IPT-EGVE Symposium (2007)
B. Fröhlich, R. Blach, and R. van Liere (Editors)

Interactive Exploration of Large Data in Hybrid Visualization Environments

M. Schirski[†1], C. Bischof[2] and T. Kuhlen[1]

[1]Virtual Reality Group, RWTH Aachen University, Germany
[2]Institute for Scientific Computing, RWTH Aachen University, Germany

Abstract

With rising data sizes and growing complexity, the results of modern numerical simulations are increasingly difficult to understand. Thus, using Virtual Reality methodology for an interactive analysis of such data gains more and more importance. However, interaction within virtual environments comes at the cost of real-time constraints, which are difficult to meet. Using a hybrid visualization environment consisting of a high-performance computing (HPC) system connected to a graphics workstation (or multiple rendering nodes) we propose a workload distribution which significantly increases interactivity during the data analysis process. Based on a novel model of the exploration process, we introduce an additional step into the conventional visualization pipeline before mapping the whole process onto system components. This incorporates the respective benefits of high-performance computing and GPU-based computation into a single visualization framework. Basically, by coupling an HPC-based extraction of a region-of-interest to GPU-based flow visualization, an interactive analysis of large datasets is made possible. Taking interactive particle tracing and volume rendering as examples, we show the applicability of our approach to an interactive exploration of datasets exceeding the memory limits of a single workstation.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Virtual Reality, I.3.6 [Computer Graphics]: Interaction Techniques, I.6.6 [Simulation and Modeling]: Simulation Output Analysis

1. Introduction

Due to large data sizes and their inherent complexity, the results of modern numerical simulations are difficult to understand. Thus, scientific visualization is vital for an in-depth comprehension of computational fluid dynamics (CFD) data. Especially the use of Virtual Reality (VR) methodology for an interactive exploration is gaining more and more importance, as it significantly facilitates the analysis of complex, three- to four-dimensional flow phenomena (cf. Figure 1). Nevertheless, dealing with immersive virtual environments creates real-time constraints, which have to be observed in order to maintain the usability of a given VR system.

While short response times are desirable for any visualization system, they become a vital requirement in the context of virtual environments, up to the need for full interactivity, i.e. direct visual feedback to user input. However, for data of reasonable size, a fully interactive exploration is typically

Figure 1: *Interactive exploration of the air flow in New Orleans inside an immersive virtual environment.*

† e-mail: schirski@rz.rwth-aachen.de

impossible with a single workstation, let alone a visualization system driving an immersive display. This holds true especially for time-varying datasets, which tend to exceed memory capabilities of modern workstations quite quickly.

In order to still use the potential of VR for an interactive exploration of large datasets, we propose a paradigm shift regarding the usage of HPC systems for supporting interactive exploration of flow data. After an analysis of the exploration process and available hardware resources, the visualization pipeline is refined and visualization tasks are distributed to the components of a given visualization system. Besides a better workload distribution, this accounts for the increasing computational power of workstations through multi-core CPUs or programmable graphics processing units (GPUs).

Using interactive particle tracing as an example, we demonstrate the effectiveness of our approach for an intuitive exploration of large datasets. Staying within appropriate interactivity and low-latency constraints allows for a direct interaction and analysis of the given data inside an immersive virtual environment. Furthermore, the applicability of this approach is shown by additional visualization methods, e.g. direct volume rendering.

The remainder of this paper is structured as follows: After an outline of related work in section 2, section 3 describes contemporary visualization systems, followed by an analysis of the interactive exploration process in section 4. Consequences of this analysis regarding the visualization pipeline are given in section 5. A detailed description of our approach and its implementation are provided in section 6, including exemplary application for interactive particle tracing and direct volume visualization. Results are presented in section 7, and section 8 draws some conclusions and gives an outlook at future work.

