

Interactive Visualization of Fluid Dynamics Simulations in Locally Refined Cartesian Grids

Martin Schulz ^{*†}

Frank Reck [‡]

Wolf Bartelheimer [†]

Thomas Ertl ^{*}

^{*} University of Stuttgart, IfI, Visualization and Interactive Systems Group

[†] BMW AG

[‡] University of Erlangen, IMMD IX, Computer Graphics Group

Abstract

This work presents interactive flow visualization techniques specifically adapted for PowerFLOWTM, a lattice-based CFD code from the EXA corporation. Their Digital PhysicsTM fluid simulation technique is performed on a hierarchy of locally refined cartesian grids with a fine voxel resolution in areas of interesting flow features. Among other applications the PowerFLOW solver is used for aerodynamic simulations in car body development where the advantages of automatic grid generation from CAD models is of great interest. In a joint project with BMW and EXA we are developing a visualization tool which incorporates virtual reality techniques for the interactive exploration of the large scalar and vector data sets. In this paper we describe the specific data structures and interpolation techniques and we report on fast particle tracing taking into account collisions with the car body geometry. An OpenGL Optimizer based implementation allows for the inspection of the flow with particle probes and slice probes at interactive frame rates.

Keywords: Computational fluid dynamics, interactive flow visualization, locally refined cartesian grids, virtual environment

1 Motivation and related work

Computational fluid dynamics (CFD) is becoming increasingly important for the evaluation of car body design concepts in the vehicle development process. Commonly used simulation applications are based upon the Navier-Stokes equation and use the finite volume method to numerically solve the partial differential equations. The finite volumes are organized in curvilinear or unstructured grids that fit the car body surface. The process to make the cell faces match the surface is the most time-consuming problem in grid generation. Complex surface structures of vehicles lead to grid generation times of up to several weeks.

Competition and reduced model cycles require the iteration loops during the vehicle development process to be as short as possible. Therefore, keeping the simulation model up to date with the designed surface changes requires a fast generation of the simulation grids. The more variants the designer is able to check the better should be the result of the optimization steps. This is in contrast to the efforts involved in regenerating a new grid structure for finite volume methods after only small vehicle surface modifications.

A promising way to speed up the grid generation is to use the Lattice Gas or Lattice Boltzmann methods to solve the flow problems. In this case the underlying grids are usually cartesian with an

additional explicit description of the vehicle surface. These grids can be generated automatically from the CAD representation of the car geometry and they can be refined in interesting regions. Thereby, a fast regeneration of the grid after small design changes is guaranteed.

PowerFLOW is lattice-based flow code developed by EXA which uses a special simulation technology called Digital Physics. The grids of PowerFLOW consist of locally refined cartesian grids. At interesting regions, defined by the engineer, the voxel size is successively cut into halves allowing for a fine resolution of interesting flow features. BMW is presently evaluating this software for the simulation of air flow around a car body by comparing their results with measurements taken in a wind tunnel. Introducing this software into the productive development process increases the strong demand for an interactive visualization tool.

Advanced interactive visualization techniques are needed to analyze the large amount of data generated by such simulations. However, existing visualization tools for curvilinear or unstructured meshes cannot be used for the data sets generated by PowerFLOW. Thus, our goal is to adapt known visualization methods to the special requirements of the locally refined cartesian grids. Specifically, we will show how the basic algorithms for cell location and interpolation have to be modified. The combination of volume data and geometric data also requires new techniques to handle particle tracing. The application developed at the University of Erlangen in cooperation with BMW and EXA aims at interactive performance in virtual environments. Therefore, special attention is paid to the positioning and manipulation of visualization probes as well as to the optimization of rendering performance.

In this respect we are guided by the pioneering work of Bryson [1] who demonstrated the usefulness of virtual reality techniques for fluid dynamic visualization in the virtual wind tunnel project. Furthermore, we rely on research results from the flow visualization community [4, 3, 6] and we try to include experiences from other case studies describing flow visualization environments [7].

