. In the same way, thematic maps such as offered by OpenStreetMap are integrated. In addition to these textures, we can visualize many different dataset types, as long as they have some spatial reference frame. Examples include, but are not limited to, climate or weather simulation data, multi-spectral satellite data, subsurface data, or interplanetary data such as space-weather simulations.

CosmoScout VR supports Virtual Reality devices such as head-mounted displays (HMDs) or stereoscopic multi-projection systems like CAVEs, but can also be used on traditional desktop PCs. In fact, the graphical user interface of CosmoScout VR has been developed in a desktop-first fashion. We understand that domain scientists do not typically work in Virtual Reality but rather on their local desktop workstations. Consequently, we focus on developing interactive tools that are primarily designed for desktop usage, but also scale well to VR setups as depicted in Figure 6.

Since CosmoScout VR also serves as basis toolkit to implement project related functionalities, a flexible and extensible architecture is crucial. Therefore, most of the functionalities are loaded at runtime from plugins as depicted in Figure 3. This allows for a rapid prototype development but also, since the core libraries usually do not need to be changed, that different features can be developed independently.

The core of CosmoScout VR is comprised by five libraries providing different APIs to a developer to add own functionality:

- **cs-core**: The core provides high-level functionalities such as navigation methods, functions to modify the simulation time or the scene scaling. Additionally, it includes a JSON parser for a scene configuration file describing the application state, e.g. which plugins to load with user defined parameters to set for the application.

- **cs-graphics**: This library provides functionalities for plugin developers to load e.g. textures or 3D models encoded in glTF format. Furthermore, it provides access to tone mapping values and buffers for HDR rendering (Schneegans et al., 2022b).

- **cs-gui**: This library provides an API to create and add user interfaces within CosmoScout VR. As underlaying technology, it uses HTML, JavaScript, and CSS. This gives developers the full flexibility to create complex user interfaces by using the rich web ecosystem. Finally, the web page is rendered using the Chromium Embedded Framework (CEF[2]) into an OpenGL texture. In case of running CosmoScout VR as desktop application, this website is rendered in screen-space while, when using a VR setup, the texture in placed into the 3D scene.

- **cs-scene**: This library allows adding celestial objects to the solar system. It is based on the NASA SPICE framework (Acton, 1996) for positioning within the real-time simulation. It allows building hierarchies of coordinate systems. This is required when e.g. placing the results of a weather simulation relative to the Earth center.

- **cs-utils**: This library provides basic functionalities to convert e.g. time or spatial units. Also, it provides access to a command line parser that can be used by end-users to enable command line programming or to modify the state of the scene.

The ability to scale the display size across different setups, such as from a large multi-display virtual environment, to a desktop equipped with a head-mounted display (HMD), and finally down to a local workstation connected to single monitor, is achieved by utilizing and expanding the underlying ViSTA toolkit (Assenmacher and Kuhlen, 2008).

The ViSTA toolkit plays a crucial role in this process. It enables launching CosmoScout VR as a mirrored application. In this setup, the same application is launched on multiple machines

[2]https://bitbucket.org/chromiumembedded/cef/src

driving a tiled display. The frame output is then synchronized by ViSTA over the network connection. Additionally, ViSTA provides support for integrating various VR input devices, including popular tracking systems like ART DTrack[3] or OptiTrack[4] . To further ensure compatibility with state-of-the-art HMDs like the HTC VIVE or Meta Quest 2, the ViSTA toolkit has been extended with drivers that use the OpenVR SDK[5]. This extension enables a seamless integration and support for these advanced HMDs. The application itself is then configured using multiple configuration files e.g. for output screen configuration and input device mapping for user interaction.

In order to support the analysis of large-scale geo-scientific datasets using remote HPC resources, CosmoScout VR has been extended with the *csp-flow-editor* plugin. This plugin provides a graph-based user interface (depicted in Figure 7) and rendering algorithms for polygonal meshes and large 3D volumes defined by different computational grid types. The graph-based user interface enables scientists to configure and control the data analysis pipelines executed on the backend, or modify parameters of the rendering algorithms such as a transfer function for coloring. Additionally, a parallel coordinate plot (see Figure 6) is implemented for the volume visualization. This plot provides the users with an interface to configure upper and lower thresholds for each data dimension, e.g. users can set a specific temperature and pressure range. Data in the volume, not fulfilling these thresholds, is discarded during rendering. With this feature, the user is able to highlight regions of interests in the volume. Since CosmoScout VR uses HTML, CSS and JavaScript to render the 2D user interface elements, the graph-based editor and the parallel coordinate plot is implemented using the D3 framework.

[3]https://ar-tracking.com/de/produktprogramm/dtrack
[4]https://optitrack.com/
[5]https://github.com/ValveSoftware/openvr

3.2 The Backend: Viracocha

Results of large-scale numerical simulations are usually not stored on a local hard drive. These datasets are mainly generated by using parallel solvers running on HPC resources. Copying the generated raw data, often multiple terabytes, from the HPC cluster to the local workstation for analysis is impracticable or even impossible. In order to avoid this data transfer, we opted for a client/server architecture as depicted in Figure 4.

The backend application is a massive parallel program based on the Viracocha (Gerndt et al., 2004) middleware layer, which itself is based on the message passing interface (MPI) (Clarke

