Novel Volume-Visualization Methods for the Interactive Rendering of CFD Simulation Data

Philipp Muigg*, Helmut Doleisch*, Markus Hadwiger*, Eduard Gröller+

*VRVis Research Center, Vienna, Austria
mailto: {muigg|doleisch|hadwiger}@vrvis.at

+Institute of Computer Graphics and Algorithms, Vienna University of Technology, Austria
mailto: groeller@cg.tuwien.ac.at

Summary:

Spatial grid data structures, which are the foundation for modern CFD simulations, have become more flexible allowing for efficient simulation techniques but also making the result visualization more difficult. Whereas initially only tetrahedral or block structured grids have been used nowadays simulation tools (e.g. Fluent, Star CD, etc.) use grids which decompose the simulated domain into different types of convex polyhedra. Especially when performing volume rendering most state of the art approaches perform a tetrahedral subdivision of these more complex cells. This can result in ambiguous visualizations since different tetrahedralizations of the cells can be applied. Additionally the number of cells which have to be visualized can increase drastically. We propose a (graphics hardware based) volume rendering approach which copes with large unstructured polyhedral grids in several novel ways:

- we perform volume rendering on the original grid (no tetrahedralization is necessary)
- we subdivide the grid in order to visualize each part according to its contents and visibility
- we support multiple rendering modes in order to emphasize different aspects of the data
- we perform resampling of unimportant regions of the volume onto a low resolution structured grid in order to retain interactivity even on standard PC hardware

In order to distinguish between important and unimportant regions of the volume a so called Degree of Interest (DOI) volume is used, which can be specified in the SimVis framework into which our approach has been integrated. We demonstrate the effectiveness of our method by using it to visualize two different CFD simulation cases.

Keywords:

Volume Visualization, Unstructured Grid Raycasting, Focus+Context Techniques

Note: A colored PDF file of this paper is available at http://www.simvis.at/downloads/papers/NAFEMS-CFD-2008.pdf

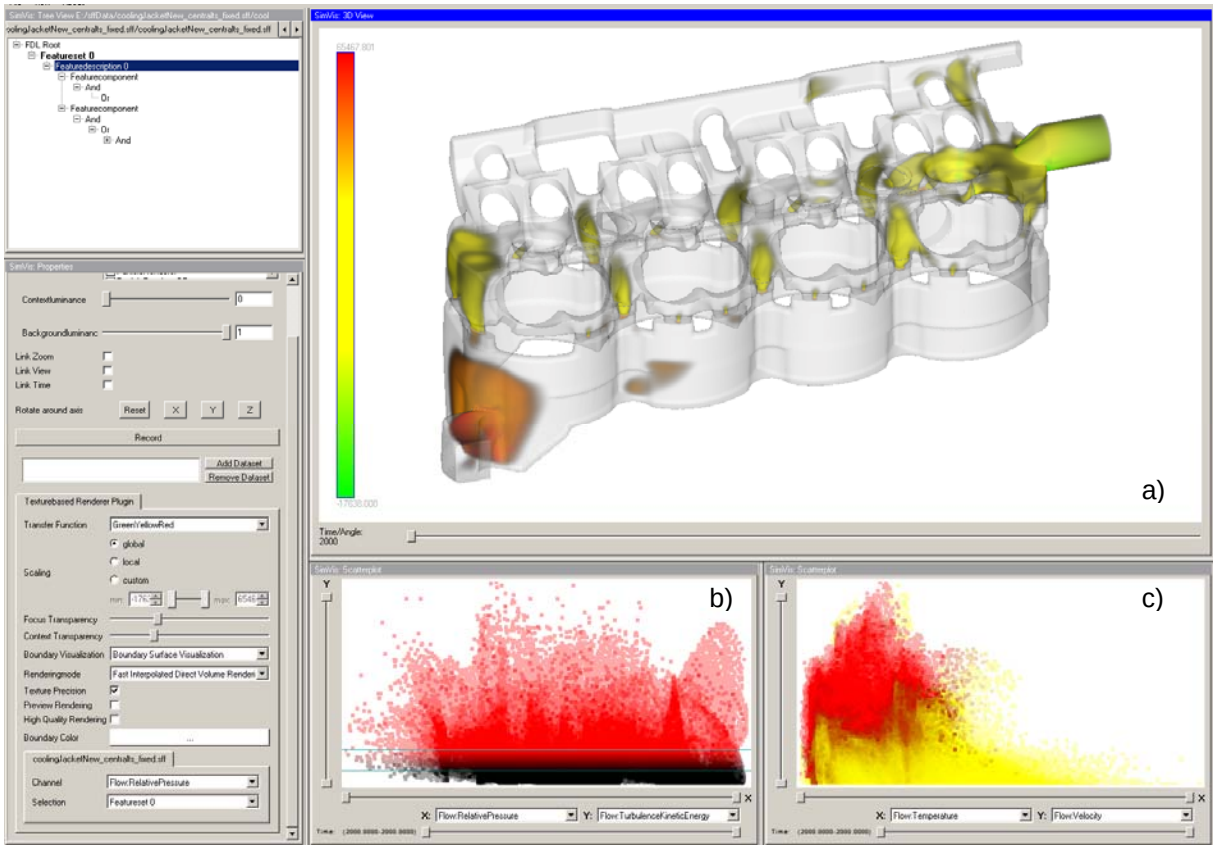

Fig. 1: This figure shows the volume rendering of a cooling jacket dataset (a) containing roughly 1.6 million cells and two scatterplots showing pressure versus turbulent kinetic energy (b) and temperature versus flow velocity (c). Cells containing high turbulent kinetic energy have been selected in scatterplot (b) and are highlighted in the second scatterplot (c) as well as in our volume rendering implementation (a). Here color is mapped to pressure.