2 Grid structure

The Lattice Gas solution of EXA works on a cartesian grid with several levels (called "VRlevel") of different voxel sizes. A typical simulation volume is a cube with a length of about 60 meters and a voxel size of about 5 millimeters in the finest regions. In each of the next higher levels the edge length of the voxels is doubled. These regions (called "VRregion") of a certain VRlevel are specified by the engineer in a CAD description. Several VRregions can share one VRlevel. An average simulation volume consists of about 20-25 million voxels. In order to reduce the disk space requirements, the average value of eight voxels is combined into one "measurement cell" which defines the simulation parameters to be constant across the entire cell. A more practical view is to define the parameter at the cell center. An average data set as stored on disk consists of around 3 million measurement cells.

^{*} Institut für Informatik, Abteilung für Visualisierung und Interaktive Systeme, Breitwiesenstr. 20-22, 70565 Stuttgart, Germany,

Email: schulz@informatik.uni-erlangen.de,

[†] BMW AG, 80788 München, Germany,

Email: wolf.bartelheimer@bmw.de

[‡] Lehrstuhl für Graphische Datenverarbeitung IMMD IX, Am Weichselgarten 9, 91058 Erlangen, Germany,

Voxels inside the vehicle are not relevant for the simulation and do not contain any parameters. The vehicle geometry intersects several voxels, which need special handling to allow particle tracing. In figure 1 a grid structure example with two VRlevels is shown. Obviously, not every voxel of an axis aligned refinement region has to be split.

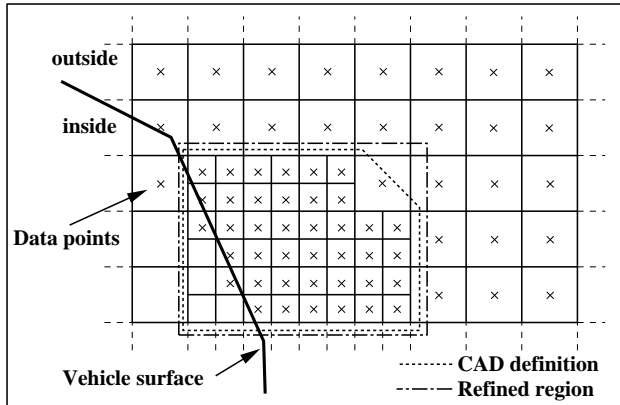


Figure 1: Grid structure of a locally refined cartesian grid containing explicit geometry.

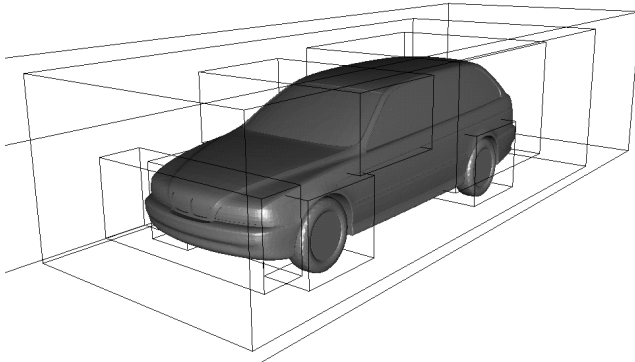


Figure 2: Example data set with locally refined regions

Figure 2 presents a data set with three VRlevels and nine VRregions. The VRlevel with the smallest cell size is cut into seven VRregions. This example clarifies the interesting regions for refinement. They are located around the wheels, in front of the wind-screen and at the front and back of the car.

Most visualization algorithms are built on cell location and interpolation as basic operations. Finding the enclosing cell to a given position should be as fast as possible in order to achieve interactive rates of visualization. For the described grid structure this means traversing the hierarchy of VRlevels. In order to speed up this process, a condition byte for each cell in each level is used to store the following information:

- the cell contains parameters
- the cell is refined and which is the underlying VRregion
- the cell is intersected by surface geometry

This three-dimensional condition byte field is stored in addition to the other cell parameters. Using this byte field an algorithm can be found which is simple and fast compared to cell location in curvilinear grids (e.g. stencil walk) or unstructured grids:

```

start in the highest VRlevel
while(1){
  calculate the cell index of the position in the grid:
  index = (position - minimal grid bound) / cell size
  if (cell has no parameters)
    return no cell found
  if (cell is refined)
    continue with underlying VRlevel/VRregion
  /* cell found */
  return this cell
}

```

With respect to interpolation the data model with cell-centered parameters, which are considered to be constant across the voxel volume, implies that “nearest neighbour interpolation” is sufficient.

