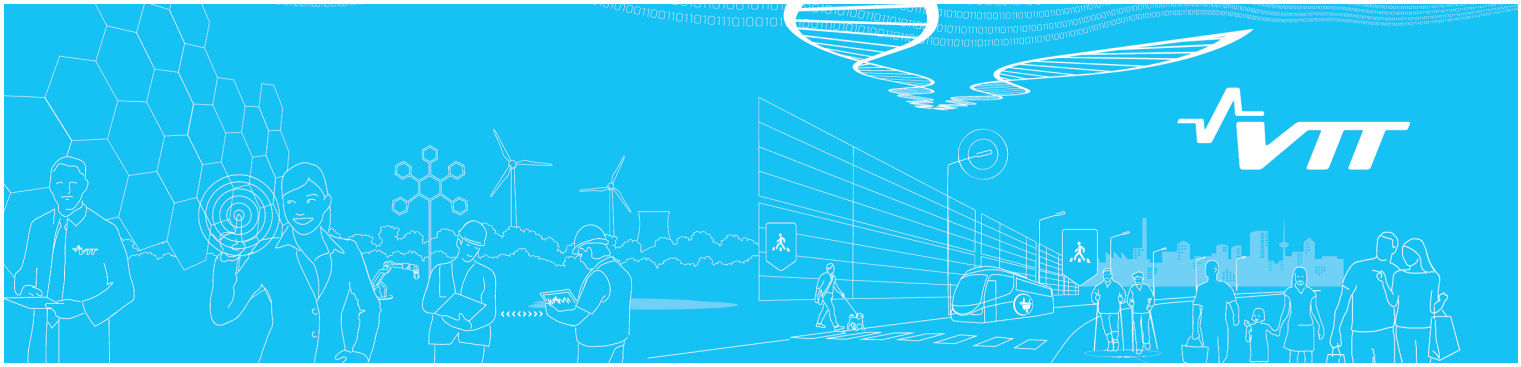
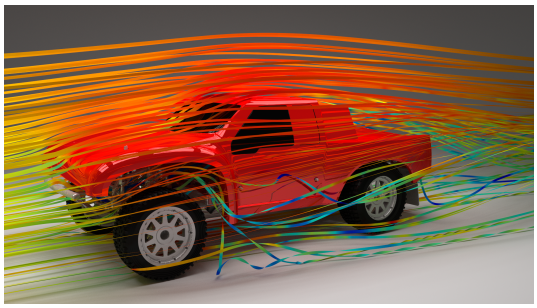


RESEARCH REPORT

VTT-R-04911-15



Open source tools in technical photorealistic large-scale visualisation

Authors: Aku Karvinen

Confidentiality: Public (starting from October 31, 2015)

Report's title
Open source tools in technical photorealistic large-scale visualisation

Customer, contact person, address	**Order reference**
Tekes, Matti Säynätjoki P.O. Box 69, FI-00101 HELSINKI, FINLAND	1059/31/2012
Project name	**Project number/Short name**
Computational methods in mechanical engineering product development	100553/SIMPRO.
Author(s)	**Pages**
Aku Karvinen	16/–
Keywords	**Report identification code**
large-scale, visualisation, ParaView, Blender	VTT-R-04911-15

Summary

As the computational power available increases at approximately constant speed it has become feasible to simulate increasingly complex and therefore increasingly computationally demanding and large cases. It is therefore more and more important to develop post-processing and visualisation methods for very large simulation cases.

Parallel simulations using several CPU cores have already been done two or three decades ago. However, post-processing and visualisation has been done mainly using only serial process i.e. one CPU core. This limits post-processing and visualisation possibilities and makes post-processing and visualisation excessively slow in large cases.

In addition, it has been very common in the engineering simulation process that the result figures are not very photorealistic but mainly meant to be a design aid. They can be used in engineering design process but customers are nowadays more accustomed to more realistic figures and therefore they can demand more also in the engineering field. One field where figures have to be photorealistic, is advertisement. Usually these figures are produced in the advertisement agency, but can also be produced by the engineer responsible for the simulation.

In this report, the capabilities of open source software are investigated in the field of post-processing and engineering visualisation.

Results show that the Salome-platform – OpenFOAM – ParaView – Blender is a very capable open source tool chain for parallel computational fluid dynamics simulation and post-processing and can produce photorealistic results to be used for example in advertisement material.

Confidentiality	Public (starting from October 31, 2015)

Espoo 31.10.2015

Written by	**Reviewed by**	**Accepted by**
Aku Karvinen, Senior Scientist	Pekka Pasanen, Research Scientist	Johannes Hyrynen, Head of Research Area

VTT's contact address
VTT Technical Research Centre of Finland, P.O. Box 1000, FI-02044 VTT, FINLAND

Distribution (customer and VTT)
Matti Säynätjoki, Tekes, 1 copy
VTT archive, 1 copy

Contents

1. Introduction

As the computational power available increases approximately constant speed it has become feasible to simulate increasingly complex and therefore increasingly computationally demanding and large cases. It is not uncommon in computational fluid dynamics (CFD), for example, to use computational grids containing tens of millions computational cells. Some simulations consisting of even several billion cells have been done. It is therefore more and more important to develop post-processing and visualisation methods for that kind of very large simulation cases. This is also due to shift from steady simulations towards unsteady i.e. transient simulations in which case the amount of data produced is usually much larger.

Parallel simulations using several CPU cores have already been done two or three decades ago. However, post-processing and visualisation has been done mainly using only serial process i.e. one CPU core. This limits post-processing and visualisation possibilities and makes post-processing and visualisation excessively slow in large cases.

One major problem in commercial software is the license fees which heavily limit parallel simulation, post-processing and visualisation. Actually, there are only few if any commercial software capable of real parallel post-processing and visualisation in the field of engineering simulation.

In addition, it has been very common in the engineering simulation process that the result figures are not very photorealistic but mainly meant to be a design aid. They can be used in engineering design process but customers are nowadays more accustomed to more realistic figures and therefore they can demand more also in the engineering field. One field where figures have to be photorealistic, is advertisement. Usually these figures are produced in the advertisement agency, but can also be produced by the engineer responsible for the simulation.