1 Introduction

Unstructured grids are an important volumetric representation that is especially common in the field of computational fluid dynamics (CFD), e.g., simulations of engineering problems computed with finite volume methods. Real-world simulation grids are often comprised of a variety of cell types, such as tetrahedra, hexahedra, octahedra, pyramids, and prisms. Besides standard techniques, such as slice planes or iso surfacing to visualize this kind of data, volume rendering provides a more thorough overview over such data. Providing interactive volume rendering of unstructured data has become feasible only recently with the evolution of modern graphics hardware which nowadays provides enormous parallel processing capabilities.

The four main approaches for GPU-based volume rendering of unstructured grids are cell projection via the projected tetrahedra algorithm [19], raycasting [25], resampling into a structured grid and rendering this grid instead [29], and point-based approaches [30]. One of the most important problems when rendering unstructured grids is obtaining a correct visibility order. Cell projection and point-based approaches require explicit visibility sorting, which is a major bottleneck of these methods. Raycasting implicitly establishes correct visibility without explicit cell sorting. This property is a huge advantage of raycasting approaches and makes them competitive with cell projection when for the latter not only the time for projection and rendering but also for sorting is considered. Our framework employs raycasting as basic rendering method, but in contrast to similar approaches does not require the entire grid to be resident in GPU memory. This is due to our volume subdivision method.

Most volume rendering approaches convert general unstructured grids into tetrahedral grids in a pre-processing stage and handle only tetrahedra during actual rendering. This subdivision of more complex cells into linear subcells, however, prevents interpolation of a C1-continuous function within non-tetrahedral cells. Thus, it is desirable for many real-world applications to avoid tetrahedralization for rendering. Another challenge for rendering is the number of cells, which can easily be several hundred thousands to millions of cells. Avoiding tetrahedralization reduces the number of cells that need to be rendered significantly. Additionally most approaches visualize only one scalar data quantity. Especially when dealing with high dimensional simulation data (e.g., from combustion

simulations) this is hardly sufficient if interactions between multiple data attributes have to be visualized.
Thus the SimVis framework, into which our volume rendering method has been incorporated, provides a so called Degree of Interest (DOI) function. This additional scalar volume containing values from [0..1] represents the user interest for each cell. It can be specified interactively in multiple linked views and can be based on multiple data attributes (see Figure 1). During the volume rendering process different rendering modes use this DOI volume in combination with a scalar data attribute in order to create the final visualization. Additionally to the visual representation the DOI function is used to discriminate between important (focus) and unimportant (context) regions of the dataset. We propose to decompose the simulated volume into sub-volumes (bricks) and treat them based on their contents. Bricks containing mainly context regions are visualized using only rough approximations of the data whereas focus regions are rendered using a high quality representation. This allows for interactive frame rates even when dealing with large datasets.

2 Related Work

Most object-order methods for rendering unstructured grids are based on the Projected Tetrahedra (PT) algorithm [19], which samples only cell faces. Depending on graphics hardware features, the volume rendering integral can be solved using improved accuracy [20], with the best results generally achieved by pre-integration [17]. Almost all cell projection approaches are constrained to tetrahedra, although the extension to polyhedra is possible using approximations [14]. However, a major performance bottleneck of all PT variants is the need for explicit visibility sorting. A variety of powerful sorting approaches have been developed [20], including hybrid CPU/GPU methods [2]. An explicit sorting step is also required by most point-based methods for rendering unstructured grids [30]. Point-sampling strategies have also been employed successfully for simplification of very large unstructured grids [22]. However, the sorting step required by all object-order methods can only be neglected when commutative blending modes such as purely emissive volumes [26] are used. In contrast, image-order approaches such as raycasting [6, 25] are usually slower in pure rendering performance than cell-projection techniques, but compensate for this fact by the lack of an explicit sorting step. They are also very flexible, e.g., with respect to adaptive sampling, and are a natural choice when complex non-linear interpolation techniques such as mean value coordinates [7] are desired. For structured grids, GPU raycasting approaches have also been shown to work very well [6, 10]. In order to tackle very large volumes, many approaches employ a hierarchical subdivision, e.g., octrees for a multiresolution representation and rendering of structured grids [1, 11, 28]. Bricking approaches such as these are usually only used for structured data, whereas for unstructured grids, cells are only roughly assigned to nodes in a hierarchy for acceleration [15]. In contrast, our approach for bricking unstructured data for memory management and hybrid rendering is very similar to bricking of structured data. Another possibility to render unstructured grids is to resample them into a structured representation for rendering, which can be performed hierarchically [13]. Powerful resampling algorithms have been developed, especially in order to leverage the rasterization power of GPUs [24, 29]. Many cell sorting and also raycasting techniques require non-convex grids to be convexified [8, 16] before they can be rendered. Our approach circumvents this by using a depth peeling approach [5, 27].
The hybrid renderer discussed in this paper has been implemented as a plugin for the SimVis system [3]. SimVis employs multiple linked views (Figure 1) for concurrently showing, exploring, and analyzing different aspects of multi-variate data. 3D views can be used to visualize features that are specified interactively in several types of attribute views, e.g., scatterplots or histograms. For this feature specification, the user visually inspects specific attributes in order to gain insight into the selected relations in the data. The interesting subsets are then brushed interactively, the result of which is integrated with the data volume in the form of an additional degree of interest (DOI) volume. This DOI attribution is used in all views of the analysis setup to visually discriminate the specified features from the rest of the data in a focus+context visualization style that is consistent in all linked views. Focus+context approaches are very well suited for showing the user the actual parts of interest in large data sets [9, 23]. The SimVis system and the hybrid raycaster support smooth brushing [4], i.e., fractional DOI values, as well as the logical combination of brushes for the specification of complex features based on multiple data attributes and derived information [3].