3 Particle tracing and collision detection

Particle tracing in stationary grids is known in different variants. The commonly used integration method is the Runge Kutta scheme of up to the 4th order. Better results than with fixed step size can be achieved with adaptive step size control, i.e. with the embedded 3(2) or 4(3) Runge Kutta methods [5]. Since we have implementations of all these methods available, we let the user choose between the different integration methods in order to get best performance for a certain desired accuracy.

In curvilinear or unstructured grids the vehicle surface is implied in the grid structure. If a particle hits the surface it just leaves the grid. However, the grids considered in this work have an explicit description of the vehicle surface (see figure 1). Particle traces near the surface could cut through the surface and appear again at another point, as shown on the left side in figure 3. Since this violates the surface boundary condition, it would be better if the particle path ends on the surface or if the particle tracing is continued in close relationship to the surface.

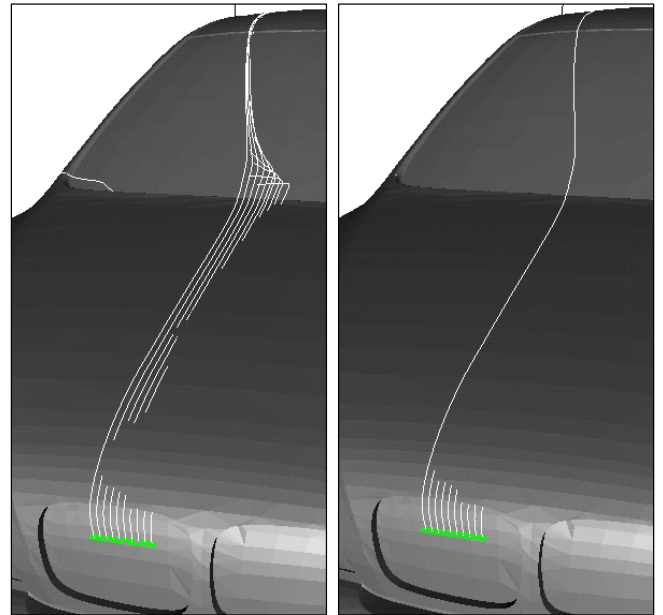


Figure 3: Particle tracing without (left) and with (right) collision detection on the car surface.

An average surface geometry consists of about 70.000 triangles. Thus, it would not be efficient to check each particle step against all triangles of the vehicle surface. In this work an octree has been implemented to reduce the number of triangles that have to be checked for a specific step to less than 20.

The octree is built as usual by dividing the whole cube into eight smaller units with half the cell size. Regions with no geometry are enclosed by a large octree cell, whereas cells containing geometry areas are refined until the number of triangles in the octree cell falls short of a minimum limit or the cell encloses only one voxel. Since the octree cuboids share their edges with the grid structure, each measurement cell of the grid is assigned to one octree cell. However, a triangle of the vehicle surface can be part of different cells, because the triangle edges are in general not in parallel to the grid axis.

During the integration of the particle trace, all cells intersected by a path have to be checked for intersection with the vehicle surface. Without the condition byte field each cell has to be found in the octree. Checking the condition-byte for geometry information reduces the necessary octree tests to a minimum (figure 4). Only the cells that are intersected by geometry have to be searched in the octree. The traversal of the octree returns the triangles for intersection testing which is done by the Glassner algorithm [2], known from ray tracing techniques. Figure 3 compares particle traces with and without intersection test. In the left figure many particle traces cut through the hood and appear again, whereas the intersection test stops them at the surface, as shown in the right image.

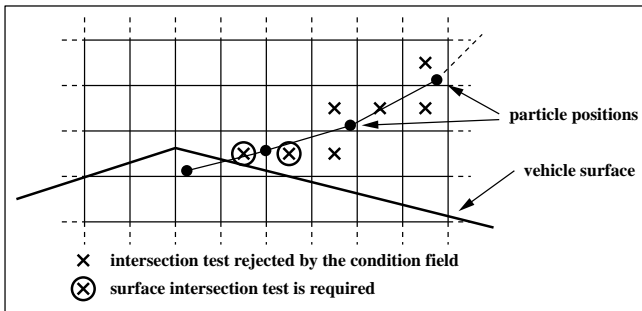


Figure 4: Particle trace intersecting the vehicle surface.