2. Related Work

Contemporary descriptions of the data flow within the visualization pipeline are fairly complete [UJK*89, SML06]. Nevertheless, additional information like computational complexity, data sizes and/or execution frequency, which are necessary for making optimal use of available computing resources (especially in an interactive context), are missing. This situation is quite similar regarding the exploration process. The addition of human factors by embedding a computer-assisted analysis into a complete data exploration process [SBM92] or by defining response time requirements [BJ96] is only marginally helpful for determining corresponding design consequences. The same applies to recent research considering human factors, which deals mainly with perceptional issues [TM04]. Current models for the exploration process are targeted at recording, analyzing, and reproducing data analysis sessions [JKMG02]. However, they consider and document the progression through a visualization process, rather than computational tasks and their cost. While these works provide valuable insight into the user's participation in the exploration process, they tend to omit issues concerning waiting time and latency completely – with the exception of [BJ96].

VR methodology has been adopted and successfully employed for the exploration of flow fields in the past [BL91]. However, processing simulation results of reasonable size interactively is still non-trivial. The typical approach to dealing with large-scale data is employing remote (parallel) computing systems for generating graphical primitives for visualization objects like cut planes or streamlines, which are then displayed interactively on a dedicated visualization front-end [BGY92, RFL*98, GHW*04]. This comes at the cost of increased latency, as every user-initiated parameter change results in a query being sent to the remote system, where the computation is executed before the corresponding results are transmitted back to the visualization front-end in order to be displayed. While the total computation time is reduced in comparison to an execution on the front-end alone, network latency and bandwidth limitations typically prevent direct visual feedback to the user.

An alternative is the parallel generation of image data using computing clusters, followed by pixel data transmission, accumulation and display. While this approach is quite successful for direct volume rendering of large data [SMW*05] and/or large polygonal models [HHN*02, RR06] in comparison to direct rendering on a single visualization host, it is rather problematic for being employed in an immersive virtual environment. Bandwidth limitations impede remote (stereo) image generation for multiple display screens, and network latency results in relatively high response times to user movement causing loss of responsiveness and potentially inducing cyber-sickness.

With the advent of flexible and powerful graphics processing units (GPUs), a lot of effort has been spent on using them as a resource for general purpose computing – an overview can be found in [OLG*07]. Most important in the context of our work are applications within the field of scientific visualization, like interactive particle tracing [KKKW05, SBK06b], volume rendering [EKE01, KKH02, KW03] or isosurface construction [Pas04, KSE04]. While they typically have to get by with the limited memory resources of the graphics card or its host, they provide useful methods for processing data locally on the visualization front-end and, thus, valuable components for the full visualization system.

3. Hybrid Visualization Environments

Today's visualization environments comprise a more complex infrastructure than just a single desktop workstation. They often consist of a visualization front-end, i.e. a dedicated graphics workstation equipped with powerful graphics hardware, which is loosely coupled to some HPC system –

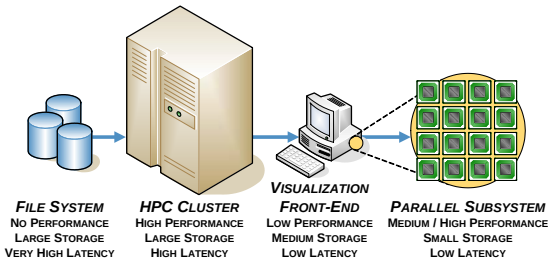

Figure 2: *Components and main data flow in a hybrid visualization environment.*

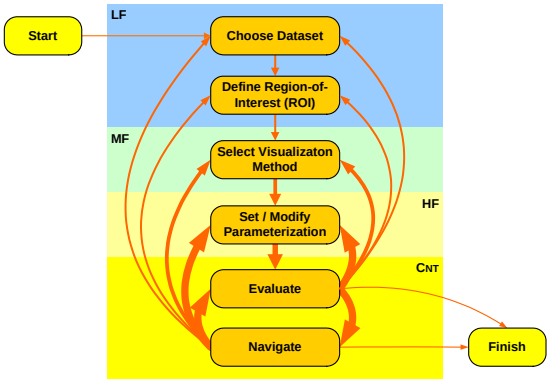

Figure 3: *The exploration process – arrow thickness indicates transition frequency. Subtasks can be classified as low-frequency (LF), medium-frequency (MF), high-frequency (HF), and continuous (Cnt).*

similar to the system, where the data to be analyzed has been generated on. Then again, the HPC back-end is connected to some file system, allowing for the persistent storage of large simulation data. With the advent of multi-core processors and/or programmable GPUs, the front-end can be considered as containing another small-scale parallel computational resource (see Figure 2). The same applies if additional acceleration modules are available, like the IBM Cell BE or the ClearSpeed Advance Accelerator. In the case of a system driving an immersive virtual environment, the front-end itself might be implemented as a distributed system comprising several rendering nodes, as well.