In this report, capabilities of open source software are investigated in the field of post-processing and engineering visualisation. One particular thing to investigate is the capability of open source software tools to produce photorealistic figures.

2. Software used

The following software tools are investigated.

2.1 Salome-platform

Salome-platform is much more than only a CAD (computer-aided design) software, even as it is often used only for that. It is a generic platform for pre- and post-processing of numerical simulation. Developers of the software are Open Cascade, EDF and CEA. The software was initially released already in the year 2001. The software is written in C++ and Python and can be used in various operating systems, including Windows, Linux and other Unix-like systems. The software is published under GNU Lesser General Public License (LGPL).

2.2 OpenFOAM

As Wikipedia says:

> OpenFOAM (for "Open source Field Operation And Manipulation") is a C++ toolbox for the development of customized numerical solvers, and pre-/post-processing utilities for the solution of continuum mechanics problems, including computational fluid dynamics (CFD).

In addition, the OpenFOAM distribution comes with several ready-to-use solvers and utilities. Therefore, in common situations, it is not necessary to create an own solver but use an existing one. OpenFOAM was first released as open source software in the year 2004. Before that the software was closed source. OpenFOAM is available for Linux and other Unix-like operating systems and for Windows using Docker containerisation system or using third party distributions, for example simFlow. OpenFOAM is published under GNU General Public License (GPL).

One big advantage of the OpenFOAM in respect of large-scale post-processing is that a user can configure an OpenFOAM case to carry out post-processing while the simulation is running. This is done using a mechanism called function object. For example, if the user knows before the simulation run that the results are needed in a certain cutting plane, user can instruct OpenFOAM to save the desired data on that plane during the simulation. The range of different function objects is extensive, for example:

- `cuttingPlane` – for the above mentioned generation of cutting plane data,
- `probes` – for probing the data at point locations,
- `fieldValue` – for averaging or integrating across for example a plane and
- `streamLine` – for generating streamlines.

One important thing to notice in the function object mechanism is that the post-processing is done in parallel in the case of parallel simulation. This can often make separate parallel post-processing unnecessary.

2.3 ParaView

Word ”Para” in ParaView [1, 2, 3] comes from the word parallel, which indicates that the software is designed for parallel processing from the ground up and is therefore suitable for very large scale visualisation. ParaView is base on The Visualization ToolKit (VTK).

ParaView can be used in a server-client mode. There are three different sub-modes to use ParaView in a server-client mode. These sub-modes are pvrenderserver, pvdataserver and pvserver. The reason behind this is the possibility to use the different available computing resources efficiently, for example one computer with high performance CPUs and another with modern GPUs. However, the overhead of decomposing a case in order to use it with pvrenderserver and pvdataserver sub-modes is so high that it is usually recommended to use the simpler pvserver sub-mode. Therefore, in this report only this server-client sub-mode is used.

2.4 VisIt

VisIt is another open source post-processing tool based on VTK and therefore capable of parallel post-processing. VisIt is developed by Lawrence Livermore National Laboratory and is

written in C and C++. It is published under BSD (Berkeley Software Distribution) license. Because of close similarities between ParaView and VisIt, VisIt is not inspected more closely in this work.

2.5 Blender

Blender [4, 5] is defined in Wikipedia as follows:

> Blender is a professional free and open-source 3D computer graphics software product used for creating animated films, visual effects, art, 3D printed models, interactive 3D applications and video games.

Blender was initially released already in the year 1995 and is written in C, C++ and Python. Because of the quite large user base, a version of Blender is available for almost every operating system used in computers. Blender is released under GNU General Public License (GPL).

Blender is not used very often in the engineering field, but there are some cases where Blender has been utilised for example in the visualisation of OpenFOAM results.

There are several options of rendering engines to be used in conjunction with Blender, for example the original rendering engine coming with Blender called Blender Render. Due to poor picture quality of produced figures using the original rendering engine in some cases, users of Blender have created more advanced rendering engines which are distributed as add-ons. These engines include for example the LuxRender, which is reported to perform very well. The Blender distribution nowadays also includes a rendering engine called Cycles Render, which is reported to produce comparable rendering quality with best add-on rendering engines. In this work Cycles Render engine is used exclusively.

3. Example case

3.1 CAD model

A radio controlled (RC) car is chosen as the example case for this report. The original CAD model is provided by STEP Tools Inc. via STEP Index software demo (http://www.steptools.com/demos/) and is created by Vanderbilt University. RC car is chosen as the example case mainly because of its geometrical complexity. The case can be regarded even more demanding than normal passenger car because of the necessity to take into account the flow under the body. Complexity of the model can be seen clearly in Figures 1 and 2 which are screen shots of the CAD/pre-processing software Salome-platform and in which all the small details of the original model are shown.

Original CAD model is modified only slightly for CFD. The wheel bolts and part of the axis of the wheels are removed in order to split the rotating wheels from the non-rotating frame and thus improving the computational model behaviour. Complexity of the case used in CFD stage can be seen in Figures 3, 4, 5 and 6 which are made with Blender rendering software. Cycles Render is used as a rendering engine. The figures made with Blender represent the actual simulated CFD model – no additional simplification is applied to the model.

The RC car, ground and lightning used in Blender are shown in Figure 7. Ground has been bent like shown to avoid need of excessively large ground plane. Lightning, on the other hand, is

chosen to be like indicated because similar lightning set up is often used in real car photography, for example in advertising. A very large planar light is needed in order to bring forth all the reflections of the car. Several narrow light sources are, however, much cheaper to produce and therefore that kind of light set up consisting of so called strip boxes, is very often used. In addition, this kind of lightning is often regarded as more lively compared to the light produced by a one large planar light (softbox), which is often considered more blunt.