An average of ten triangles has to be checked for one cell. The high-lighted triangles of figure 5/7 have been referenced for intersection tests. Only a very small number of the whole car surface is used for the trace of one particle. The example of figure 5/7 clearly demonstrates the requirement of the intersection test. First the particle flies close to the surface and finally hits the wheelhouse.

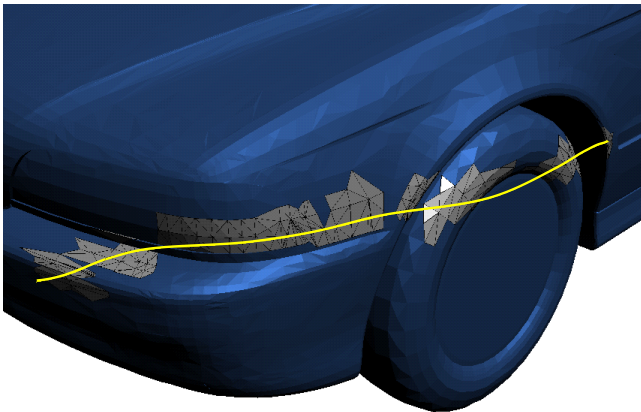


Figure 5: The high-lighted triangles had to be checked for collision during the particle trace calculation.

Even more interesting is the collision detection for particles with

masses and size, e.g. dirt particles. Because of gravity and acceleration, they hit the vehicle surface more often which requires the octree structure for a fast hit point calculation.

As can be seen in figure 9 our system allows particle traces to be visualized as streaklines, ribbons, and glyphs. The ribbons show the local rotation of the flow by calculating the Jacobian of the particle velocity. The glyph is constructed as an arrow pointing into the flow direction. The peak, the shaft and the stump can be colored and scaled by different scalar parameters. Scaling the shaft by the vorticity is a good example for scalar parameter mapping.

4 Implementation and measurement probe interactions

The visualization system implemented in this project is based upon SGI's graphics API *OpenGL Optimizer*, which is available for SGI, HP and Windows NT platforms. The underlying object oriented *Cosmo3D* scene graph API offers several optimization techniques, e.g. occlusion culling, polygon reduction and accelerated moving of picked objects. Thereby, *Cosmo3D* allows full access to the graphical objects, including *OpenGL*-calls.

At the current project status user input is supported for a standard 2D mouse and for the DLR space mouse, which allows simultaneous input in six degrees of freedom in a precise manner. In the future we plan to support tracking devices to run the application in a CAVE or on large screen stereo projection walls. Using these input devices the user can manipulate the position and orientation of the view point or of one of the measurement probes.

Using measurement probes gives the user an intuitive and easy way to analyze the data sets. The probes can be freely moved and oriented to define the initial positions for particle traces or the orientation of slices. Therefore, the shape of the probes can be expanded in all three dimensions and the sample points are aligned on the rake or inside the cube (see figure 6).

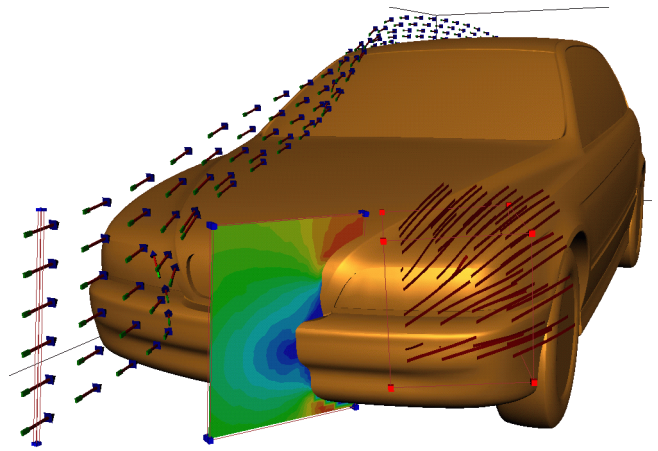


Figure 6: Freely movable measurement probes, as rake, slice and cube.

In case of the slice measurement probe, planes can be colored by the scalar value calculated at the sample points. Using the texture hardware allows interactive visualization of isolines (see figure 9). The deficit of slices is the restriction to two dimensions. Direct volume rendering is an expensive way to get a three dimensional presentation. To keep up with interactive rates, we arranged several slices in a cubical probe, avoiding the occlusion by rendering a specific range of the scalar parameter by using transparent textures. For example, in figure 8 the slices are colored by the velocity norm and for large values the texture is transparent.