When comparing the capabilities and limitations of the components of such a system, it becomes clear, that the HPC system seems to provide all the necessary features for dealing with large datasets, i.e. high computational power and storage capacity. However, as it is effectively decoupled from the visualization front-end, the connecting network becomes a major bottleneck. Bandwidth and latency issues prevent using this power for direct feedback to the user. In addition, the highly parallel nature of such a system creates additional issues in terms of parameter distribution and result accumulation. On the other hand, while the visualization front-end itself possesses relatively limited capabilities, it can generate immediate feedback to user input, as it provides the user interface and (through the graphics subsystem) image generation. Recent work has demonstrated that programmable graphics hardware yields a high-performance computing resource in its own respect, too. However, compared to the host system or even HPC systems, memory capabilities are quite restricted.

In order to make optimal use of such a hybrid (and highly heterogeneous) visualization environment, the capabilities and limitations of its components have to be considered.

4. The Exploration Process

Contemporary visualization systems still need user input for optimally communicating the results of fluid flow computations. This is typically an iterative process with the user repeatedly specifying the parameterization of single visualization objects until the desired result is achieved. Thus, high interactivity is of prime importance for the acceptance of such a system, i.e. short response times and a quick computation and depiction of results. This applies even more so to VR-based data exploration environments as high latency is not just a major inconvenience but actually renders such a system unusable.

A standard procedure for employing scientific visualization to gain insight into the results of a simulation run consists of the following steps. At first, the dataset to be analyzed is selected, followed by optionally choosing a subsegment of this data. Then, the user chooses a promising visualization method with a suitable initial parameterization from a catalog of available techniques. Upon display of the corresponding results, he starts searching for an ideal visualization according to the current context or problem. This comprises several iterations consisting of adjusting the parameterization, followed by waiting for the results to be displayed, and evaluating these results. As an example, when interactively moving a cut plane through the flow domain, the corresponding data is evaluated as soon as it is presented. The movement stops, when a satisfying result is achieved or a more detailed evaluation is in order. During the whole process, the user constantly navigates within the dataset. When the visualization technique is applied successfully or if satisfactory results are not to be expected, the user might select a different visualization technique, another data segment or even a completely different dataset in order to further deepen his understanding. Note that the execution frequency rises with every step of this process – in more detail (cf. Figure 3):

- *Dataset selection*. Typically, this is done only once. Selecting a new dataset is considered an initiation of an entirely different exploration session.
- *Selection of a segment to be examined in more detail*. This

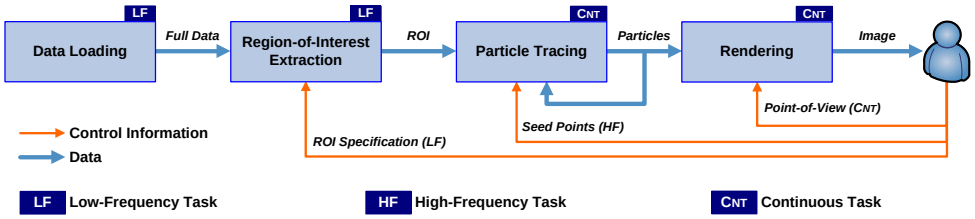

Figure 4: *Visualization pipeline for interactive particle tracing with components labeled according to execution frequency.*

applies if the data domain is very large, the depicted process is very complex, or the user is primarily interested in a subset of the simulation data. As such a segment is typically being analyzed until it is sufficiently understood, this operation is carried out quite infrequently, as well.

- *Selection of a visualization method / visualization object*. Selecting a different tool or creating a new instance thereof might occur quite frequently, i.e. whenever a visualization method has been applied successfully or does not seem to provide the expected insights.
- *(Re-)Parameterization of a visualization object* is performed with quite a high frequency. It depends primarily on the system's response times and, thus, the complexity of the parameterization, the data, and/or the visualization method, as well as the size of the search space.
- *Result evaluation*. As soon as the results of a re-parameterization are displayed, the user analyzes and evaluates them. This happens with at least the same frequency as the re-parameterization as it is directly dependent on its results.
- *Navigation*. During the whole exploration process, the user is continuously navigating within or around the dataset, either actively with an input device or passively via a head tracker inside a virtual environment.