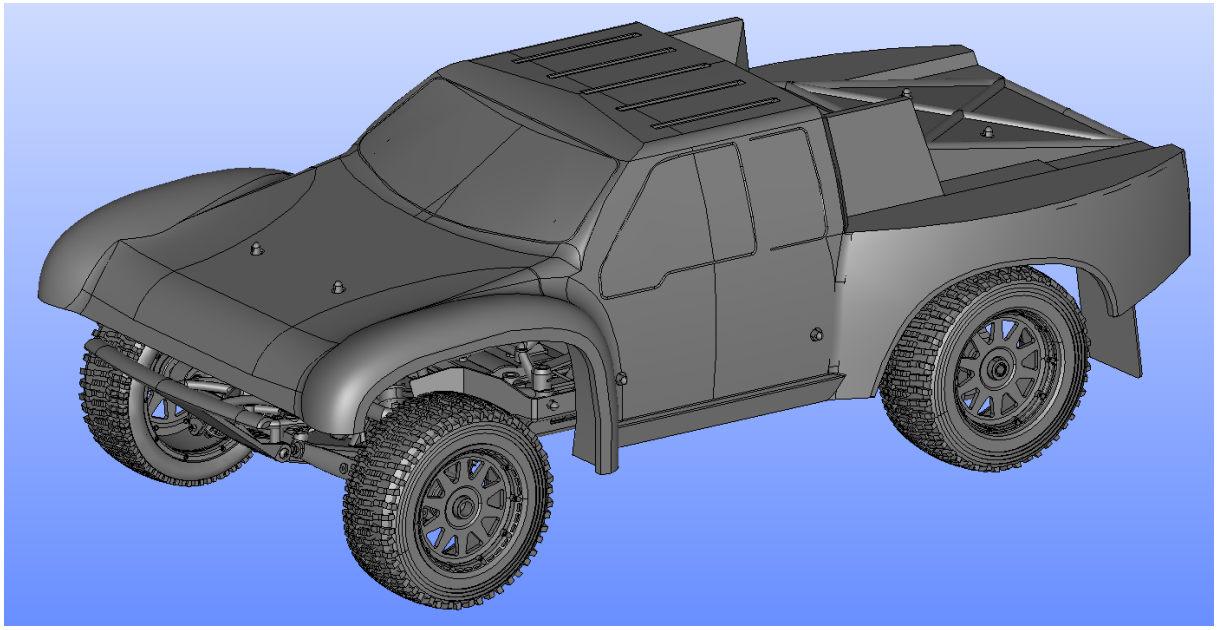


Figure 1. Original CAD model including several small details.

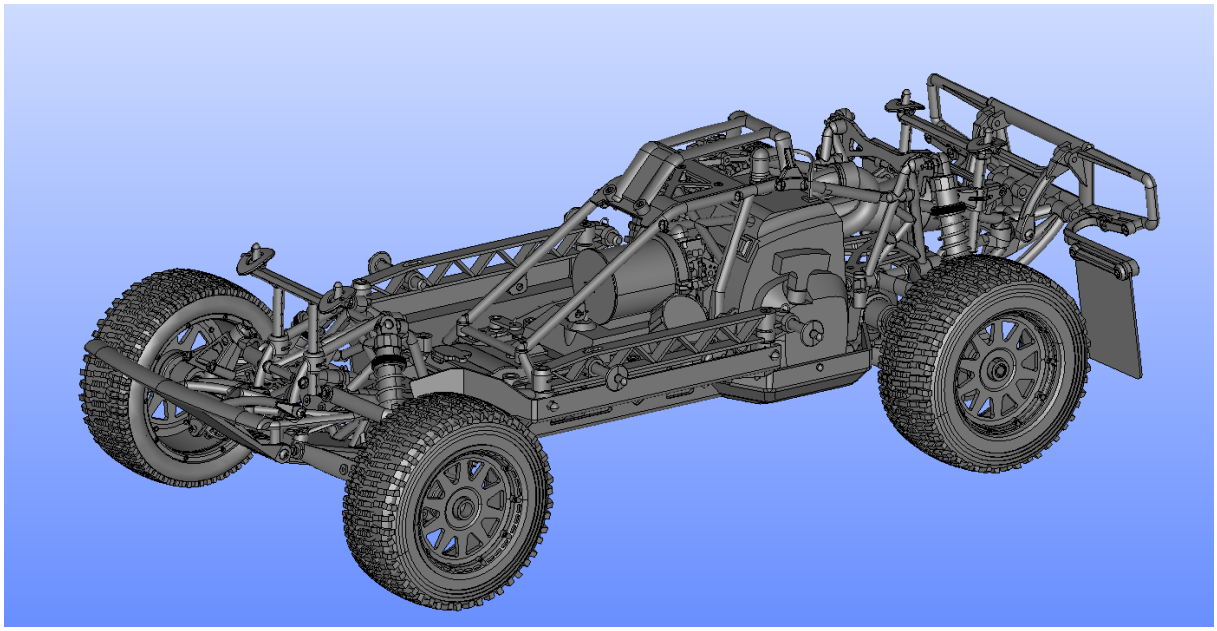


Figure 2. Original CAD model without the car body.