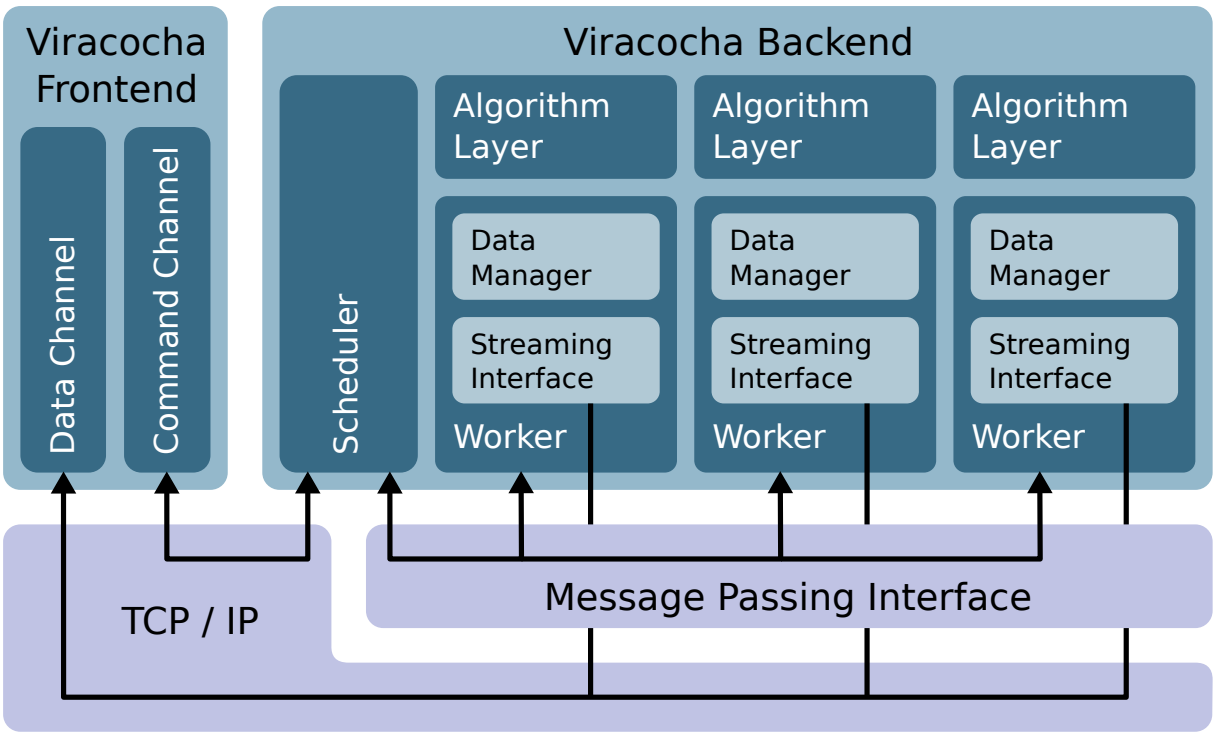

Figure 4: The Viracocha frontend library includes classes to send requests to the scheduler of the parallel backend application. The scheduler decomposes the job into independent work items and distributes them to multiple workers via message passing (MPI). These workers then execute the requested algorithm. An implemented algorithm on the worker can use data manager functionalities to handle data I/O and results are transferred to either a single frontend or a parallel rendering application via the streaming interface.

et al., 1994). We completely re-designed and extended this framework together with the University of Aachen (RWTH) to be more flexible for adding domain specific algorithms and optimized scheduling strategies for parallel data processing.

The *Viracocha Backend* library currently exploits a single *Scheduler* which distributes the workload to multiple workers. The library also provides an interface for developers to implement domain specific algorithms using the *Algorithm Layer*. This interface additionally provides access to the *Data Manager* and *Streaming Interface*. The *Data Manager* enables loading of data files and exploits caching mechanisms for efficient data I/O. The *Streaming Interface* is used to send result data back to the *Viracocha Frontend* application.

In order to define the execution workflow, two strategies are important:

- **Decomposition Strategy**: This aspect specifies how the job is decomposed into smaller work items for parallel processing. Therefore, it uses meta information of a dataset, e.g. the number of data chunks (files), their bounding boxes, and scalar ranges to create a work table with multiple work items as depicted in Figure 5. We have to notice that just meta information, as data paths, bounding boxes, time information, or scalar ranges, are encoded into the work items. Only small information and no raw data is exchanged during these stages.

- **Assignment Strategy**: This strategy outlines the process of assigning the generated work items from the work table to available workers for parallel processing. The scheduler executes this strategy, ensuring a dynamic processing order for the work items. For example, when a requested computation requires some time, the processing order is continuously updated based on the perspective of the frontend camera. By taking advantage of this functionality, it becomes possible to achieve view-dependent assignment and data processing using tree-like acceleration structures (Flatken et al., 2015).

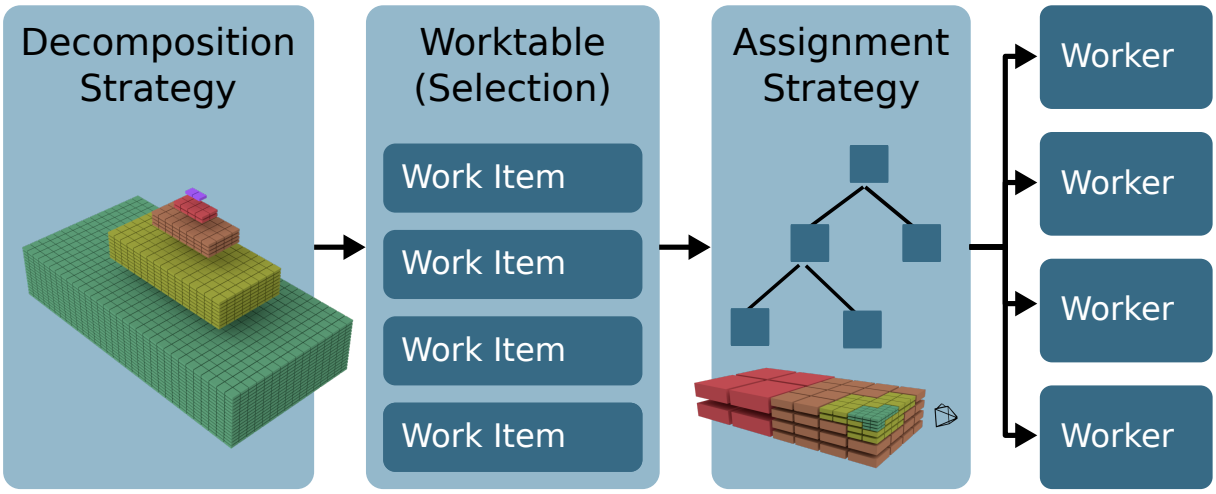

Figure 5: An initial set of work items is generated for the multi-block dataset when executing the decomposition strategy. From this selection in the work table, an tree-based acceleration structure is generated and used by the assignment strategy to dynamically adapt the processing order. This order can e.g. be view-dependent or importance driven.

Figure 5 depicts an implementation example for a hierarchical multi-block dataset. For each block of the dataset, a work item is created and added to the work table. Here, the data blocks to be processed can already be filtered out e.g. by scalar ranges when extracting iso-contours, or by their bounding boxes when executing slicing operations. From this selection, a tree-based acceleration structure is generated on-the-fly in the assignment strategy. This tree is then used to calculate the distance for the blocks to the current frontend camera position. Green colored blocks (high resolution) are close to the camera while red blocks (coarse resolution) are further away. Finally, the scheduler utilizes this data structure to assign the work items dynamically to the workers. Results are then immediatly streamed to the frontend application (CosmoScout VR) for local rendering.