Obviously, a minimum response time is desirable for all operations involved, but its importance rises with its frequency. As an example, a response time of several seconds is tolerable for operations, which are carried out every few minutes, while it is unacceptable for high frequency operations, because they accumulate quite quickly, causing waiting times of several minutes and, thus, resulting in a significant prolongation of the exploration process.

5. The Visualization Pipeline

When using a rather simplified visualization pipeline consisting of filtering, mapping, and rendering, it is difficult to make optimal use of available computational resources of a visualization environment as described in section 3. Thus, we refine the visualization pipeline by breaking up the filtering step into different subtasks based on the exploration process as presented in section 4. Depending on the visualization task at hand, elements of the visualization pipeline are labeled according to their execution frequency (see Figure 4). This information is then used to distribute subtasks onto the visualization system components. As a result, operations, which are computationally very expensive and/or which process large amounts of data are performed on the HPC back-end. On the other hand, operations performed frequently or depending on direct feedback, are executed locally on the visualization front-end, allowing for low response times and high-frequency iterations.

In most cases this requires limiting the amount of data to be processed on the front-end. In accordance with the exploration process model, this can be achieved by limiting local extractions to a user-defined region-of-interest and defining a memory budget to be spent on approximating this data. As an example, specifying a Cartesian grid (i.e., position, size and resolution) and interpolating the full dataset accordingly yield the data for a local analysis. If the user wants to examine another, typically smaller part in more detail, the grid is adjusted appropriately and interpolated upon, again. With decreasing grid size, approximation quality improves and interpolation error decreases.

6. Interactive Exploration in a Virtual Environment

Based on the models from the previous sections, we have implemented a flow exploration system in order to show the validity of this approach. It provides an interactive analysis of large flow datasets inside an immersive virtual environment, thus allowing for utilizing VR methods for gaining insight into flow data of reasonable size. Employing interactive particle tracing and direct volume rendering for visualization, it uses a two-step approach to interactive exploration. In the first step, the user specifies a region-of-interest (ROI), which is then resampled into a Cartesian grid on the HPC back-end, before it is transmitted to the front-end for an interactive analysis in the second step (see Figure 5). Executing the corresponding operations locally on the visualization front-end allows for low-latency feedback for parameter modifications and navigation.

If the ROI itself is modified, e.g. for closer inspection of a segment of the region currently under investigation, the new specification is again sent to the back-end and its result transmitted to the front-end. Typically, the ROI is iteratively scaled down during this process. Maintaining a constant res-

Figure 5: *Analysis of a user-defined region-of-interest (left) via particle tracing (middle) and volume rendering (right).*

olution of the corresponding Cartesian grid results in a continuous reduction of the interpolation error during the ongoing analysis. This procedure fits the common exploration process quite well, as it provides a rough overview over the complete dataset at the beginning with increasing precision towards its end.

VR-based interaction techniques play a major role in facilitating the exploration process. In this case, the ROI is modified using input devices with 6 degrees-of-freedom (DOF). Depending on the employed visualization method, the same applies to its parameterization. In addition, immersive imagery is provided through constant head tracking. Control and data flow within the visualization system are shown in Figure 6.

6.1. The HPC Back-End

The back-end is responsible for providing data to be interactively analyzed on the front-end. Its high computational power and typically superior connection to mass storage are ideal qualifications for this task. A loose coupling to the visualization system, however, is of lesser importance only, as the relative rarity of this operation does not require highly interactive feedback.

Upon startup, the currently active dataset is loaded into memory in order to speed up access times. If the whole dataset does not fit into memory, caching and pre-fetching strategies are to be employed [GHW*04]. Upon ROI definition by the user, the corresponding Cartesian grid is used for interpolating the full dataset and the results are sent back to the visualization front-end. The interpolation process consists of locating the containing cell for every point in the Cartesian grid and interpolating the attributes of the points forming the cell in the original grid.