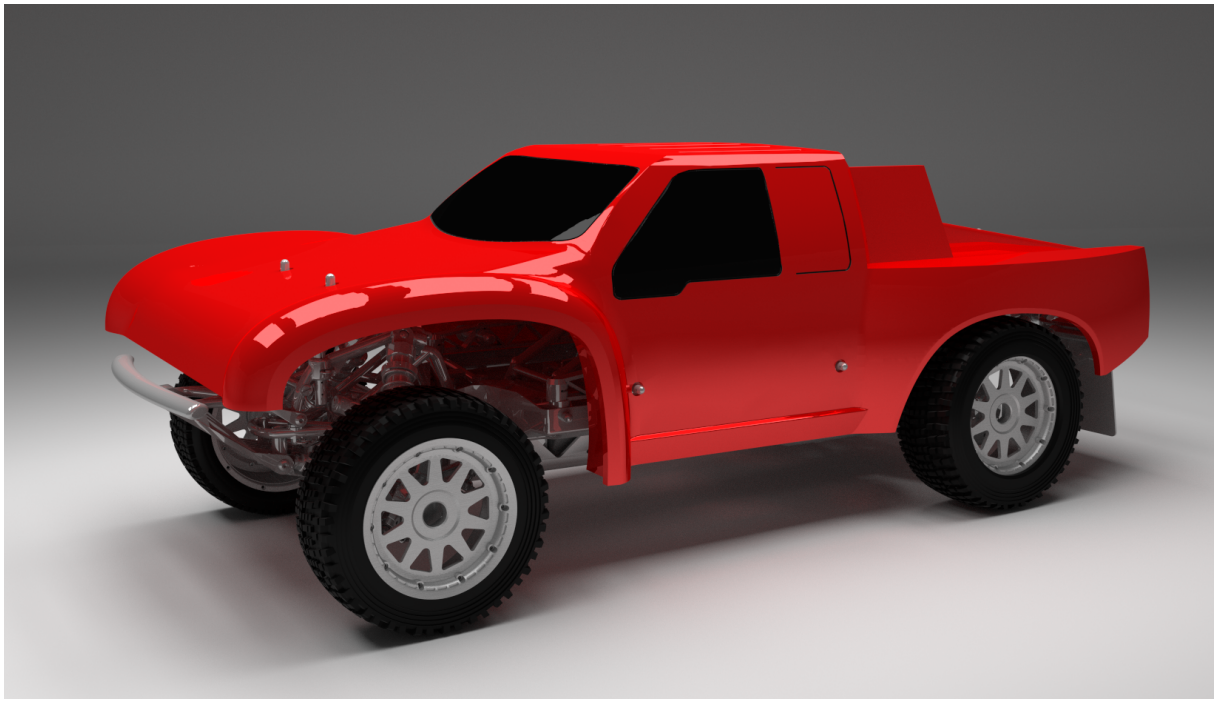


Figure 3. Case rendered using Blender.

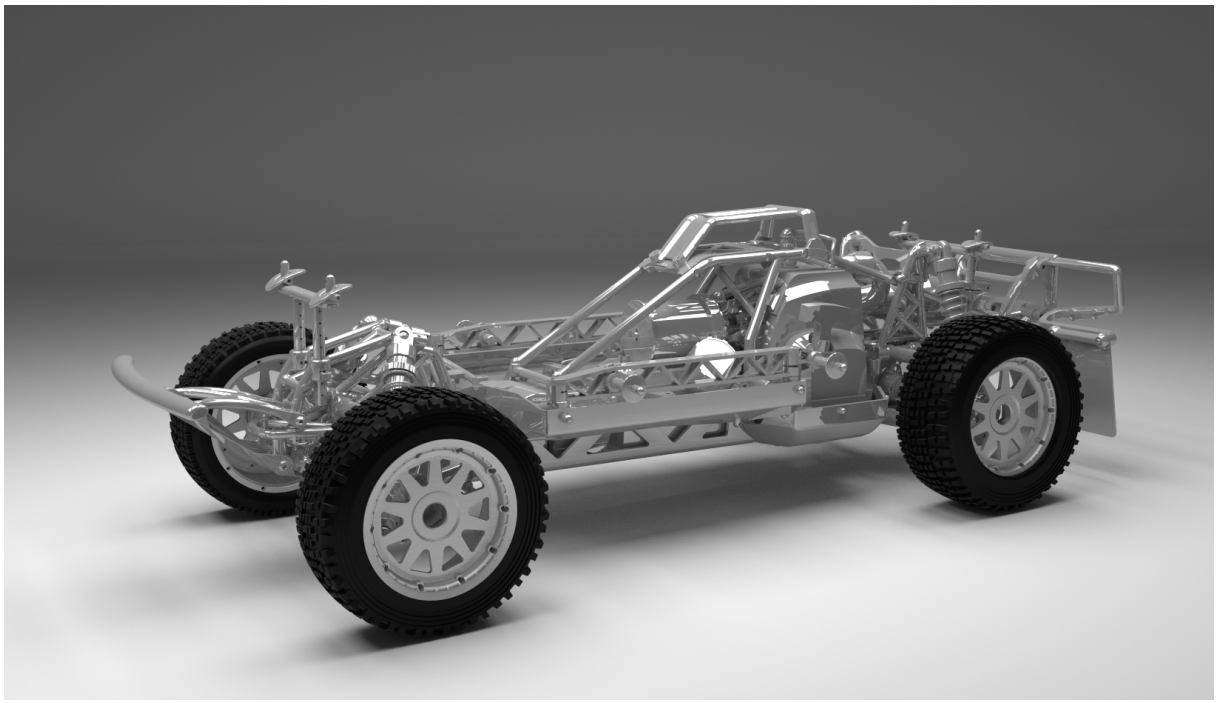


Figure 4. Case without the body rendered using Blender.