This backend application is usually executed on an HPC system and allows for efficient feature extraction and data streaming. Examples include the extraction of vertical profiles or probing a specific locations of a time-dependent climate simulation (depicted in Figure 7), or the clouds of a weather simulation represented by different iso-contours (Schneegans et al., 2017) as

depicted on Figure 1.

When focusing on interactive data exploration, where parameters of the visualization pipeline are frequently updated by the user, this iterative process leads to costly data processing and heavy I/O demands. Parallelizing feature extraction already improves response times — the time a user has to wait until results are visible — but when data is really huge, execution times are still high. In order to further increase the interactivity, the assignment strategies and data streaming are employed (Flatken et al., 2015, Schneegans et al., 2017). These techniques provide a continues stream of partial data (out-of-core), where processed results are directly streamed to the frontend for visualization.

To enable the development of domain specific solutions, programming interfaces are offered as often as possible. These interfaces provided as abstract C++ classes enable application developers to integrate own algorithms, messages types, scheduling strategies, and custom user interfaces into Viracocha and CosmoScout VR.

4 Use Cases

The following subsections present some examples of domain specific applications in which we use the described system to enable interactive data analysis and visualization.

4.1 Exploration of Mantle Convection

Geo-scientists have been collecting tremendous data about our Solar system from space missions and telescope observations. It does not only help to explore the surface structure of planets. We also get an insight what happens below the surface. An important scientific interest is to get a better understanding of the interior dynamics of terrestrial planets. Observation data has enabled great progress in improving the knowledge of their thermo-chemical evolution. This has led to sophisticated models that can now be used to perform numerical simulations on HPC clusters to

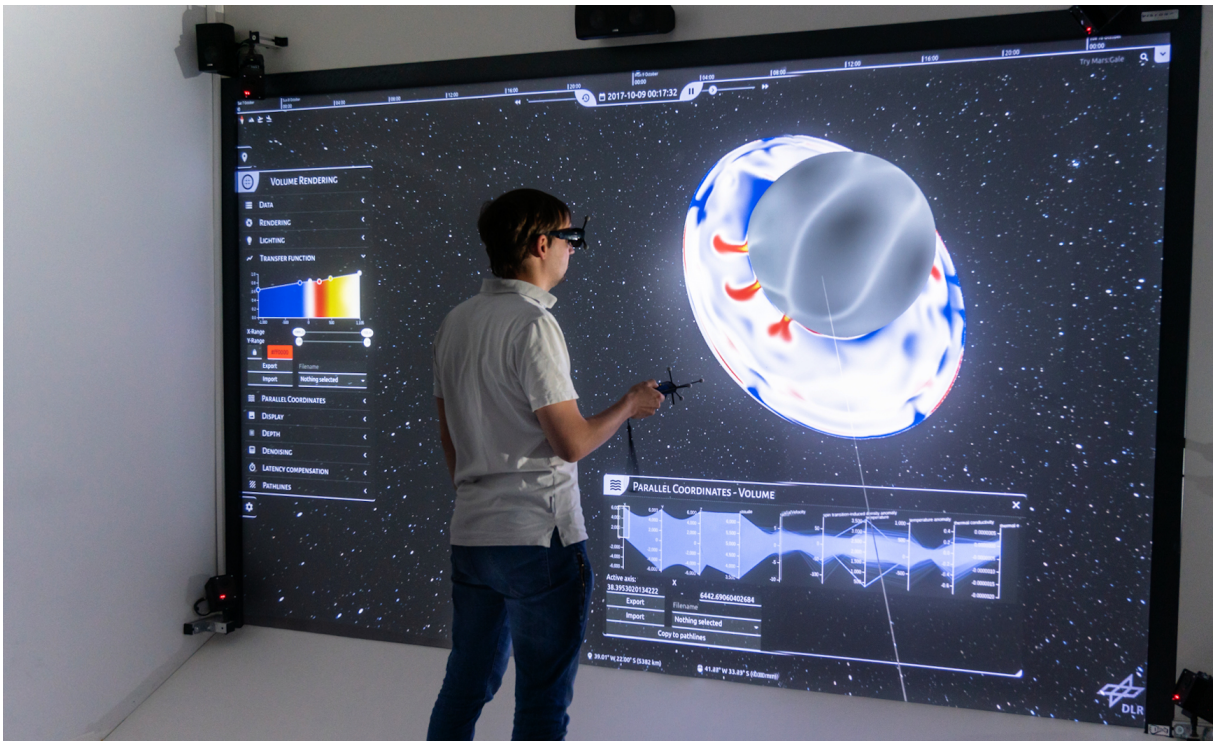

Figure 6: Volume rendering of the Mars mantle convection: Interactive exploration in VR is enabled by the desktop-based user interface of CosmoScout VR. While the interaction with complex widgets, like a transfer function editor or the parallel coordinates plot, might become quite sensitive, the embedding of well-known 2D user elements in VR eases the learning process for domain experts.

generate huge datasets about mantle convection flows (Plesa and Hüttig, 2008).

To be able to analyze the simulation data together with high-resolution terrain topography, Viracocha has been extended with appropriate processing filters e.g. to slice through the grid-based simulation data or to extract contour features from it. The execution of the filters are sped-up mainly by time-parallelization. Multiple time steps are then processed concurrently on Viracocha workers.

But we have additionally developed an interactive volume-rendering (Fritsch et al., 2021) which has been implemented as a further CosmoScout VR plug-in. The basis of this approach is Intel's OSPRay ray-tracing toolkit (Wald et al., 2017). The extracted features are transmitted

to the frontend where they are finally processed and rendered. But volume rendering is pretty time-consuming. Thus, the risk arises that interactive frame-rates can not been met. To address this issue, our ray-tracing of the volume runs asynchronously to the rendering of the remaining scene. However, this would lead to noticeable jumping effects between both render passes. Thus, we have integrated image warping techniques to hide the varying frame rates. This works also well for the generation of stereoscopic image pairs needed in immersive environments. Examples of our volume rendering approach is depicted on Figure 6.

4.2 Exploration of Climate-Chemistry Simulations

The Earth System Chemistry integrated Modelling (ESCiMo) initiative supports the climate change research community with highly accurate simulations of couple processes of the complex Earth system. One of their models has been used to simulate the chemistry of the climate system. The results are supposed to be used to support upcoming scientific assessments by the World Meteorological Organization (WMO), the United Nations Environment Programme (UNEP), and the Intergovernmental Panel on Climate Change (IPCC) (Jöckel et al., 2016).