Using tetrahedral grids as initial data structure, cell search is performed similar to the approach in [SBK06b]. This two-phase approach locates a point close to the query position via a *kd*-tree in the broad phase and finds the containing cell with a short tetrahedral walk in the narrow phase. Depending on the grid structure, this method might fail if the closest point is near a boundary cell potentially causing an early termination of the tetrahedral walk. We alleviate this problem by restarting the tetrahedral walk at additional cells corresponding to points encountered during the *kd*-tree search. While this approach increases the success rate for problematic cases, it results in prolonged search times for points, which lie outside the original grid. However, this approach is still significantly faster than the implementation from the Visualization ToolKit (VTK) [SML06], which we used for comparison.

HPC cluster capabilities are utilized through a hybrid parallelization approach. Mapping time steps to cluster nodes via MPI allows for efficient parallelization even on distributed memory systems. Within a single cluster node, thread-level parallelization with OpenMP reduces wait times even more by capitalizing on multi-core systems. The same applies if the whole HPC cluster consists of a shared memory system. However, in this case optimized MPI implementations using shared memory communication allow for more efficient communication between cluster nodes.

6.2. Interactive Particle Tracing

Once data for the ROI is transmitted to the front-end, interactive particle tracing is used as a local visualization method for intuitive exploration thereof. Two options are offered for controlling the insertion of new particles into the flow domain: direct injection and automated seeders. With direct injection, a 6-DOF input device is used for directly pointing to the position where particles are to be injected. Pulling the trigger releases new particles into the flow field, where they are directly advected and depicted accordingly.

Alternatively, the user can freely position seeder entities inside the flow field. Depending on its mode of operation, a seeder continuously releases particles into the flow field. By configuring particle count per injection and waiting time between two successive injection operations, a variety of effects can be created. In addition, several options are offered for the shape of the seeder. As examples, Figure 1 shows direct seeding via the input device, while Figure 7 shows a selection of box seeders and a line seeder.

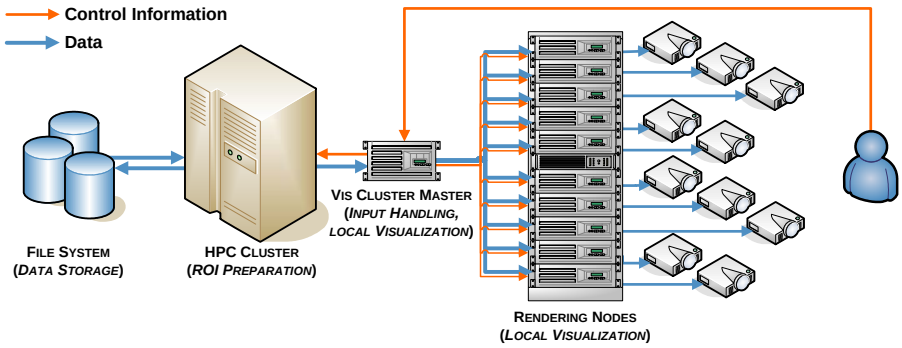

Figure 6: *Data and control flow within a hybrid visualization environment including immersive VR technology.*

In order to achieve real-time feedback even for large numbers of particles, particle tracing and rendering are performed entirely within the graphics subsystem. As is typical for general purpose computations on GPUs, the flow field and particle population are stored as textures in graphics memory. In the case of a time-varying flow field, multiple textures are allocated and used as a ring buffer for storing the time steps which are currently required for flow field interpolation for particle integration. Particle information is stored in two textures which are alternately used as source and destination for the integration process. For integrating particle movement through the flow field, numerical integration schemes of varying precision are implemented in a pixel shader. The user can choose between an Euler integrator and 3rd or 4th order Runge-Kutta integrators. For rendering, a billboard-based approach is used, which allows for generating all necessary geometry within the graphics system, as well. This includes instantaneous particles as well as particle tracers [SBK06a] (see Figure 7).