Thereby, we can see the shape of the vortex caused by the wheel housing.

5 Results

The characteristic size parameters of a typical data set are presented in table 1 with the corresponding vehicle surface containing 70.820 triangles (see figure 2). While a full cartesian grid of the finest resolution would have about 60 million cells, the locally refined grid has just about 3 million cells containing flow parameters. Note again that not every cell in a refined region is divided into the underlying VRlevel. The example data set can be visualized on a *SGI Octane SI* with a R10000 175 MHz processor and 256 MB memory running under *IRIX 6.5* using *OpenGL Optimizer 1.2*. Being a typical workstation configuration in engineering environments we consider this to be the target platform for our visualization system.

| VRregion | VRlevel | child regions | dimensions of cells | | | cell size in cm |
|----------|---------|---------------|---------------------|-----|-----|-----------------|
| 0 | 2 | 1 | 309 | 44 | 69 | 5.0 |
| 1 | 1 | 2 - 8 | 287 | 80 | 125 | 2.5 |
| 2 | 0 | - | 87 | 72 | 49 | 1.25 |
| 3 | 0 | - | 87 | 72 | 49 | 1.25 |
| 4 | 0 | - | 43 | 71 | 189 | 1.25 |
| 5 | 0 | - | 191 | 125 | 189 | 1.25 |
| 6 | 0 | - | 111 | 67 | 189 | 1.25 |
| 7 | 0 | - | 87 | 72 | 49 | 1.25 |
| 8 | 0 | - | 87 | 72 | 49 | 1.25 |

Table 1: Grid structure of a typical example data set.

In table 2 the calculation time for 100 particle traces, each with 1000 steps, with and without geometry check is compared. The time measurement includes the interpolation, integration and generation of the visualization geometry. The “worst case” for the algorithm with geometry check is a trace near the vehicle surface with a polygon intersection test for each step. For such a trace, we need for 100 particles about 3.52 seconds with and 1.67 seconds without geometry testing. The “best case” is a trace with no geometry interferences at all, where the times are 2.12 seconds with and 1.66 seconds without surface test. This clearly shows the limited overhead of geometry testing: while in the worst case we need about twice the time the increase for the best case is only 27 percent. An average particle exhibits an intersection test in less than 30 percent of the steps, so the average time for 100 particles traces is around 2 seconds. Keep in mind that all numbers are for 100 particles each with 1000 calculated steps. Reducing the number of steps, e.g. by using adaptive integration methods, speeds up the calculation. The effect of the decreasing ratio for higher order integration schemes is caused by the constant time for the geometry creation.

| Order of Runge Kutta scheme | trace near geometry “worst case” | | | trace in volume “best case” | | |
|-----------------------------|----------------------------------|---------------|-------|-----------------------------|---------------|-------|
| | with check | without check | ratio | with check | without check | ratio |
| 2. | 3.22 | 1.35 | 2.38 | 1.80 | 1.35 | 1.33 |
| 3. | 3.56 | 1.72 | 2.07 | 2.16 | 1.69 | 1.27 |
| 4. | 3.80 | 1.94 | 1.96 | 2.41 | 1.94 | 1.24 |

Table 2: Tracing and geometry generation time in seconds for 100 particles, each with 1000 calculated steps.

Using a probe with about 10 particle traces allows interactive exploration of the given complex data set. The particle rake can be positioned freely and intuitively with the space mouse and the traces are updated instantaneously. For the engineers this a significant

progress compared to their previous plot-job oriented visualization procedures. With respect to the slice probe a plane of the size of 70 x 70 points can be moved at interactive rates.

The most time consuming part in the visualization process is the geometry refresh. The example model of 70.000 triangles can be rendered with 8 frames per second on the described platform. Our goal is to achieve frame rates of more than 10 by geometry optimization like adapted tri-stripping and polygon reduction.