The analysis of the data, however, poses a challenge. It consists of hundreds of chemical variables and a high temporal resolution. The simulation covers a period of nearly 100 years (50 in the past and 50 in the future) which yielded an enormous data size of more than two petabyte. But climate change depends on many uncertain parameters. An precise prediction is thus impossible. To provide nevertheless a mature data basis for an flexible post-processing, an ensemble of simulations have been computed, i.e., multiple simulation runs have been carried out, each launched with varying starting conditions.

To enable interactive analysis, the ensemble data is analyzed where they have been produced. Thus, Viracocha runs directly on HPC clusters of the German Climate Computing Center (DKRZ) in Hamburg. We have added multiple application-tailored processing filters to Viracocha which

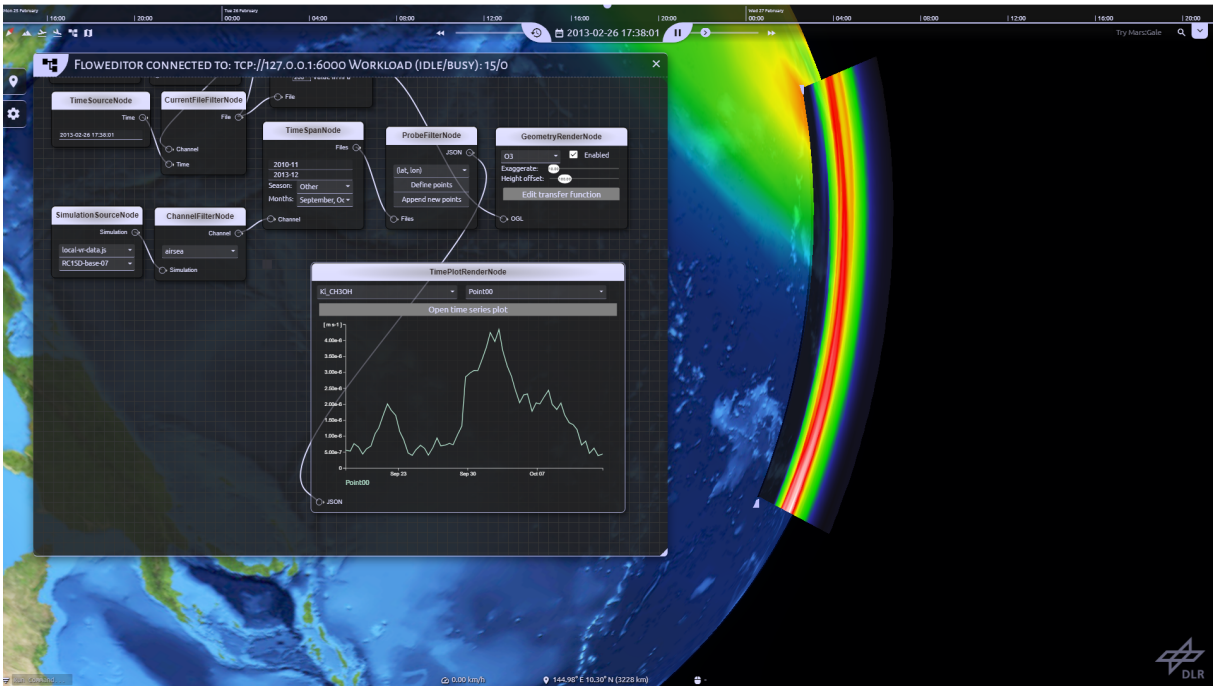

Figure 7: The graph-based user interface from the csp-flow-editor plugin is exploited to build an analysis pipeline for the ESCiMo climate-chemistry simulation. Results computed on the backend are iso-contours above the earth surface, a vertical profile trough the atmosphere, and a time series plot depicted in the flow editor.

can be carried out in parallel. The filters compute e.g. averages and standard deviations of selected variables and for specified time ranges. Also the extraction of vertical profiles using 3D fields is a usual tool for climate researchers. If these filters are applied to several simulations of the ensemble, a difference operator is able to emphasize high uncertainty in variables and show where they occur (time-dependently).

A more interactive task is probing and plotting of time series values at user defined positions. The probing location is specified on the frontend (in an virtual environment), transferred to the backend, where the extracted results are computed and streamed back to the frontend for rendering. But probing is not the only configuration task on the frontend. Rather the whole backend processing pipeline can be modified at run-time. Here, the flow editor comes into play. Figure 7 shows an example how the flow editor looks like, besides another common use-case that

maps chemical values defined in one scalar field onto isosurfaces extracted on a second scalar field.

4.3 Exploration of Weather Simulations

Climate research tries to find global relationships between several variables with considerable impact on climate change over a large-scale time period. Weather forecast, however, addresses the extraction of fine-granular phenomena in a local region and in a short period of time. Thus, the spacial resolution is much higher as for climate simulations. The number of time-steps is also high but often not as high as for long-time climate simulations.

The HD(CP)² project provided a large, time-dependent weather forecast simulation dataset as quest for the 2017 IEEE SciVis Contest (Schneegans et al., 2017). It contains 240 time steps with a total size of 2.6 TB. Each time step consists of a 3D volume of multiple scalars describing the weather conditions over Germany.

For the weather use-case, the data size of extracted features per time step is much higher as it is the case for climate data exploration. Therefore, an interactive analysis requires now much higher bandwidth from backend to frontend. However, the I/O performance is too limited to fulfill this requirement. Therefore, progressive data-streaming has been incorporated into the processing pipeline. Each time step has been decomposed into small chunks of data and then stored as leaf nodes in a multi-resolution octree. A node on higher level of the octree shows just half of the resolution as of its eight child nodes for the same region they cover (cf. also Figure 5). This allows Viracocha to extract features close to the viewer with higher resolution and features further away with descending accuracy. And nodes belonging to one time step can now be processed in parallel.

With increasing time, the resolution becomes finer. This happens without interrupting ongoing exploration. When the user moves through the data or changes a parameter, e.g. the

iso-value of an isosurface, results are progressively generated by Viracocha's workers and immediately streamed to CosmoScout VR, always updating parts in the user's vicinity first. The same happens when the simulation time progresses: Parts of the data that are in close proximity to the user reflect the change in a few milliseconds, while data further away may take a few seconds to update.