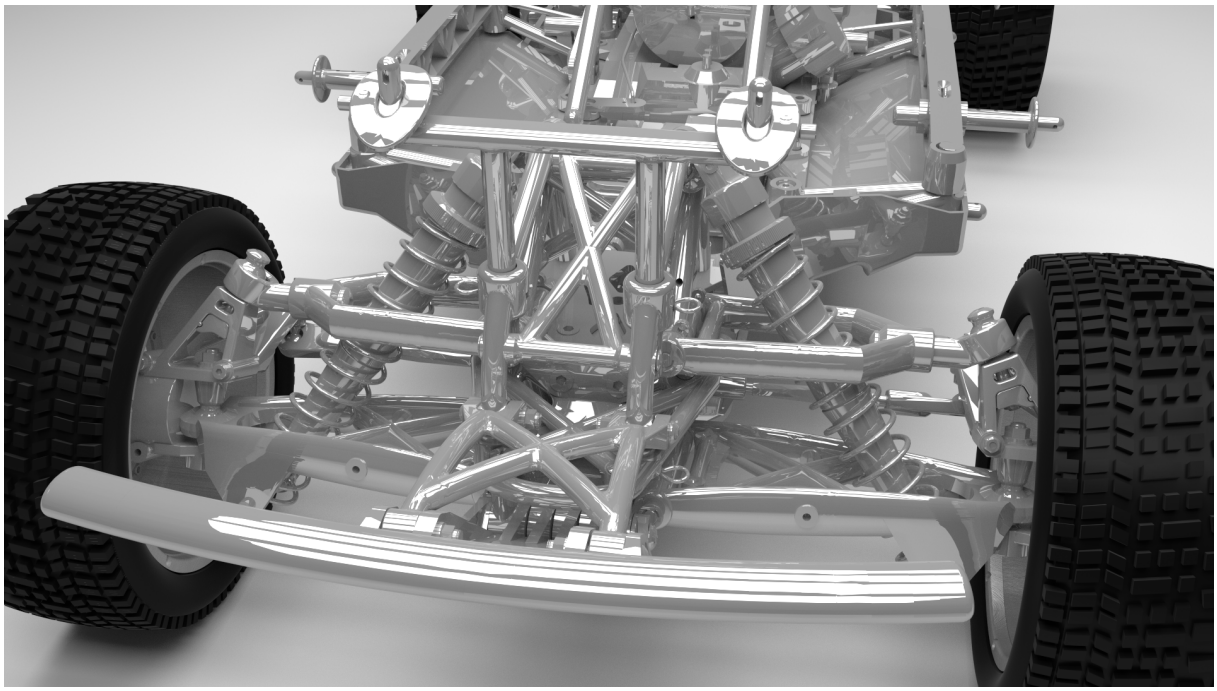


Figure 5. Details of the case without the body rendered using Blender.

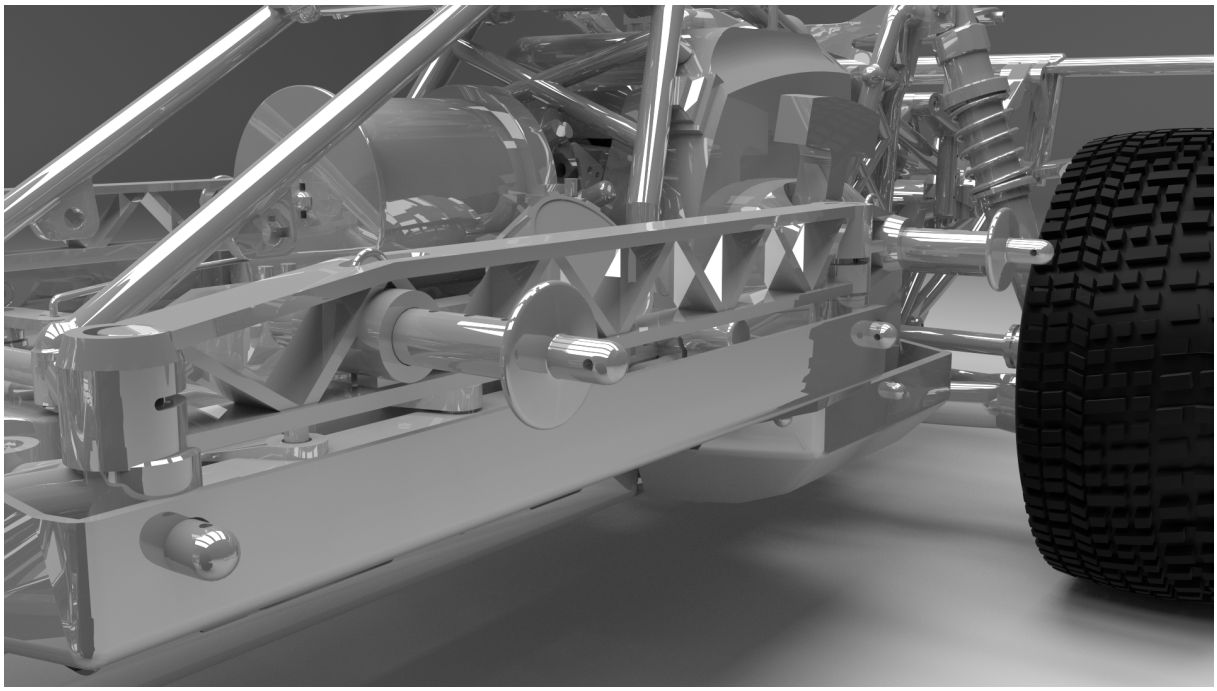


Figure 6. Details of the case without the body rendered using Blender.

3.2 CFD model

The CFD computational mesh of the slightly simplified CAD model is made using utility snappyHexMesh which is part of the OpenFOAM distribution. The surface mesh of the car body is shown in Figure 8. By inspecting the detailed figure of the surface mesh of the frame of the car (Figure 9), it is evident that all the smallest details of the design are taken into account. Because of the limitations in the computational capacity, the mesh is not fine enough and does

not include adequate amount of cells in the vicinity of the walls in respect of CFD theory. The mesh is also coarser far from the car, as can be seen in Figure 10.

The CFD mesh has about 16 million control volumes. The velocity is assumed to be constant (5 m/s). The velocity field is, however, simulated using an unsteady simulation method with unsteady Navier Stokes equations (URANS). Turbulence is modelled using a specially tailored model for the URANS simulations, namely SST k-ω SAS model [6, 7]. Vehicle wheel rotation is taken into the account by using the rotating wall velocity boundary condition. This means that