3 Rendering Pipeline

The rendering pipeline which is the basis for our volume rendering approach consists of three stages as shown in Figure 2. The volume subdivision (see section 3.1) is a pre processing stage which has to be invoked only once at the beginning of a visualization session. It creates a hierarchical subdivision of the simulated domain into individual bricks and prepares cells which are intersected by brick boundaries to be clipped in subsequent stages.
The second stage (see section 3.2) classifies all bricks based on the contained DOI volume and available memory on the graphics card. Bricks for which the DOI function is constantly zero are

classified as "empty" which implies that no volumetric representation has to be stored. As long as sufficient memory is available bricks which are not "empty" are considered "unstructured". If a previously defined memory threshold is reached all subsequent bricks are considered "structured". For those bricks resampling onto a low resolution structured grid is performed.
The final stage (see section 3.3) performs the actual visualization by doing volume rendering for the "unstructured" and "structured" bricks and transparent surface rendering for the "empty" bricks.

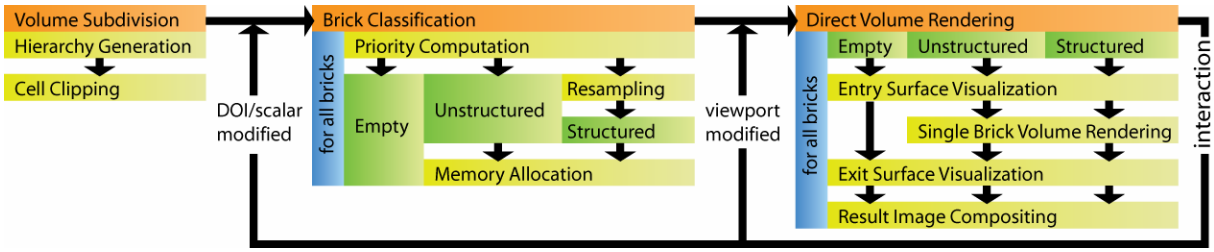

Fig. 2: This figure shows the rendering pipeline used by our method. First the simulated domain is decomposed into bricks in the volume subdivision stage which are classified based on the contained DOI volume. The three classes are "empty" for bricks in which the DOI function is constantly zero, "structured" for bricks which can be resampled onto a low quality representation and "unstructured" for bricks which should be visualized using the original data. After this the volume rendering of each brick is performed based on its classification.

3.1 Volume Subdivision

An important advantage of our volume subdivision scheme is that it allows to apply optimizations from structured volume rendering to all brick types uniformly, including unstructured bricks. Each brick is a leaf node of a kD-tree, which is used here because it provides better control over the number of entries in its child nodes than similar data structures such as octrees. For each brick, view frustum culling is performed, and empty bricks are skipped entirely (by the volume renderer). This helps with distributing rendering and memory resources only among visible parts of the data. Besides this, rendering individual bricks enables efficiently dealing with highly non-convex meshes. Many datasets from the engineering field have very complex surface geometries (e.g., Figure 1 (a)), which would require a high number of rendering passes when treated as one entity. Our bricking scheme greatly reduces the overall number of depth peels, because each brick is peeled independently. The contents of each brick are clipped against the brick boundary in order to guarantee correct compositing of brick contributions during raycasting. The depth of the kD-tree is determined by limiting the number of cells and vertices within each brick to 64K, which allows for memory savings when storing unstructured grid topology. If time-varying meshes have to be visualized, the brick decomposition is performed for every timestep, except for time-dependent data specified on static grids.

3.2 Brick Classification

The bricking scheme we employ facilitates distribution of processing and memory resources such that important portions of the dataset are favored over less interesting regions, which is done based on each brick's importance. In order to determine the importance of a brick, we combine three simple measures that can be evaluated efficiently. The first measure is the average DOI value within the brick, which represents the user's interest in its contents. The second and third measure are the entropy of the histograms of the DOI and the scalar field within the brick, respectively, in order to reflect its information content. All three measures result in values in [0..1] and are weighted equally. The individual bricks are then sorted based on their importance, and texture memory resources are distributed in this order. The original unstructured representation of a brick is used as long as enough memory is available, while for all other bricks a small 3D texture is allocated and filled on-demand by resampling the unstructured data. Since empty bricks do not need any volumetric representation, they are ignored by the memory management system.