The teaser Figure 1 presents a screenshot of the weather use case. In Section 6, we present some performance results demonstrating the efficiency of the implemented processing approaches.

5 Hardware Setup

The software framework runs intensively every day in the Virtual Reality laboratory at the German Aerospace Center (DLR) in Brunswick, Germany (shown in Figure 8). This gives us plenty of opportunities to test and evaluate it frequently.

The VR lab is operated with active stereo projectors and an optical tracking system with 6 cameras. The tracking is used not only for head tracking and input device tracking. We have used it as well for pseudo-haptic finger tracking investigations (Hummel et al., 2016). The lab is equipped with four Barco F50 projectors with WXGA resolution. Three projectors have been mounted in a backroom to beam their images on the rear of the main display. Edge blending with overlapping regions reduces the overall resolution but still provides high-resolution images without any gaps. The remaining projector has been mounted at the ceiling for the floor projection. The shadow is slightly casted backwards, i.e., it is not disturbing. Each projector is connected to a workstation equipped with 2x Intel Xeon E5-2630 v3 CPUs, 128 GB of main memory, and a single NVIDIA Quadro P6000 graphics card.

For distributed data processing, the VR-Lab is linked to a high-performance data analytics (HPDA) cluster. There, the Viracocha backend application is running and waiting for processing

Figure 8: The Virtual Reality lab at the German Aerospace Center in Brunswick, Germany is used regularly to evaluate the framework described in this report. The lab uses four stereo projectors (three for the back projection and one for the floor projection). Each projector is connected to a separate workstation. Images courtesy of German Aerospace Center, DLR (CC-BY).

jobs. To incorporate high-resolution satellite imagery, we have connected a further powerful server into the setup, which hosts a MapServer [6].

The HPDA cluster is comprised of four CPU-based compute nodes. Each node is equipped with four Intel Xeon Skylake 6132 processors and has 384 GB of main memory. The CPUs run at 2.60 GHz and have 14 cores each. In total, 224 cores are available for parallel processing. The HPDA cluster is connected with 4 Gbit/s to the VR-Lab. Additional to this HPDA cluster, the backend has also been executed on the Mistral high-performance computing cluster located at the German Climate Computing Center (Deutsches Klimarechenzentrum, DKRZ). This cluster is used for analysis of climate chemistry simulations, as these massive datasets cannot be easily transferred to a local workstation or other cluster resources. The Mistral cluster provides more

[6] `https://mapserver.org/`

than 100,000 CPU cores with a performance of 3.6 PFLOPS.

The supply of high-resolution satellite imagery is achieved using the Web Map Service (WMS) standard. Therefore, we deployed an in-house MapServer which is equipped with two Intel® Xeon® Gold 6148 processors. Each processor has 20 cores and supports hyper threading. Overall, this server provides 40 CPU cores and has a total of 192 GB main memory available. The WMS interface of CosmoScout VR, however, allows to connect to any open WMS server on the web.

6 Performance Example

The combination of CosmoScout VR on the VR-cluster frontend and Viracocha on the HPDA backend is for highest visualization performance and data throughput. To benchmark and demonstrate the efficiency of our solutions, we have taken the weather simulation data and its tailored plugins (cf. Subsection 4.3). Figure 9 depicts the measured timings for parallel and view-dependent extraction of four isosurfaces.

Timings are captured starting from a low resolution perspective, where surfaces are generated for just a few octree nodes. When zooming in, the resolution increases and a continuous stream of requests is sent to the backend (orange line). Most of the time, enough workers are available to instantly process the requests and send the extracted geometry of the isosurfaces back to the frontend. The amount of rendered octree nodes (green dotted line) steadily increases up to 1103. At frame 288, the simulation time step is changed, which leads to a complete re-processing of all visible nodes. As requests are prioritized by view distance, the octree nodes close to the camera are updated first. Initial results are available after about 80 ms, ensuring the required response times. And after 2.4 s, the complete scene is updated. One time step of the octree contains about 12.6 GB of data from which 658.4 MB of geometry have been extracted and sent to the CosmoScout VR client for the benchmarked perspective.