6 Conclusions and future work

Advanced simulation techniques like the Lattice Gas Automata concept for CFD introduce complex new grid structures which require the modification of well-known visualization algorithms. We have shown that careful adaption to the hierarchy of the locally refined cartesian grids of EXA’s PowerFLOW solver allows particle tracing with sets of lines, ribbons, and glyphs at interactive frame rates. Special effort is required to avoid the penetration of particles into the car body, an aspect which is not relevant for geometry adapted unstructured or curvilinear grids. Interactive manipulation of probes for particle rakes or slice planes can be driven by space mice attached to the graphics workstations or by pointing devices in immersive visualization environments. In the future we plan to implement other mapping techniques like LIC for depicting surface flow and iso-surfaces representing scalar values like pressure and temperature. Furthermore, we continue to improve rendering performance by reducing and optimizing the car body geometry and culling occluded parts. Ultimately, we head for the interactive exploration of time-dependent computations and the integration of our visualization system into the development process of BMW.

7 Acknowledgment

This research is sponsored by the Bavarian Consortium for High Performance Scientific Computing (FORTWIHR). We would like to thank our project partners Dr. K. R. Traub, EXA Corp., and Dr. Nölting, EXA GmbH, for fruitful discussions and their continuing support.

References

- [1] S. Bryson and C. Levit. The virtual windtunnel. In *IEEE Computer Graphics and Applications*, 12(4):25-34, 1992.
- [2] A. Glassner. *An Introduction to Ray Tracing*. Academic Press London, 1989.
- [3] D. A. Lane. Scientific Visualization of Large-Scale Unsteady Fluid Flows. In G. Nielson, H. Hagen, and H. Mueller, editors, *Scientific Visualization*, pages 125–145. IEEE Computer Society, 1997.
- [4] F. Post and T. van Walsum. Fluid Flow Visualization. In H. Hagen, H. Mueller, and G. Nielson, editors, *Focus on Scientific Visualization*, pages 1–40. Springer Berlin, 1997.
- [5] C. Teitzel, R. Grosso, and T. Ertl. Efficient and Reliable Integration Methods for Particle Tracing in Unsteady Flows on Discrete Meshes. In W. Lefer and M. Grave, editors, *Visualization in Scientific Computing '97*, pages 31–41, Wien. Springer.
- [6] S.K. Ueng, C. Sikorski, and K.L. Ma. Efficient construction of streamlines, streamribbons and streamtubes on unstructured grids. *IEEE Transactions on Visualization and Graphics*, 2(2):100–109, June 1996.
- [7] S. P. Useton. exVis: Developing A Wind Tunnel Data Visualization Tool. In R. Yagel and H. Hagen, editors, *Proc. Visualization '97*. IEEE, 1997.

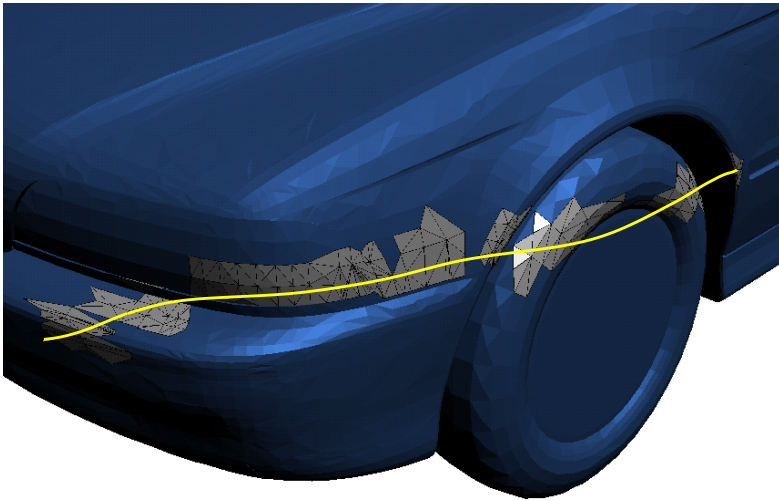


Figure 7: The lighted triangles had to be checked for collision during the particle trace calculation.

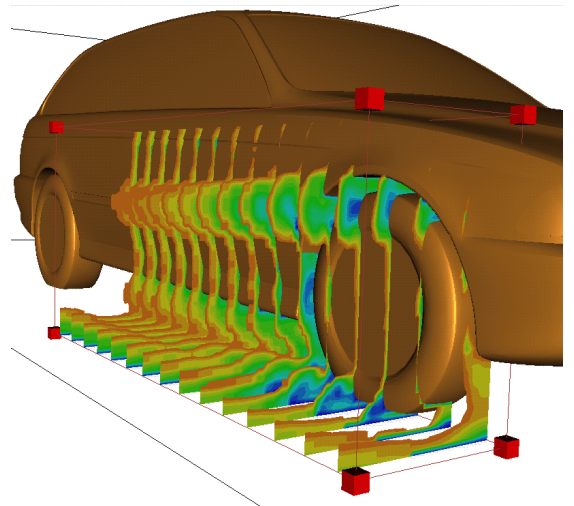


Figure 8: Freely movable slicing probe, using texture hardware to visualize the shape of a vortex.