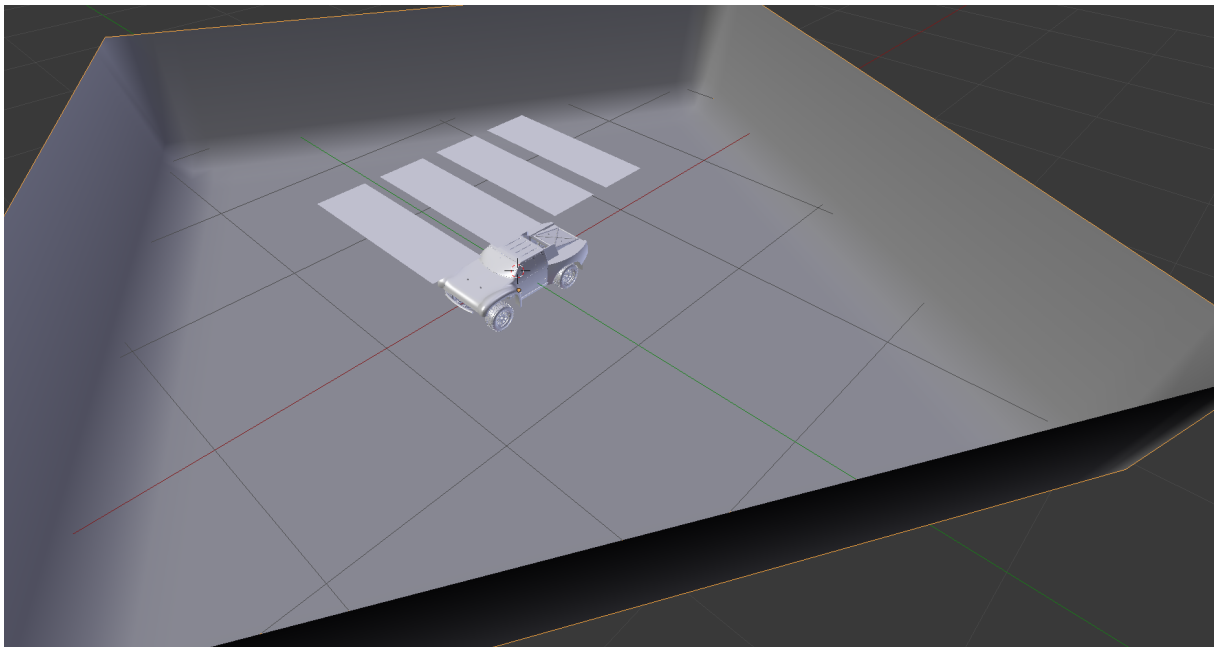


Figure 7. Case and lightning rendering set up.

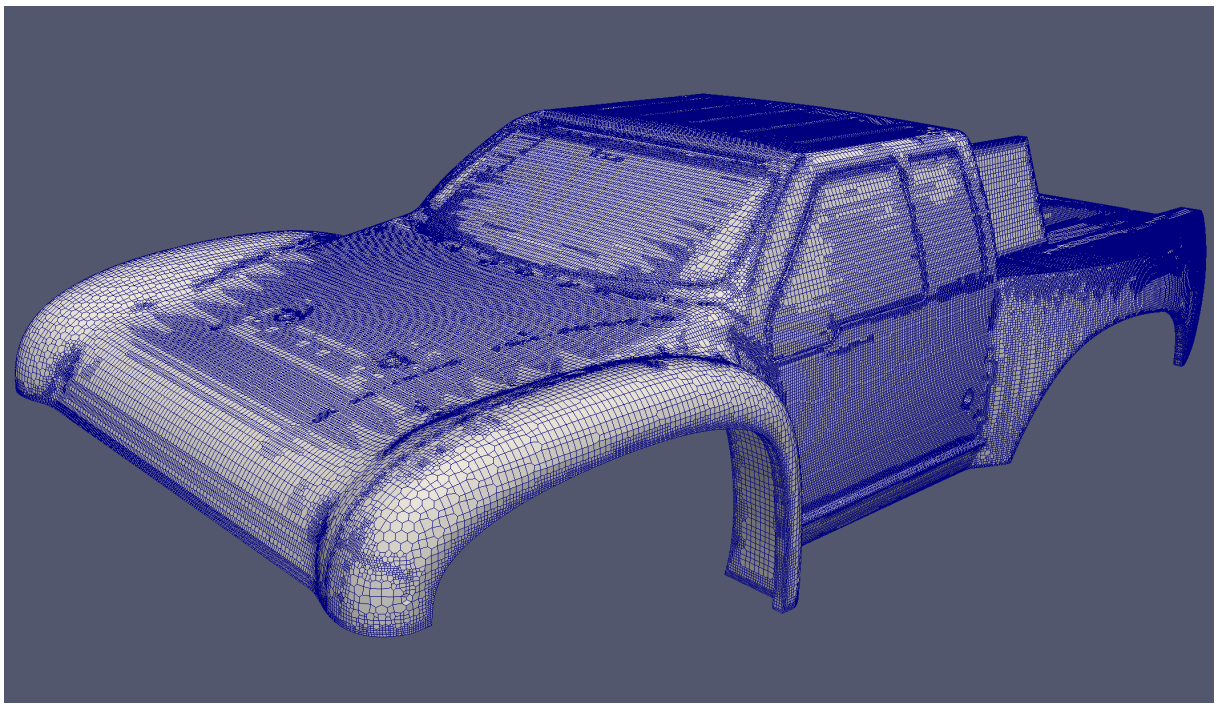


Figure 8. Computational grid of the body using ParaView.

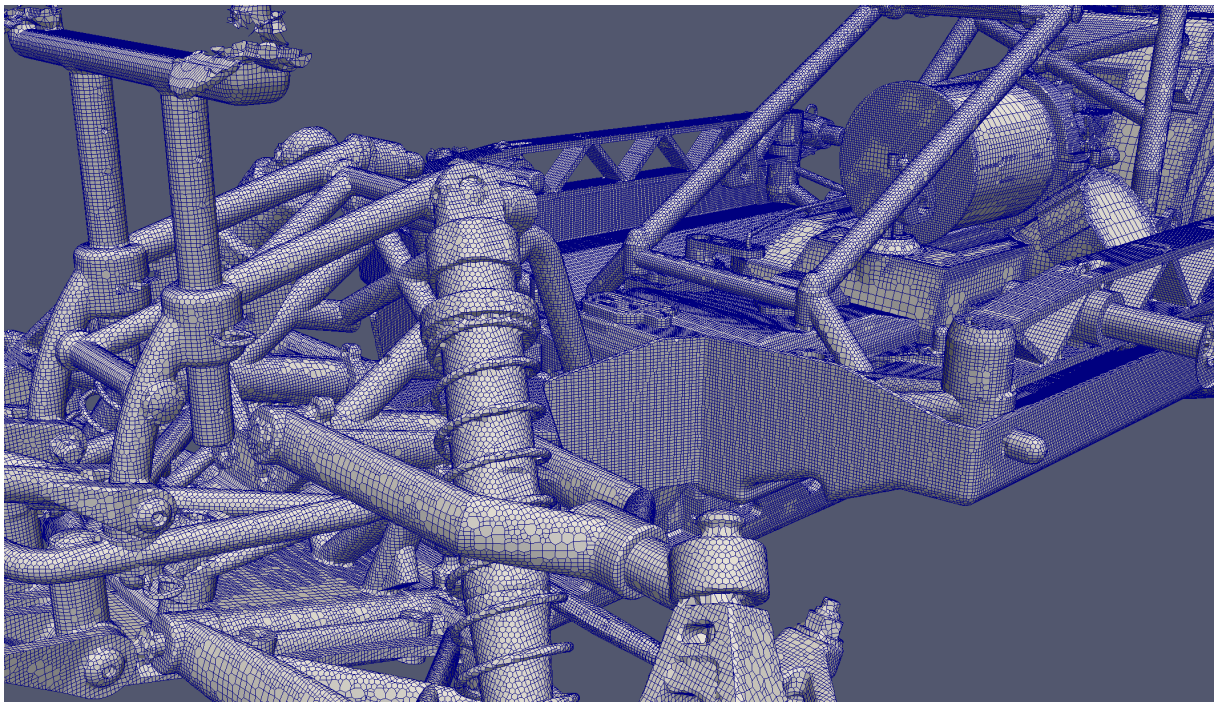


Figure 9. Computational grid of the front part of the frame using ParaView.