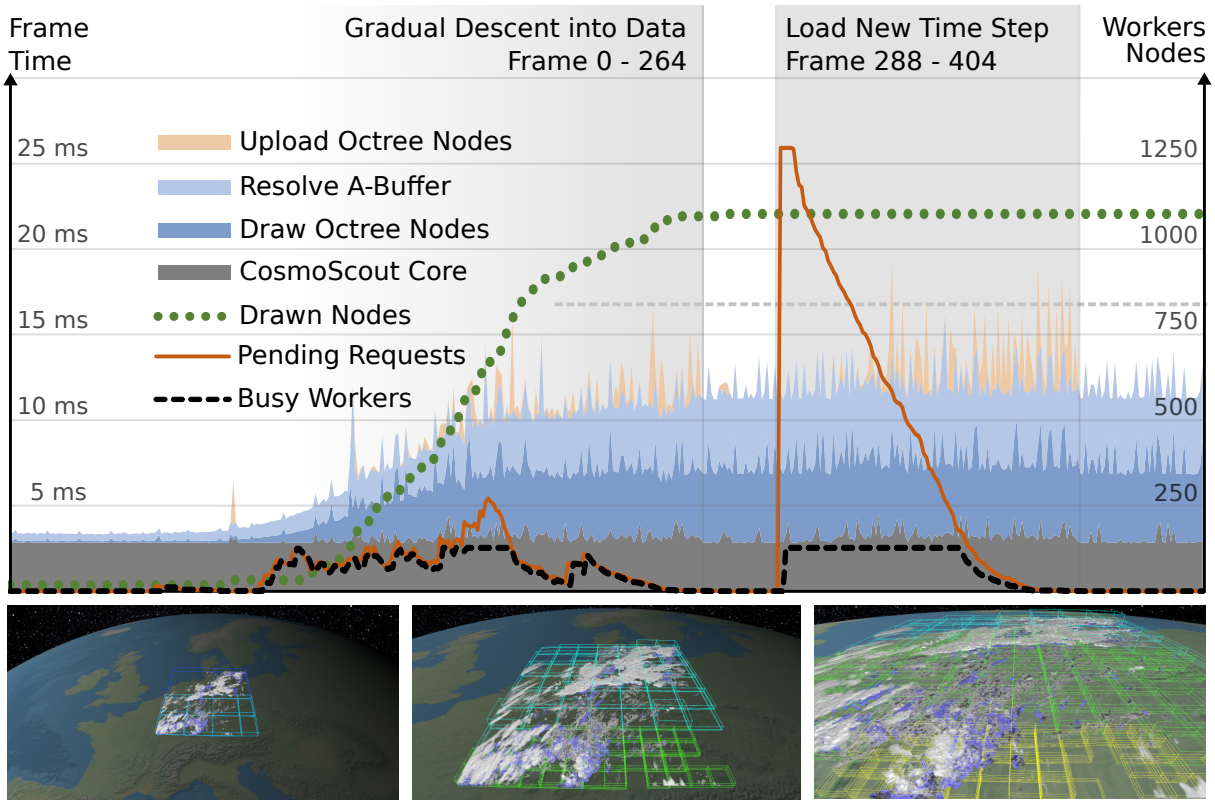

Figure 9: The graph shows different aspects of the work load for rendering 450 frames. These aspects are GPU timings, the count of rendered octree nodes, pending compute requests, and the currently active worker processes on the parallel backend. For the first half of the frames, the load is measured while the camera approaches Germany where the weather data is depicted. At frame 288, the visualized time step changes, resulting in numerous pending requests. The dotted grey line indicates a frame time of 16.67 ms (60Hz frame rate threshold).

The table below depicts the achieved timings when launching the backend with a varying number of MPI processes.

Backend Processes	8	16	32	64	128	256
Update Time	25.0 s	12.7 s	6.7 s	3.9 s	2.4 s	2.3 s

The time for re-processing of all 1103 visible nodes with 8 MPI processes took 25 s. When running with 256 MPI processes, the time could be reduced to 2.3 s. It is worth noting that the scaling is nearly linear up to 128 processes. Beyond that, the efficiency decreases due to

limitations in network bandwidth between the CosmoScout VR clients and the HPDA cluster, as well as the need for disk reads of the raw data. Despite these limitations, the overall frame time remained mostly below 16.67 ms throughout the session, allowing us to interactively explore the 2.6 TB dataset.

7 Conclusion

Our success story is defined around the development of an environment that enables the interactive exploration of scientific datasets of any scale. The main success factor is the incorporation of high-performance hardware components within a distributed setup and the development of distributed visualization methodologies. This distribution leverages the strengths of each processing stage. We have demonstrated the possible extreme speed-up we could gain in the backend with high-performance data analysis clusters. For the geo-science domain, the integration of map servers, that provides us with high-resolution satellite imagery and relies on standards from the geo-scientific community, is important not only for the efficiency but also for a high acceptance by domain experts.

Our main objective for the frontend is the use of large immersive virtual environments that requires a powerful hardware setup for interactive 3D visualization. In this report, we have presented a software architecture that effectively combines backend and frontend methodologies. We have demonstrated the scalability of our system, that scales down from a large multi-display virtual environment to a single-user desktop computer. This supports a continuous working environment for geo-scientists. We have recognized that a complete migration of all working steps into virtual environments is not desirable. Rather, the working processes have to be flexible enough to switch seamlessly from one environment to another. However, regardless of the setup, interactivity has the highest priority. Therefore, we have implemented suitable approaches, such as view-dependent data extraction and image warping, ensuring interactive frame rates. We

evaluated that even desktop-based applications can benefit from these implementations.

To emphasize the usefulness of our distributed approaches for interactive large-scale data analysis, we have presented several use-cases. They proved that the setup is very efficient and contains high potential to facilitate insightful analysis within virtual environments. With a focus on weather simulation, performance measurements have been carried out which evaluated and validated the system's ability to handle large-scale datasets in more detail.

7.1 Lessons Learned

Although we have made a powerful data exploration solution available, we have to admit that we have not been fully successful to bring the geo-science domain to virtual reality. The adoption of domain processes and tools to VR poses several still open challenges that hinder widespread acceptance. More practical reasons are:

- large immersive environments are very expensive;
- virtual environments are often located in separate buildings or even off-site. This requires some effort to overcome the reluctance to leave her or his office;
- scientists prefer the ease and comfort of working with their familiar local desktop machines;
- turnkey software solution is not provided. The complexity involved in launching VR applications in high-performance and distributed environments further impede the adoption of geo-science tools;
- the advantages of immersive exploration is considered limited compared to the work at a local desktop. Furthermore, implementations does not match the full functionality offered by common domain-specific tools.

Human factors play a crucial role in our work. While we have successfully deployed applications in our distributed environment, it is important to address the challenges associated with working in virtual reality (VR), which can lead to increased workload and physical fatigue, such as arm fatigue from using 3D controllers. This issue is more evident in geo-science applications, where navigating through (virtual) fields is a common task. However, we observed that (indirect) navigation can quickly induce discomfort, also known as motion sickness or cyber sickness.

While direct manipulation and interaction are considered key advantages of VR, the existing graphical user interfaces (GUIs) and interaction methods supported in VR are often not well-designed for precise and numerical input. This limitation hinders the seamless integration of VR into scientific workflows.

As a result, scientists often choose to remain in their offices and continue using the desktop version instead, relegating VR to be primarily used for outreach or marketing purposes. Head-mounted displays (HMDs), while cost-effective, are not deemed suitable for seamless VR integration. Maybe smaller, easy to install VR office setups can help. Therefore, LED displays with high frame rates are very promising. However, it is even more crucial to improve the perceived benefits and the added value of VR applications. This might convince scientists to use VR as a daily working tool. Also, navigation and interaction approaches have to extensively consider human factors. Currently, there is a lack of established standards in common software libraries. We believe that by further addressing these issues, VR can be more than a tool for gamers and enthusiasts, and, thus become a valuable asset in scientific domains.

In the meantime, although focusing on virtual reality as main target, real-time data processing for interactive exploration can be exploited for great performance improvements in desktop application as well. High update rates and low response times enables large-scale data exploration even on desktops.

Figure 10: AR working environments allow intuitive on-site collaboration sessions. Each user possesses the correct perspective to the real world and superimposed virtual scenes. This considerably supports joint interactive geo-science data exploration and mission planning activities.

7.2 Future Work

A key advantage of large-scale immersive environments is collaboration. Therefore, our focus will be on developing software components that support collaborative scientific data exploration among a group of domain experts. Multi-user VR, supporting more than one user with a correct view perspective in one virtual environment, holds a great potential for enabling discussions and collaborative discoveries. Additionally, we are also considering to support theater-like setups, where a larger audience can be immersed into the virtual environment. In such a configuration, head tracking will not be enabled, but the use of stereo vision can still provide a fascinating insight into scientific findings.