3.3 Direct Volume Rendering

Our volume rendering approach is based on raycasting which means that for each image pixel a view ray is traced through the simulated volume which is treated as a semi transparent medium. Opacity and color are determined by a transfer function based on the DOI and the scalar data volumes. The final visualization is created by combining the contribution from each brick in image space in front-to-back visibility order. Since bricks are represented differently based on their classification we have implemented a hybrid rendering approach which consistently combines surface visualizations (at the ray entry and exit points through the simulated domain for all three brick classifications), unstructured

grid volume rendering (for bricks classified as "unstructured") and structured grid volume rendering (for bricks classified as "structured"). Figure 3 shows the traversal of two viewing rays through a "structured" and an "unstructured" brick. Along each ray the volumes are sampled by using different interpolation techniques. In the structured case simple trilinear interpolation is used whereas barycentric interpolation at cell faces (which are triangulated on the fly) and mean value interpolation in the interior of cells is used in the unstructured brick. In order to increase the sampling resolution we interpolate linearly between barycentric and mean value samples.
Since most meshes used in CFD applications are concave a view ray can enter and exit the simulated domain multiple times. We address this by using an approach proposed by Weiler et al. [27] for unstructured bricks which is similar to depth peeling [5]. In order to properly visualize the mesh boundary even for structured bricks we have adapted this method by performing depth peeling for entry and exit faces in parallel and performing raycasting between them. Again the results from each depth layer are combined in image space in front to back visibility order. For empty bricks no volume rendering is performed at all. Instead depth peeling is used to allow for transparent surface rendering without the necessity to sort the surface polygons.
Interactivity is highly important during an analysis session. Thus several speed/image quality tradeoffs can be made in order to allow for fast response time during interaction and high image quality when only a still image is required. First of all the result image resolution can be reduced on the fly allowing for large performance gains. Additionally the mean value interpolated samples can be skipped and the sampling distance can be increased along a view ray in unstructured bricks. In order to increase image quality our approach can render the whole dataset in its original unstructured representation even if the amount of available graphics memory is not sufficient to store all unstructured data. This is made possible by swapping individual bricks out of and into GPU memory during the rendering process.

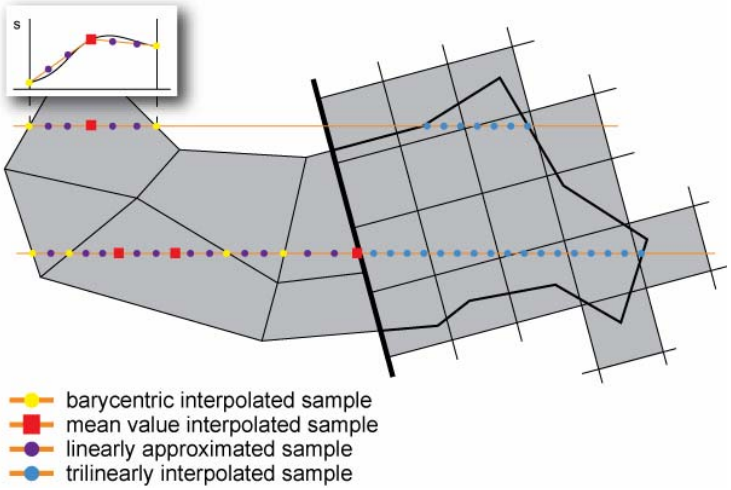

Fig. 3: During volume rendering view rays (orange) are traced through the dataset volume. Along these rays the DOI volume as well as a scalar volume (such as temperature) are sampled, transformed into opacity and color values and accumulated. Within the different brick representations (“structured” and “unstructured”) different interpolation methods are used.

4 Rendering Modes

The hybrid raycasting method proposed in this paper has to deal with two different data volumes, one being the DOI function, which has been specified by the user and the other one being a scalar field of the underlying dataset. In order to visualize this multi-volume several rendering modes can be selected by the user by specifying a boundary visualization technique and an optical volume model. Figure 4 shows four different optical models ((a), (d), (e), and (f)) and three boundary visualization techniques ((a), (b), and (c)) which can be combined arbitrarily. The upper three images present standard direct volume rendering (DVR) in combination with a silhouette enhancement technique (a), shaded backface (b) and surface rendering (c). The lower row uses silhouette rendering in combination with shaded cell face (d), smooth iso surface (e), and iso surface (f) rendering.
The transfer function which is used by most optical models to translate sampled DOI and attribute values into color and opacity values is based on the following equations:

$$c(\vec{x}) = \lambda(\vec{x}) tf\left(s_{win}(\vec{x})\right) + \left(1 - \lambda(\vec{x})\right)\vec{c}_l,$$

$$\alpha(\vec{x}) = \lambda(\vec{x})\, f_\alpha$$

Here $c(\vec{x})$ is the color and $\alpha(\vec{x})$ the transparency at sample position $\vec{x}$. $\lambda(\vec{x})$ is a blending factor from the unit interval. $tf(s)$ is a simple 1D transfer function which maps the unit interval to a color. We use a windowed lookup function $s_{win}(\vec{x})$ to sample a scalar data attribute and map a specified interval in its attribute space (such as temperatures from 280 to 290 Kelvin) onto the unit interval [0..1]. The resulting color $c(\vec{x})$ is a linear combination of the transfer function lookup and the context luminance $\vec{c}_l$ which means that the blend factor $\lambda(\vec{x})$ determines the saturation. The result transparency $\alpha(\vec{x})$ is also influenced by $\lambda(\vec{x})$ and the focus transparency f_α. All parameters such as $\vec{c}_l$ and f_α, the interval used for windowing and the transfer function can be specified by the user.