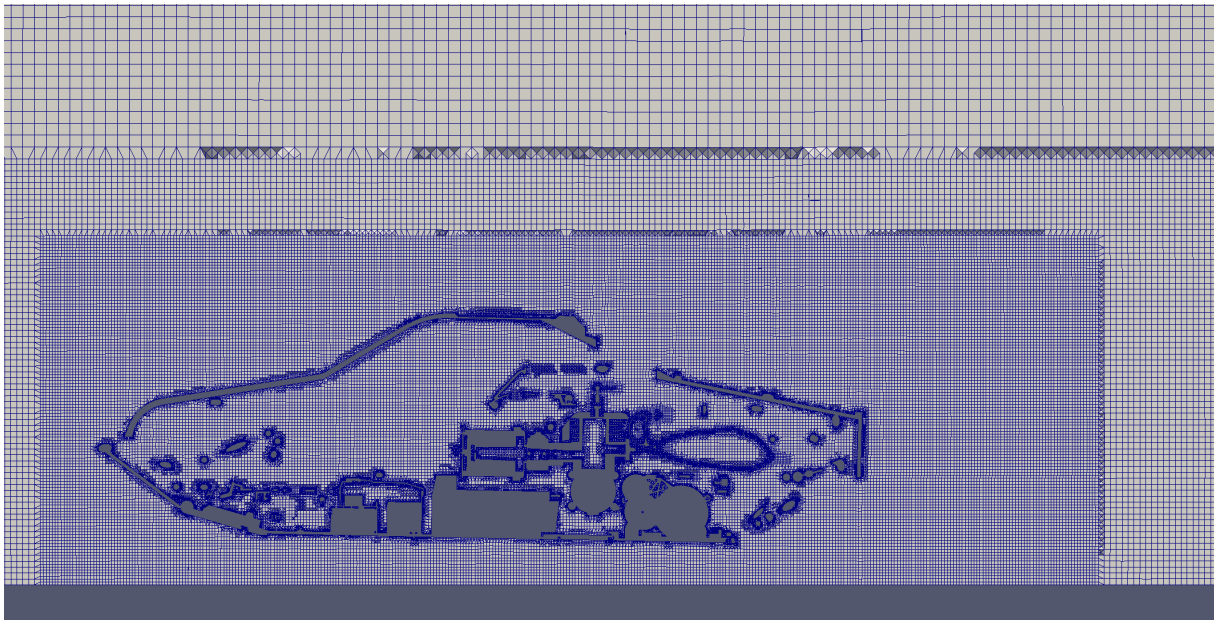


Figure 10. Computational grid in the midplane using ParaView.

the wheels are assumed to be fully rotationally symmetric and blocks of the tyres and the holes of the rims are not taken into account in that respect. Another option would be to use arbitrary mesh interface (AMI), but because of the added computational demand, it is not employed in this work.

In the simulation, an adaptive time stepping method was used which yielded a time step in the order of 16 μs when the Courant number was limited to be under 0.8. The simulation was first carried out for one second of simulated time to obtain statistically steady flow field. After that,

the simulation was continued for three more seconds (from 1 s to 4 s) and simulation data was gathered and time averaged in that time period.

The simulation was done on a modern Linux cluster using 10 computational nodes and 200 CPU cores. The overall computation wall time was 611 hours which translates to over 120,000 CPU hours. The amount of produced CFD data was about 8.6 TB, including all the solved variables. The data was saved at the interval of $\Delta t = 0.004$ s in simulation time in order to produce a 24 frames per second video with a slow motion factor of 10.

3.3 Post-processing, visualisation and rendering

Due to quite large mesh size, ParaView was used in server-client mode. Rendering server (pvrenderserver) or data server (pvdataserver) were not utilised but all the server computation was done in one server i.e. pvserver. Due to firewall restrictions, there are two possible ways to use server-client mode in case the server and client computers are in different network locations. One option is to use ssh tunneling technique and other is to use reverse connection method. The latter option is used in this work. In this method the ParaView client is started first in reverse connection mode. After that, the pvserver is started on the server computer by the command

```
pvserver --use-offscreen-rendering -rc -sp=11111 -ch=XXX.XXX.XXX.XXX
```

in which `-rc` means the reverse connection, `-sp` means the server port and

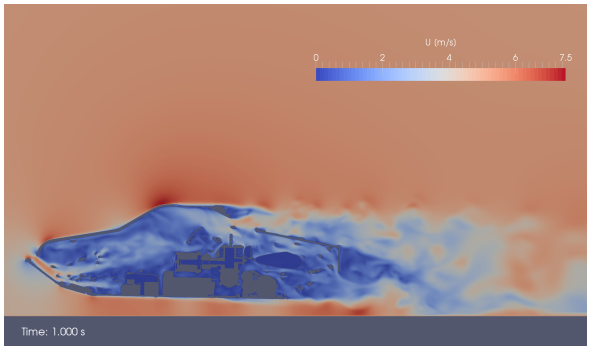
```
-ch=XXX.XXX.XXX.XXX}
```

means the client host i.e. the IP address of the computer where the client is running and `XXX.XXX.XXX.XXX` is replaced by the actual IP number. As soon as pvserver is started, the ParaView client recognises it and connection is established. It should be noted that the pvserver can be run as parallel just by running it using the usual `mpirun` command.