A major drawback of current stereo projectors is the low pixel density and reduced dynamic

range. Large display walls based on current monitor technologies overcome those issues. LED walls, especially in multi-tiled, non-planar installations, are even more promising. The visual quality might be much better as the view angles to the wall does not play a negative effect on the experience. Additionally, display walls are often equipped with finger touch capacities, which can reduce the issues with GUIs in VR. It also offers great opportunities to combine scientific exploration with Visual Analytics (VA).

While complete immersion into virtual worlds has many applications, even in geo-science, Mixed Reality (MR) can provide exceptional experiences, as some of our initial research in that direction has shown (see Figure 10). Therefore, our future research will increasingly focus on immersive augmented reality using see-through head-mounted displays and augmented reality glasses. These devices offer great opportunities for collaborative scientific data analysis while significantly reducing the issue of cyber sickness. Moreover, since AR devices are far handier, they can also bring intuitive data exploration into the office of a domain expert. Depending on the requirements, this can serve as a transformation of geo-science applications to the Virtuality Continuum, i.e., from MR to VR (and backwards).

However, given the processing power of current augmented reality devices, the design and implementation of a framework which enables interactive visualization of large scientific datasets on such devices is still a grand challenge. Nevertheless, we are confident that the distributed data processing pipeline, presented in this report, will serve as solid basis for this endeavor.

References

Acton, 1996. Acton, C. H. (1996). Ancillary data services of NASA's Navigation and Ancillary Information Facility. *Planetary and Space Science*, 44(1):65 – 70.

Ahrens et al., 2014. Ahrens, J., Jourdain, S., O'Leary, P., Patchett, J., Rogers, D., Fasel, P., Bauer, A., Petersen, M., Samsel, F., and Boeckel, B. (2014). In situ MPAS-ocean image-based visualization. In *The International Conference for High Performance Computing, Networking, Storage and Analysis (SC14), Visualization & Data Analytics Showcase, November 16-21*, pages 1–4, New Orleans, LA.

Ahrens et al., 2000. Ahrens, J., Law, C., Schroeder, W., Martin, K., and Papka, M. (2000). A parallel approach for efficiently visualizing extremely large, time-varying datasets. techreport #LAUR-00-1620, Los Alamos National Laboratory.

Assenmacher and Kuhlen, 2008. Assenmacher, I. and Kuhlen, T. (2008). The ViSTA virtual reality toolkit. *Proceedings of the IEEE VR SEARIS*, pages 23–26.

Ayachit, 2015. Ayachit, U. (2015). *The ParaView Guide: A Parallel Visualization Application*. Kitware, Inc., Clifton Park, NY, USA.

Bauer et al., 2016. Bauer, A. C., Abbasi, H., Ahrens, J., Childs, H., Geveci, B., Klasky, S., Moreland, K., O'Leary, P., Vishwanath, V., Whitlock, B., and Bethel, E. W. (2016). In situ methods, infrastructures, and applications on high performance computing platforms. *Computer Graphics Forum*, 35(3):577–597. STAR – State of The Art Report.

Bethel et al., 2013. Bethel, E. W., Childs, H., and Hansen, C., editors (2013). *High Performance Visualization: Enabling Extreme-Scale Scientific Insight*. CRC Press, Taylor & Francis, Boca Raton, Fla.

Bethel et al., 2000. Bethel, W., Tierney, B., Lee, J., Gunter, D., and Lau, S. (2000). Using high-speed WANs and network data caches to enable remote and distributed visualization. In *Proceedings of the 2000 ACM/IEEE Conference on Supercomputing (SC00)*. LBNL-45365.

Bowman et al., 2004. Bowman, I., Shalf, J., Ma, K.-L., and Bethel, W. (2004). Performance modeling for 3d visualization in a heterogeneous computing environment. techreport LBNL-56977, Lawrence Berkeley National Laboratory.

Bryson, 2005. Bryson, S. (2005). Direct manipulation in virtual reality. In Hansen, C. D. and Johnson, C. R., editors, *The Visualization Handbook*, chapter 21, pages 413–430. Elsevier Academic Press.

Camp et al., 2012. Camp, D., Childs, H., Garth, C., Pugmire, D., and Joy, K. I. (2012). Parallel stream surface computation for large data sets. In *IEEE Symposium on Large Data Analysis and Visualization (LDAV)*. IEEE.

Childs et al., 2011. Childs, H., Brugger, E., Whitlock, B., Meredith, J., Ahern, S., Bonnell, K., Miller, M., Weber, G. H., Harrison, C., Pugmire, D., Fogal, T., Garth, C., Sanderson, A., Bethel, E. W., Durant, M., Camp, D., Favrek, J. M., Rubel, O., Navratil, P., Wheelera, M., Selbya, P., and Vivodtzev, F. (2011). VisIt: An end-user tool for visualizing and analyzing very large data. In *Proceedings of SciDAC Conference 2011, Denver, CO, The Brown Palace Hotel, October 07, 2011*. Department of Energy Office of Science.

Clarke et al., 1994. Clarke, L., Glendinning, I., and Hempel, R. (1994). The MPI message passing interface standard. Technical report.

Cohen-Or and Zadicario, 1998. Cohen-Or, D. and Zadicario, E. (1998). Visibility streaming for network-based walkthroughs. In *Proceedings of the Graphics Interface Conference, Vancouver, BC, Canada, June 18-20, 1998*.

Cruz-Neira et al., 1992. Cruz-Neira, C., Sandin, D. J., DeFanti, T. A., Kenyon, R. V., and Hart, J. C. (1992). The CAVE: audio visual experience automatic virtual environment. *Communications of the ACM*, 35(6):64–72.

Fabian et al., 2011. Fabian, N., Moreland, K., Thompson, D., Bauer, A. C., Marion, P., Gevecik, B., Rasquin, M., and Jansen, K. E. (2011). The ParaView coprocessing library: A scalable, general purpose in situ visualization library. In *2011 IEEE Symposium on Large Data Analysis and Visualization*. IEEE.

Flatken et al., 2015. Flatken, M., Merkel, J., Berres, A., Hotz, I., Gerndt, A., and Hagen, H. (2015). Dynamic scheduling for progressive large-scale visualisation. In Bertini, E., Kennedy, J., and Puppo, E., editors, *Proceedings of Eurographics Conference on Visualization (EuroVis), Short Papers, Cagliari, Italy, May 25-29, 2015, pp. 37–41, , 2015.*, pages 37–41. The Eurographics Association.