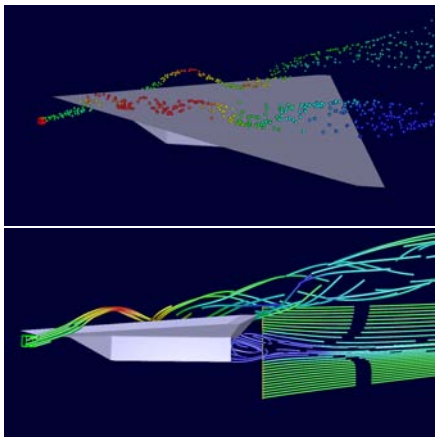

Figure 7: *Flow visualization with interactively seeded instantaneous particles (top) and particle tracers (bottom).*

6.3. Volume Rendering

As an alternative to interactive particle tracing, direct volume rendering is offered as a means for analyzing the contents of the current ROI. Besides navigation, the user can interactively modify the transfer function for increasing insight into the given data. In contrast to the interaction approaches presented above, we do not rely on 6-DOF input devices here. Instead, parameter manipulation is performed through an Ultra-Mobile PC, which is connected to the visualization front-end via WLAN. This allows for considerably more precision during the manipulation of numerical parameters.

6.4. Error Feedback

The resampling of the original flow field into a Cartesian grid introduces some interpolation error. In order to allow for a quality estimation of the displayed visualization results, appropriate feedback is provided to the user in the form of the mean square error of the vector field. Optionally, the local error is stored as a scalar field on the Cartesian grid. This allows for modifying particle color and/or transparency according to local error, e.g. particle opacity is reduced for higher error values, thus hinting at the reduced reliability of the corresponding data. Alternatively, the error field is displayed via volume rendering, informing the user of the spatial error distribution (see Figure 8).

7. Results

We have implemented and tested our approach on a visualization system consisting of a visualization front-end comprising 11 Linux PCs driving a 5-sided immersive projection system. The master node is equipped with dual Intel Xeon processors at 3.06 GHz, the rendering nodes contain Intel Pentium 4 CPUs at 2.8 GHz. Every node is equipped with 4 GB RAM and an NVIDIA GeForce 6800 GT graphics card. The master node is connected to a Sun Fire E2900 with 12 Ultra Sparc IV dual core CPUs at 1.2 GHz holding 48 GB of main memory.

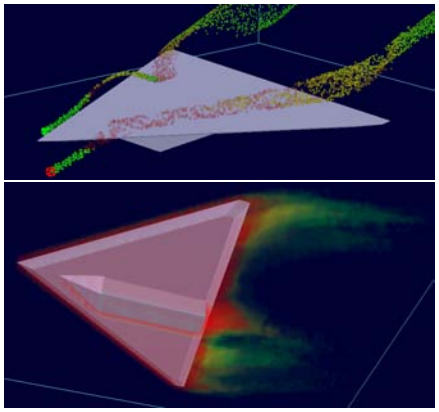

Figure 8: *Interpolation error is depicted via particle color and transparency (top) or volume rendering (bottom).*

This setup allows for an interactive extraction of a user-defined region-of-interest with subsequent local exploration. As an example, we can extract a ROI at 128^3 containing vector and scalar data from a single tetrahedral grid consisting of 12.2 million cells using 4 threads in 1.8 seconds. Increasing the number of threads to 8 reduces the required time to 0.9 seconds. This applies to the point location scheme without the second pass for search failures. Adding a second pass as described in section 6.1 increases the processing time for 4 threads to 3.1 seconds. All in all, including byte order swapping, serialization and transmission over a non-dedicated 100 Mbit/s network results in a total waiting time of about 6 seconds. For time-varying data, multiple processors are used for working on several time steps in parallel, thus amplifying the benefits of parallelization. Depending on time step count and dimensions of the ROI, waiting times between 5 and 30 seconds occur for the tested datasets.

Once the data is transmitted to the front-end, GPU-based particle tracing is used for an interactive exploration of the data at hand. Even when using a 4th order integrator for over 65,000 particles, interactive frame rates can easily be maintained, thus allowing for comfortable interaction within the virtual environment. When accepting reduced rendering quality for particle data, over one million particles can be computed and displayed interactively. In contrast to conventional visualization methods via line-based approaches (e.g., streamlines and path lines), interactive particle tracing yields a considerably better insight into the dynamics of a given flow field, as mapping fluid speed onto particle movement is much more intuitive than mapping it to colors.