The definition of $\lambda(\vec{x})$ depends on the optical model. For standard DVR and shaded cell face rendering we use $\lambda(\vec{x}) = DOI(\vec{x})$ with $DOI(\vec{x})$ being the degree of interest value at position $\vec{x}$. In the case of smooth iso surface rendering $\lambda(\vec{x}) = (1 - 2abs(s_{win}(\vec{x}) - 0.5))DOI(\vec{x})$ is used, which limits the visualization to portions of the data for which the DOI is greater than zero and the scalar attribute is within the windowing interval. In order to realize the optical model seen in Figure 4 (d) we use only one sample per cell and add additional shading based on the cell face through which a view ray enters.

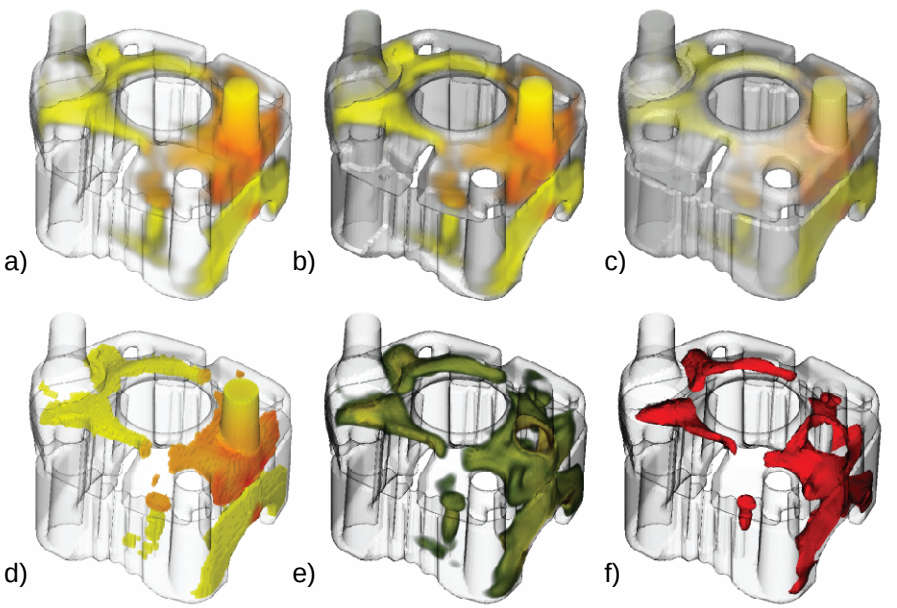

Fig. 4: The top row of shows three different boundary visualization techniques (from left to right): silhouette rendering, shaded backface rendering and surface rendering. All three have been combined with standard direct volume rendering (DVR). The lower row presents three additional volume models (from left to right): shaded cell face rendering, smooth iso surfaces and iso surfaces.

5 Application Results

In this section some example applications of our volume rendering approach in conjunction with the SimVis system are presented. Two very different datasets are used to demonstrate the power of our visualization method.

5.1 Large Static Data

The dataset presented in this section contains the simulation of a cooling jacket for a four cylinder diesel engine. Figure 5 (a) shows a small overview over the simulated domain which is split into two regions, the cylinder block indicated in red containing the inlet and the cylinder head indicated in yellow containing the outlet (red indicates high yellow lower pressure values). Both parts are connected by small gaskets which can be used to steer the flow of the coolant by varying their sizes. In order to identify critical regions where cooling of the engine might be insufficient we have used the feature definition capabilities of the SimVis framework. Figure 5 (b) shows an overview over regions which exhibit slow flow velocity and high temperature which indicates regions where the cooling fluid is stagnating and heating up. In order to look inside the volume without loosing spatial context information we use the shaded backface rendering mode to visualize the surface of the dataset. Since

only small portions of the dataset exhibit near critical or critical conditions we use the cell face volume model which displays the selection on a cell-by-cell basis without applying any interpolation. Three example regions have been selectively enlarged in Figures 5 (c), (d) and (e). Our volume rendering approach can be combined with an arbitrary number of visualization techniques which display data using opaque primitives. Thus it is possible to mix a stream line visualization (one streamline is seeded for each cell in the selected region) with our volume rendering results. In all three images vortical flow along the surface of the simulated domain can be observed which is probably the cause for the higher temperatures.

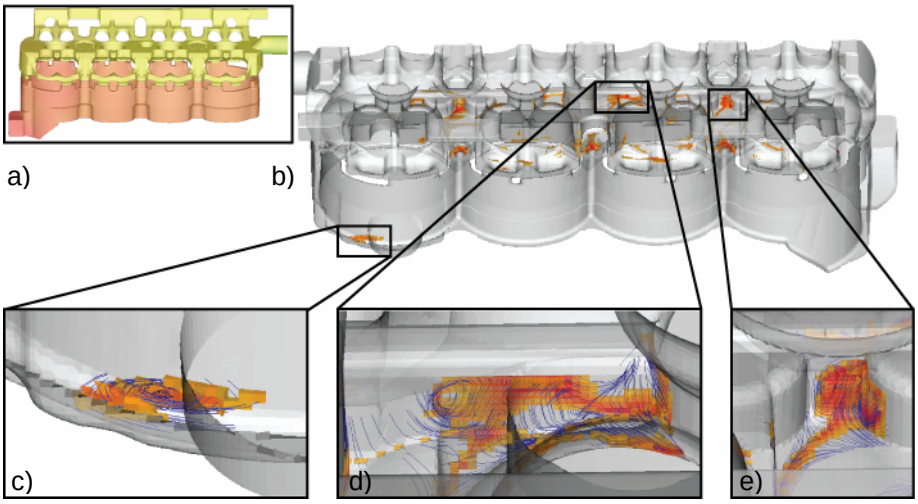

Fig. 5: This figure shows the interior of a cooling jacket used for four cylinder diesel engines. We have selected regions of slow flow velocity and high temperature in order to locate critical areas within the dataset. The three zoom-ins show interesting regions in combination with a stream line visualization which is blended with our volume rendering approach. Note that in all three cases vortical structures can be identified.