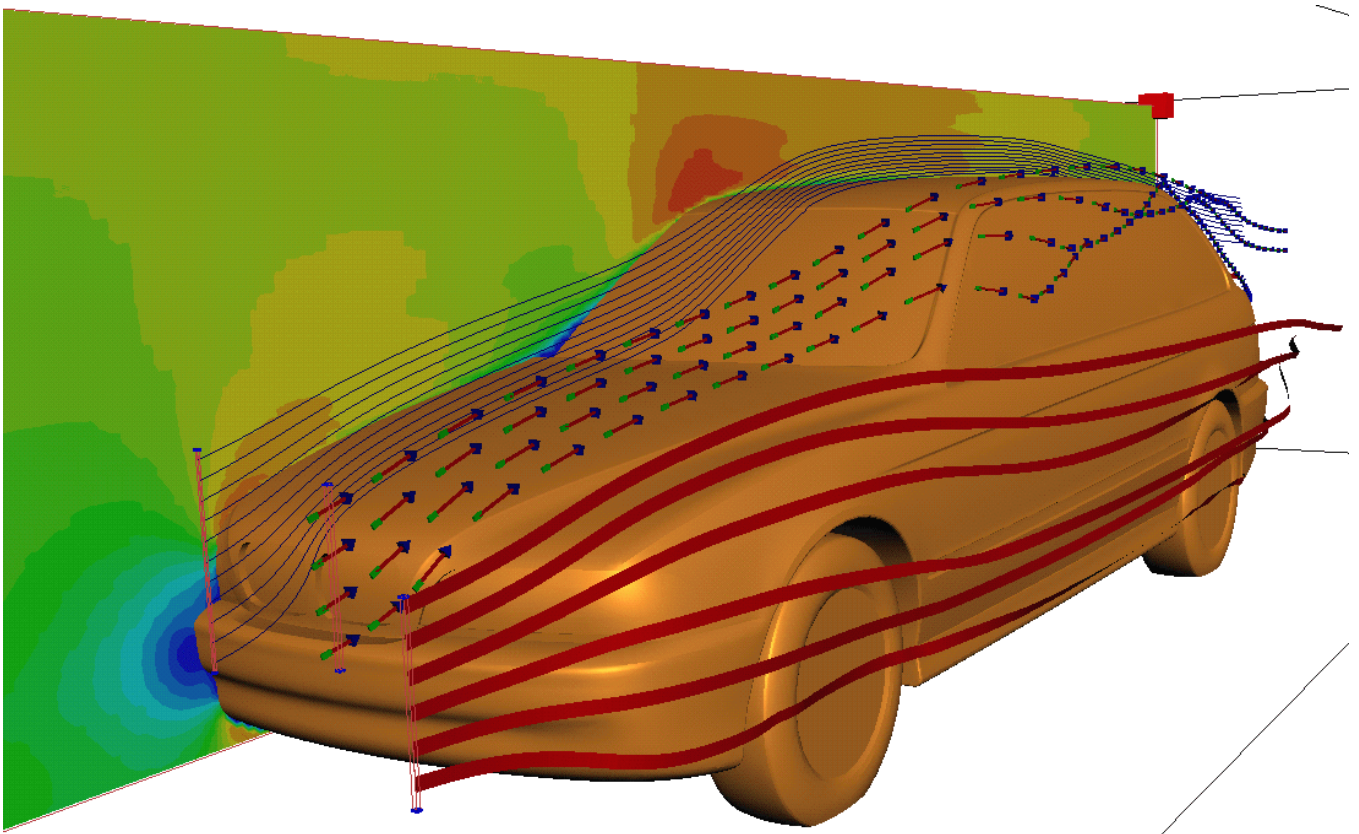


Figure 9: Particle trace visualization with streaklines, ribbons and glyphs. Isolines on the slice are visualized interactively by the texture hardware.