When testing our approach for the exploration of large datasets, the achieved results went in line with our initial assumptions, i.e. users spent significant amounts of time for the interactive analysis of local flow phenomena. A modification of the ROI occurred only upon focus shift or focus concentration. As the resulting waiting times were rather infrequent, they were tolerable even when lasting up to several seconds. The majority of time was spent seeding particles and evaluating the currently displayed particle movement while moving around within the dataset.

For direct volume rendering, pixel load is still problematic. As we employ a rather brute-force slicing approach based on 3D textures, a screen resolution of up to 1600x1200 creates some issues. While a data segment can be explored interactively for a limited projected size, frame rates drop to critical regions when the current ROI is strongly enlarged. However, employing more efficient volume rendering techniques as presented in [KW03] should alleviate this problem.

8. Conclusions and Future Work

Based on a novel model of the interactive exploration process and an analysis of contemporary visualization systems, we have extended the visualization pipeline by introducing an explicit, user-defined region-of-interest. A frequency analysis of the computational tasks at hand allows for mapping them onto system components with a focus on interactivity and low latency, i.e. low-frequency tasks are executed on the remote visualization back-end while high-frequency tasks are executed locally on the front-end. An exemplary implementation provided very satisfying results by offering an interactive and very intuitive exploration of even large flow data using interactive particle tracing. To the best of our knowledge, ours is the first solution incorporating high-performance computing and GPU-based flow exploration into a single visualization framework. Compared to previous approaches, this improves interactivity during the analysis of large datasets significantly.

While our approach is targeted at interactive visualization in immersive virtual environments, it can also be applied to desktop-based visualization systems. However, user interaction tends to suffer from applying a 2D interface to the exploration of 3D to 4D data.

The next steps include an extension of our approach towards additional exploration methods, e.g. an interactive positioning of cut planes inside a resampled ROI and concurrent calculation of a precise solution on the back-end. In order to improve ROI extraction response times for time-varying data, a time step prioritization scheme similar to [WHS*06] will be incorporated. In addition, we will experiment with streaming approaches in order to support virtually infinite time sequences.

Although there are still some issues to be addressed, the presented approach shows promising results and acts as a beneficial starting point for further research.

Acknowledgements

The authors would like to thank the Institute of Aerodynamics (AIA) at RWTH Aachen University, the German

Aerospace Center (DLR) in Göttingen, and Drs. Koomullil and Soni, University of Alabama at Birmingham, for the datasets kindly made available.

References

[BGY92] BRYSON S., GERALD-YAMASAKI M.: The Distributed Virtual Windtunnel. In *Proceedings of Supercomputing '92* (1992), pp. 275–284.

[BJ96] BRYSON S., JOHAN S.: Time Management, Simultaneity and Time-Critical Computation in Interactive Unsteady Visualization Environments. In *Proceedings of IEEE Visualization '96* (1996), pp. 255–261.

[BL91] BRYSON S., LEVIT C.: The Virtual Windtunnel: An Environment for the Exploration of Three-Dimensional Unsteady Flows. In *Proceedings of IEEE Visualization '91* (1991), pp. 17–24.

[EKE01] ENGEL K., KRAUS M., ERTL T.: High-Quality Pre-Integrated Volume Rendering Using Hardware-Accelerated Pixel Shading. In *Proceedings of Eurographics/SIGGRAPH Graphics Hardware Workshop* (2001), pp. 9–16.

[GHW*04] GERNDT A., HENTSCHEL B., WOLTER M., KUHLEN T., BISCHOF C.: Viracocha: An Efficient Parallelization Framework for Large-Scale CFD Post-Processing in Virtual Environments. In *Proceedings of Supercomputing '04* (2004).

[HHN*02] HUMPHREYS G., HOUSTON M., NG R., FRANK R., AHERN S., KIRCHNER P. D., KLOSOWSKI J. T.: Chromium: A Stream-Processing Framework for Interactive Rendering on Clusters. In *Proceedings of SIGGRAPH '02* (2002), pp. 693–702.

[JKMG02] JANKUN-KELLY T., MA K.-L., GERTZ M.: A Model for the Visualization Exploration Process. In *Proceedings of IEEE Visualization 2002* (2002), pp. 323–330.