5.2 Time Dependent Data

The Two Stroke Engine dataset contains the simulation of a loop scavanged two stroke engine for which the fuel injection and combustion process has been modeled. An in depth analysis of this dataset has been performed by Schmidt et al. [18]. They compare two different fuel injection pressures (in the simulated engine gasoline direct injection is used) with respect to the fuel distribution within the combustion chamber at the ignition time and the loss of combustable mixture through the exhaust. Since the flow in the intake (through which fresh air is sucked into the combustion chamber, if the intake ducts are not blocked by the piston) and exhaust ducts is essential to the functioning of the engine those parts have been modeled additionally to the combustion chamber as shown in Figure 6 (top left). The machinery modeled in the Two Stroke Engine dataset is not static (the piston is moving up and down) which means that the underlying volume mesh changes over the simulated time span: vertex positions are modified and the overall mesh topology changes in order to account for the degeneration of cells. The fuel injection itself starts at 165° crank angle (ca) whereas the ignition is performed at 345° ca. The scalar values of highest importance in this dataset are the equivalence ratio, which is the ratio between fuel and air, and the reaction progress variable, which represents the progress of the combustion (zero representing unburnt and one fully burnt mixture). Figure 6 shows a selection of optimal equivalence ratio (from 0.7 to 1.4) which is additionally restricted by marking only portions of the data, where no burning has occurred (reaction progress variable equaling zero). In order to account for the smooth nature of the underlying flow features (equivalence ratio just above 1.4 or below 0.7 not instantly implies not burnable) smooth boundaries have been chosen for both selections. In Figure 6 different timesteps, denoted by the current crank angle (ca), of these selections are shown. At ca = 190° the gasoline injection is nearly completed and the air intake ducts and exhaust system opening are being closed by the piston. Successively some of the gasoline starts leaking through the exhaust as shown at ca = 215°. Here only portions of a lean mixture are lost which is indicated by the green color. At ca = 240° some gasoline is being sucked into the intake system, which will be scavenged in the next combustion cycle. Additionally portions of very rich mixture (indicated by red) move down towards the exhaust system opening and are sucked out just before it is closed at ca = 265°. The final image taken at ca = 345° depicts the gasoline distribution shortly after the ignition, showing that the ignition spark itself is located well within a region of good equivalence ratio (indicated by yellow). Additionally it can be noted that the overall distribution of burnable mixture

within the combustion chamber is highly uneven leading to higher emissions and lower efficiency in this case (Schmidt et al. [18] have shown that higher injection pressure can lead to a more homogenous gasoline distribution within the combustion chamber). It is obvious that especially for simulations like this, the proper visualization of mesh boundaries is of very high importance since they indicate the moving parts of machinery which strongly influence complex flow conditions within the simulated domain.

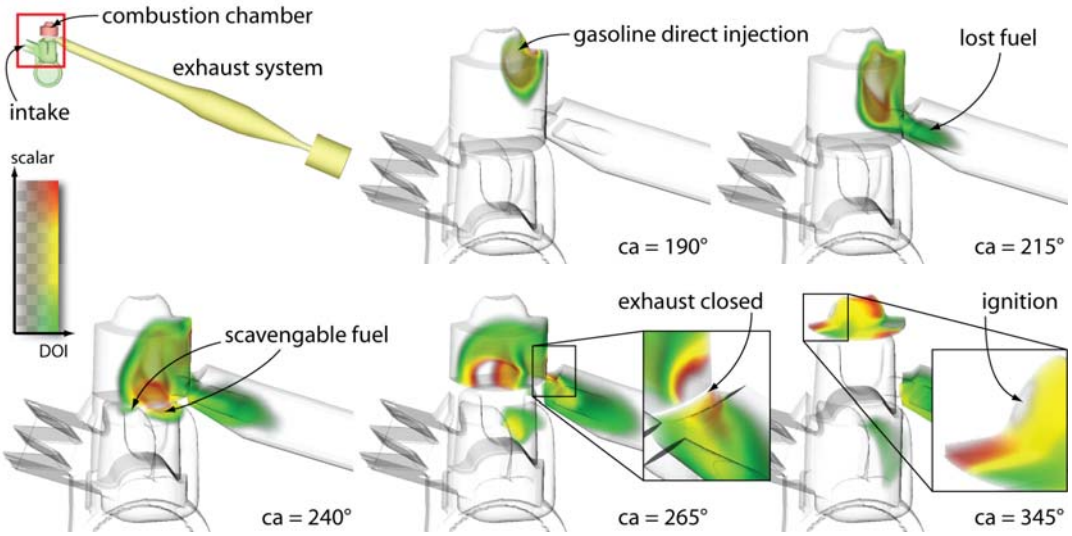

Fig. 6: This figure shows multiple timesteps of the Two Stroke Engine dataset. The selected regions represent unburnt fuel exhibiting an equivalence ratio between 0.7 and 1.4. Standard DVR is used in combination with silhouette boundary rendering to create these visualizations.