Fritsch et al., 2021. Fritsch, J., Flatken, M., Schneegans, S., Gerndt, A., Plesa, A.-C., and Hüttig, C. (2021). RayPC: Interactive ray tracing meets parallel coordinates. In *IEEE Visualization Contest 2021*.

Gerndt et al., 2004. Gerndt, A., Hentschel, B., Wolter, M., Kuhlen, T., and Bischof, C. (2004). VIRACOCHA: An efficient parallelization framework for large-scale CFD post-processing in virtual environments. In *The International Conference for High Performance Computing and Communications, SC2004*, pages 6–12, Pittsburgh, PA, USA.

Hummel et al., 2016. Hummel, J., Dodiya, J., Eckardt, L., Wolff, R., Gerndt, A., and Kuhlen, T. W. (2016). A lightweight electrotactile feedback device for grasp improvement in immersive virtual environments. In Höllerer, T., Interrante, V., Lécuyer, A., and Suma, E., editors, *IEEE Virtual Reality Conference (VR), Greenville, SC, USA, March 19-23, 2016*, pages 39–48. IEEE.

Jöckel et al., 2016. Jöckel, P., Tost, H., Pozzer, A., Kunze, M., Kirner, O., Brenninkmeijer, C. A. M., Brinkop, S., Cai, D. S., Dyroff, C., Eckstein, J., Frank, F., Garny, H., Gottschaldt, K.-D., Graf, P., Grewe, V., Kerkweg, A., Kern, B., Matthes, S., Mertens, M., Meul, S., Neumaier, M., Nützel, M., Oberländer-Hayn, S., Ruhnke, R., Runde, T., Sander, R., Scharffe, D., and Zahn, A. (2016). Earth system chemistry integrated modelling (ESCiMo) with the modular earth submodel system (MESSy) version 2.51. *Geoscientific Model Development*, 9(3):1153–1200.

Kendall et al., 2010. Kendall, W., Peterka, T., Huang, J., Shen, H.-W., and Ross, R. (2010). Accelerating and benchmarking Radix-k image compositing at large scale. In Ahrens, J., Debattista, K., and Pajarola, R., editors, *Eurographics Symposium on Parallel Graphics and Visualization*. The Eurographics Association.

Kreylos et al., 2003. Kreylos, O., Bethel, W., Ligocki, T., and Hamann, B. (2003). Virtual-reality-based interactive exploration of multiresolution data. In Farin, G., Hamann, B., and Hagen, H., editors, *Hierarchical and Geometrical Methods in Scientific Visualization*, pages 205–224. Springer, Heidelberg.

Kulik et al., 2011. Kulik, A., Kunert, A., Beck, S., Reichel, R., Blach, R., Zink, A., and Froehlich, B. (2011). C1x6: A stereoscopic six-user display for co-located collaboration in shared virtual environments. In *Proceedings of the 2011 SIGGRAPH Asia Conference*. ACM.

Ma et al., 2002. Ma, K.-L., Schussman, G., Wilson, B., Ko, K., Qiang, J., and Ryne, R. (2002). Advanced visualization technology for terascale particle accelerator simulations. In *Supercomputing Conference, Baltimore, MD, USA*, pages 19–30.

Molnar et al., 1994. Molnar, S., Cox, M., Ellsworth, D., and Fuchs, H. (1994). A sorting classification of parallel rendering. *IEEE Computer Graphics and Applications*, 14(4):23–32.

Mori et al., 2012. Mori, M., MacDorman, K., and Kageki, N. (2012). The uncanny valley [from the field]. *IEEE Robotics & Automation Magazine*, 19(2):98–100.

Müller et al., 1999. Müller, K., Shareef, N., Huang, J., and Crawfis, R. (1999). IBR-assisted volume rendering. In *Proceedings of IEEE Visualization 1999, Late Breaking Hot Topics, San Francisco, CA, October 24-29, 1999*.

Paul et al., 2008. Paul, B., Ahern, S., Bethel, E. W., Brugger, E., Cook, R., Daniel, J., Lewis, K., Owen, J., and Southard, D. (2008). Chromium Renderserver: Scalable and open remote rendering infrastructure. *IEEE Transaction on Visualization and Computer Graphics (TVCG)*, 14(3):627–639.

Peterka et al., 2009. Peterka, T., Goodell, D., Ross, R., Shen, H.-W., and Thakur, R. (2009). A configurable algorithm for parallel image-compositing applications. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis - SC'09*. ACM Press.

Plesa and Hüttig, 2008. Plesa, A.-C. and Hüttig, C. (2008). Numerical simulation of planetary interiors: Mantle convection in a 2D spherical shell. In *Workshop on Geodynamics*.

Schneegans et al., 2022a. Schneegans, S., Flatken, M., and Gerndt, A. (2022a). CosmoScout VR.

Schneegans et al., 2017. Schneegans, S., Neary, L., Flatken, M., and Gerndt, A. (2017). STRIELAD - a scalable toolkit for real-time interactive exploration of large atmospheric datasets. In *IEEE Visualization Contest 2017*.

Schneegans et al., 2022b. Schneegans, S., Zeumer, M., Gilg, J., and Gerndt, A. (2022b). CosmoScout VR: a modular 3D solar system based on spice. In *2022 IEEE Aerospace Conference (AERO)*, pages 1–13. IEEE.

Shalf and Bethel, 2003. Shalf, J. and Bethel, E. W. (2003). The grid and future visualization system architectures. *IEEE Computer Graphics and Applications*, 23(2):6–9.

Shetty et al., 2011. Shetty, N., Chaudhary, A., Coming, D., Sherman, W. R., O'Leary, P., Whiting, E. T., and Su, S. (2011). Immersive paraview: A community-based, immersive, universal scientific visualization application. In *2011 IEEE Virtual Reality Conference*, pages 239–240.

Shreiner et al., 2013. Shreiner, D., Sellers, G., Kessenich, J., and Licea-Kane, B. (2013). *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.3*. Addison Wesley, 8th edition. Khronos OpenGL ARB Working Group.

Wald et al., 2017. Wald, I., Johnson, G., Amstutz, J., Brownlee, C., Knoll, A., Jeffers, J., Gunther, J., and Navratil, P. (2017). Ospray - a cpu ray tracing framework for scientific visualization. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):931–940.

Yu et al., 2008. Yu, H., Wang, C., and Ma, K.-L. (2008). Massively parallel volume rendering using 2-3 swap image compositing. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing (SC'08)*, number 48.