[KKH02] KNISS J., KINDLMANN G. L., HANSEN C. D.: Multidimensional Transfer Functions for Interactive Volume Rendering. *IEEE Transactions on Visualization and Computer Graphics 8*, 3 (2002), 270–285.

[KKKW05] KRÜGER J., KIPFER P., KONDRATIEVA P., WESTERMANN R.: A Particle System for Interactive Visualization of 3D Flows. *IEEE Transactions on Visualization and Computer Graphics 11*, 6 (2005), 744–756.

[KSE04] KLEIN T., STEGMAIER S., ERTL T.: Hardware-Accelerated Reconstruction of Polygonal Isosurface Representations on Unstructured Grids. In *Proceedings of the 12th Pacific Conference on Computer Graphics and Applications (PG '04)* (2004), pp. 186–195.

[KW03] KRÜGER J., WESTERMANN R.: Acceleration Techniques for GPU-Based Volume Rendering. In *Proceedings of IEEE Visualization 2003* (2003).

[OLG*07] OWENS J. D., LUEBKE D., GOVINDARAJU N., HARRIS M., KRÜGER J., LEFOHN A. E., PURCELL T. J.: A Survey of General-Purpose Computation on Graphics Hardware. *Computer Graphics Forum 26*, 1 (2007), 80–113.

[Pas04] PASCUCCI V.: Isosurface Computation Made Simple: Hardware Acceleration, Adaptive Refinement and Tetrahedral Stripping. In *Proceedings of Joint Eurographics – IEEE TVCG Symposium on Visualization (VisSym) 2004* (2004), pp. 293–300.

[RFL*98] RANTZAU D., FRANK K., LANG U., RAINER D., WÖSSNER U.: COVISE in the CUBE: An Environment for Analyzing Large and Complex Simulation Data. In *Proceedings, 2nd Workshop on Immersive Projetction Technology (IPT'98), Ames, IA* (1998).

[RR06] ROTH M., REINERS D.: Sorted Pipeline Image Composition. In *Proceedings of Eurographics Symposium on Parallel Graphics and Visualization (EGPGV) 2006* (2006), pp. 119–126.

[SBK06a] SCHIRSKI M., BISCHOF C., KUHLEN T.: Exploring Flow Fields with GPU-Based Stream Tracers in Virtual Environments. In *Proceedings of Eurographics 2006 (Short Papers)* (2006), pp. 115–118.

[SBK06b] SCHIRSKI M., BISCHOF C., KUHLEN T.: Interactive Particle Tracing on Tetrahedral Grids Using the GPU. In *Proceedings of Vision, Modeling, and Visualization (VMV) 2006* (2006), pp. 153–160.

[SBM92] SPRINGMEYER R. R., BLATTNER M. M., MAX N. L.: A Characterization of the Scientific Data Analysis Process. In *Proceedings of IEEE Visualization '92* (1992), pp. 235–242.

[SML06] SCHROEDER W., MARTIN K., LORENSEN W.: *The Visualization Toolkit - An Object-Oriented Approach to 3D Graphics*, 4 ed. Kitware, Inc., 2006.

[SMW*05] STRENGERT M., MAGALLÓN M., WEISKOPF D., GUTHE S., ERTL T.: Large Volume Visualization of Compressed Time-Dependent Datasets on GPU Clusters. *Parallel Computing 31*, 2 (2005), 205–219.

[TM04] TORY M., MÖLLER T.: Human Factors in Visualization Research. *IEEE Transactions on Visualization and Computer Graphics 10*, 1 (2004), 72–84.

[UJK*89] UPSON C., JR. T. F., KAMINS D., LAIDLAW D. H., SCHLEGEL D., VROOM J., GURWITZ R., VAN DAM A.: The Application Visualization System: A Computational Environment for Scientific Visualization. *IEEE Computer Graphics and Applications 9*, 4 (1989), 30–42.

[WHS*06] WOLTER M., HENTSCHEL B., SCHIRSKI M., GERNDT A., KUHLEN T.: Time Step Prioritising in Parallel Feature Extraction on Unsteady Simulation Data. In *Proceedings of Eurographics Symposium on Parallel Graphics and Visualization (EGPGV) 2006* (2006), pp. 91–98.