5.3 Performance Characteristics

Several parameters can be used to steer the memory consumption, the speed, and the image quality of the presented visualization approach. In order to guarantee responsiveness during an analysis and exploration session two different sets of parameter settings can be chosen: one is applied during interaction, and one is used for generating static images. Additionally, it is possible to progressively update the display during the generation of a static image (every n'th brick the accumulated color and opacity information is copied to the front buffer). In Table 1 some performance measures are presented for the different datasets shown in this paper. Different quality settings have been used to measure rendering performance during interaction, for a static image and for the dataset without using resampling (here, an unstructured brick always forces the memory manager to allocate texture memory, even if data from another brick has to be overwritten). During interaction only one depth peel for every brick is rendered, and the unstructured brick sampling settings are configured to perform only barycentric interpolation at the cell boundaries. Additionally the result image resolution is reduced to 256^2 (instead of 512^2 for the static measurements). In the static case a maximum of four mean value samples are computed per cell per ray. These settings are equal to those used to create the "Presentation" column of Table 1 resulting in the same measures for datasets which completely fit into the texture memory allocated for unstructured data. For comparison with a fast state-of-the-art tetrahedral volume renderer, the last column of Table 1 shows the performance of the publicly available implementation of the HAVS algorithm [2] for our datasets converted to tetrahedral grids. It can be observed that HAVS is more than five times faster than our approach when generating high-quality images for the Small Cooling Jacket. However, the time required by visibility sorting increases at least linearly with the number of cells (172ms for the Two Stroke Engine and 2.5s for the Large Cooling Jacket), which is probably the main factor why our raycasting method outperforms HAVS for the Large Cooling Jacket dataset even under "Presentation" quality settings that cause streaming of texture data onto the GPU during rendering.
All tests have been carried out on an AthlonX2 4400+ with 4GB of RAM, and a GeForce 8800GTX with 768MB video memory.

Dataset	# Cells	#Tets	#Bricks	Interact.	Static	Pres.	HAVS
Large Cooling Jacket	1,536K	7,149K	60(10)	234ms	1s	2.4s	6s
Two Stroke Engine	150K	701K	8(0)	50ms	153ms	153ms	297ms
Small Cooling Jacket	77K	377K	2(0)	53ms	784ms	784ms	156ms

Tab. 1: Various datasets used throughout this paper. The overall number of cells in the datasets as well as the equivalent number of tetrahedral cells using a minimal subdivision is given for comparison. The overall number of bricks is shown along the number of resampled bricks (in brackets). The three timings use the quality settings “Interaction”, “Static” and “Presentation”. For comparison purposes, performance numbers using the public version of HAVS [2] are also given for the tetrahedralized datasets (using the command line options “-none -p”).

6 Conclusions and Future Work

We have presented a fast end efficient volume rendering approach which utilizes the processing power of modern graphics hardware to provide interactive framerates even for large datasets (currently datasets with up to 7 million cells of mixed type are supported). By subdividing the volume into individual bricks and using the feature specification capabilities of the SimVis framework we provide the user with means to focus his interest on important regions of the dataset. These regions can be rendered using our high quality volume rendering approach whereas unimportant areas are visualized using fewer resources. We provide a large variety of visualization techniques for the dataset surface and the simulated volume which can be combined freely into a huge number of rendering modes.

There are various interesting areas for future work. The resampling approach currently used to convert unstructured data into a structured representation can be enhanced greatly by utilizing the memory bandwidth and processing capabilities of nowadays graphics hardware. Furthermore high quality resampling can be used to increase the image quality of our algorithm if very large meshes, containing cells smaller than individual image pixels, should be visualized. Besides improvements to our resampling approach the memory requirements for unstructured grids can be reduced greatly by refining our current data structure storing the grid topology. This would allow for more efficient ray traversal through the unstructured volume and additionally enable our implementation to cope with even larger datasets.

7 Acknowledgements

We thank Stephan Schmidt and Oliver Schögl for their support during this work. The datasets are courtesy of the Institute for Internal Combustion Engines and Thermodynamics, University of Technology Graz, Austria and AVL List GmbH, Graz, Austria. This work has been partly funded by the Bridge funding program of the Austrian Funding Agency (FFG) in the scope of the MULSIMVIS project (Nr. 812106).

8 References

[1] Boada I., Navazo I., and Scopigno R.: "Multiresolution volume visualization with a texture-based octree", The Visual Computer, 17(3), 2001, 185–197

[2] Callahan S. P., Ikits M., Comba J. L. D., and Silva C. T.: "Hardwareassisted visibility sorting for unstructured volume rendering", IEEE Transactions on Visualization and Computer Graphics, 11(3), 2005, 285–295

[3] Doleisch H., Gasser M., and Hauser H.: "Interactive feature specification for focus+context visualization of complex simulation data", Proc. of the 5th Joint IEEE TCVG - EUROGRAPHICS Symposium on Visualization (VisSym 2003), 2003, 239–248

[4] Doleisch H. and Hauser H.: "Smooth brushing for focus+context visualization of simulation data in 3D", WSCG 2002 Conference Proceedings, 2002, 147–154

[5] Everitt C.: "Interactive order-independent transparency", 2001

[6] Hadwiger M., Sigg C., Scharsach H., Bühler K., and Gross M.: "Realtime ray-casting and advanced shading of discrete isosurfaces", Computer Graphics Forum 24(3), 2005, 303–312

[7] Ju T., Schaefer S., and Warren J.: "Mean value coordinates for closed triangular meshes", Proceedings of SIGGRAPH 2005, 2005, 561–566

[8] Kraus M. and Ertl T.: "Cell-projection of cyclic meshes", Proceedings IEEE Visualization 2001, 2001, 215–222

[9] Krüger J., Schneider J., and Westermann R.: "ClearView: An interactive context preserving hotspot visualization technique", Proceedings IEEE Visualization 2006, 2006, 941–948

[10] Krüger J. and Westermann R.: "Acceleration techniques for GPU-based volume rendering", Proceedings IEEE Visualization 2003, 2003, 287–292

[11] LaMar E., Hamann B., and Joy K. I.: "Multiresolution techniques for interactive texture-based volume visualization", Proceedings IEEE Visualization ’99, 1999, 355–361
[12] Langer T., Belyaev A., and Seidel H.-P.: "Spherical barycentric coordinates". Proceedings of Eurographics Symposium on Geometry Processing 2006, 2006, 81–88
[13] Leven J., Corso J., Cohen J., and Kumar S.: "Interactive visualization of unstructured grids using hierarchical 3D textures" Proceedings IEEE Symposium on Volume Visualization 2002 (VolVis 2002), 2002, pages 37–44
[14] Max N., Williams P., Silva C., and Cook R.: "Volume rendering for curvilinear and unstructured grids". Proceeings of Computer Graphics International, 2003, 210–215
[15] Parker S., Parker M., Livnat Y., Sloan P.-P., Hansen C., and Shirley P.: "Interactive ray tracing for volume visualization", IEEE Transactions on Visualization and Computer Graphics 5(3), 1999, 238–250
[16] Röttger S., Guthe S., Schieber A., and Ertl T.: "Convexification of unstructured grids", Proceeings of the 9th Fall Workshop on Vision, Modeling and Visualization (VMV 2004), 2004, 283–292
[17] Röttger S., Kraus M., and Ertl T.: "Hardware-accelerated volume and isosurface rendering based on cell-projection", IEEE Visualization, 2000, 109–116
[18] Schmidt S., Schögl O., Kirchberger R., Doleisch H., Muigg P., Hauser H., Grabner M., Bornik A., and Schmalstieg D.: "Novel visualization and interaction techniques for gaining insight into fluid dynamics in internal combustion engines", Proceedings of the NAFEMS Worldcongress, 2005, full Proceedings on CDROM
[19] Shirley P. and Tuchman A. A.: "Polygonal approximation to direct scalar volume rendering", Proceedings San Diego Workshop on Volume Visualization, Computer Graphics, volume 24, 1990, 63–70
[20] Stein C. M., Becker B. G., and Max N. L.: "Sorting and hardware assisted rendering for volume visualization", Proceedings IEEE Symposium on Volume Visualization ’94 (VolVis ’94), 1994, 83–89
[21] Trenker M., Lang H., Muigg P., and Doleisch H.: "Validation of vortex flow phenomena in electrical machinery using advanced simulation and visualization techniques", Proceedings of the NAFEMS Worldcongress, 2007, full Proceedings on CDROM
[22] Uesu D., Bavoil L., Fleishman S., Shepherd J., and Silva C. T.: "Simplification of unstructured tetrahedral meshes by point sampling", Proceedings Volume Graphics, 2005, 157–165
[23] Viola I., Feixas M., Sbert M., and Gröller M. E.: "Importance-driven focus of attention", Proceedings IEEE Visualization 2006, 2006, 933–940
[24] Weiler M. and Ertl T.: "Hardware-software-balanced resampling for the interactive visualization of unstructured grids", Proceedings IEEE Visualization 2001, 2001, 199–206
[25] Weiler M., Kraus M., Merz M., and Ertl T.: "Hardware-based ray casting for tetrahedral meshes", Proceedings IEEE Visualization 2003, 2003, 333–340
[26] Weiler M., Kraus M., Merz M., and Ertl T.: "Hardware-based viewindependent cell projection", IEEE Transactions on Visualization and Computer Graphics 9(2), 2003, 163–175
[27] Weiler M., Mallón P. N., Kraus M., and Ertl T.: "Texture-encoded tetrahedral strips", Proceedings IEEE Symposium on Volume Visualization 2004 (VolVis 2004), 2004, 71–78, 2004.
[28] Weiler M., Westermann R., Hansen C. D., Zimmerman K., and Ertl T.: "Level-of-detail volume rendering via 3D textures", Proceedings IEEE Symposium on Volume Visualization 2000 (VolVis 2000), 2000, 7–13
[29] Westermann R.: "The rendering of unstructured grids revisited", Proceedings of the 3rd Joint IEEE TCVG - EUROGRAPHICS Symposium on Visualization (VisSym 2001), 2001, 65–74
[30] Zhou Y., and Garland M.: "Interactive point-based rendering of higherorder tetrahedral data", IEEE Transactions on Visualization and Computer Graphics 12(5), 2006, 1229–1236