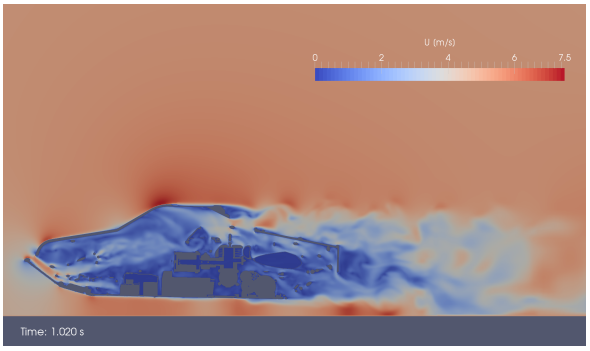
Unsteady nature of the turbulent flow can be seen clearly in the Figure 11 in which the velocity field at the midplane of the car is shown at the selected times. Same time dependency can be seen also in Figure 12 in which the 3D visualisation made using turbulent kinetic energy is shown.

Streamlines and ribbons colored by the velocity magnitude and rendered by Blender are shown in Figures 13 and 14. Export from ParaView to Blender was done using the X3D data format. Special care was needed to make colors of streamlines and ribbons work. Desired functionality was achieved using Attribute node in the Cycles rendering engine and using Col as a name of the color as can be seen in Figure 15.

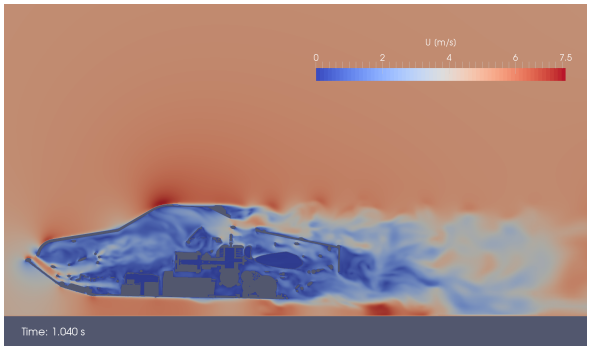
Rendering in Blender is a very time-consuming process. For example, rendering Figure 13 using quite modern NVIDIA Quadro K5000 graphics card took about four hours when GPU (graphical processing unit) and CUDA (Compute Unified Device Architecture) was used. On the other hand, parallelisation in Blender is extremely efficient and using eight CPU cores the rendering time is even slightly smaller than in the case of one GPU.



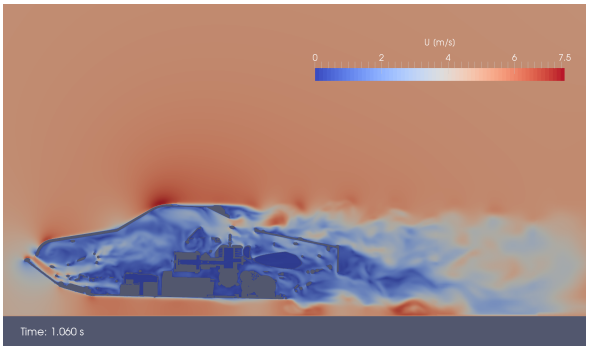
(a) t = 1.00 s.



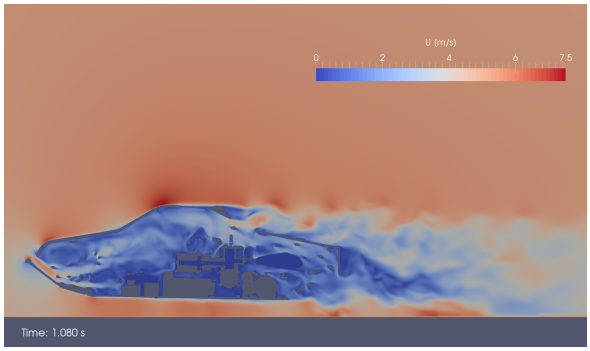
(b) t = 1.02 s.



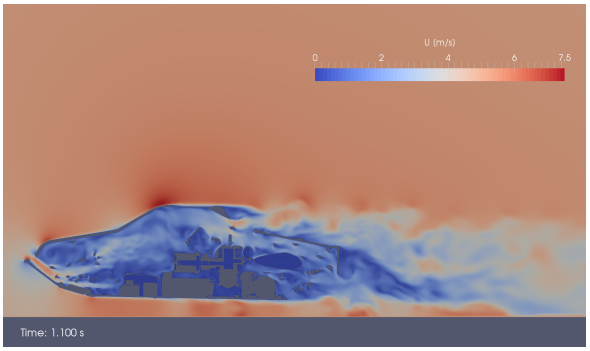
(c) t = 1.04 s.



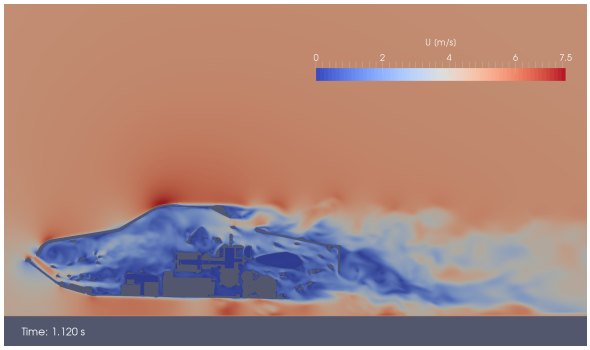
(d) t = 1.06 s.



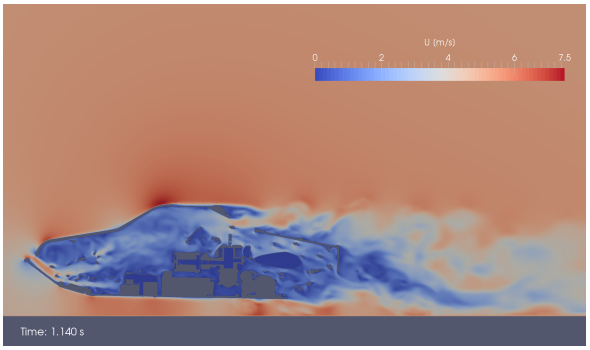
(e) t = 1.08 s.



(f) t = 1.10 s.

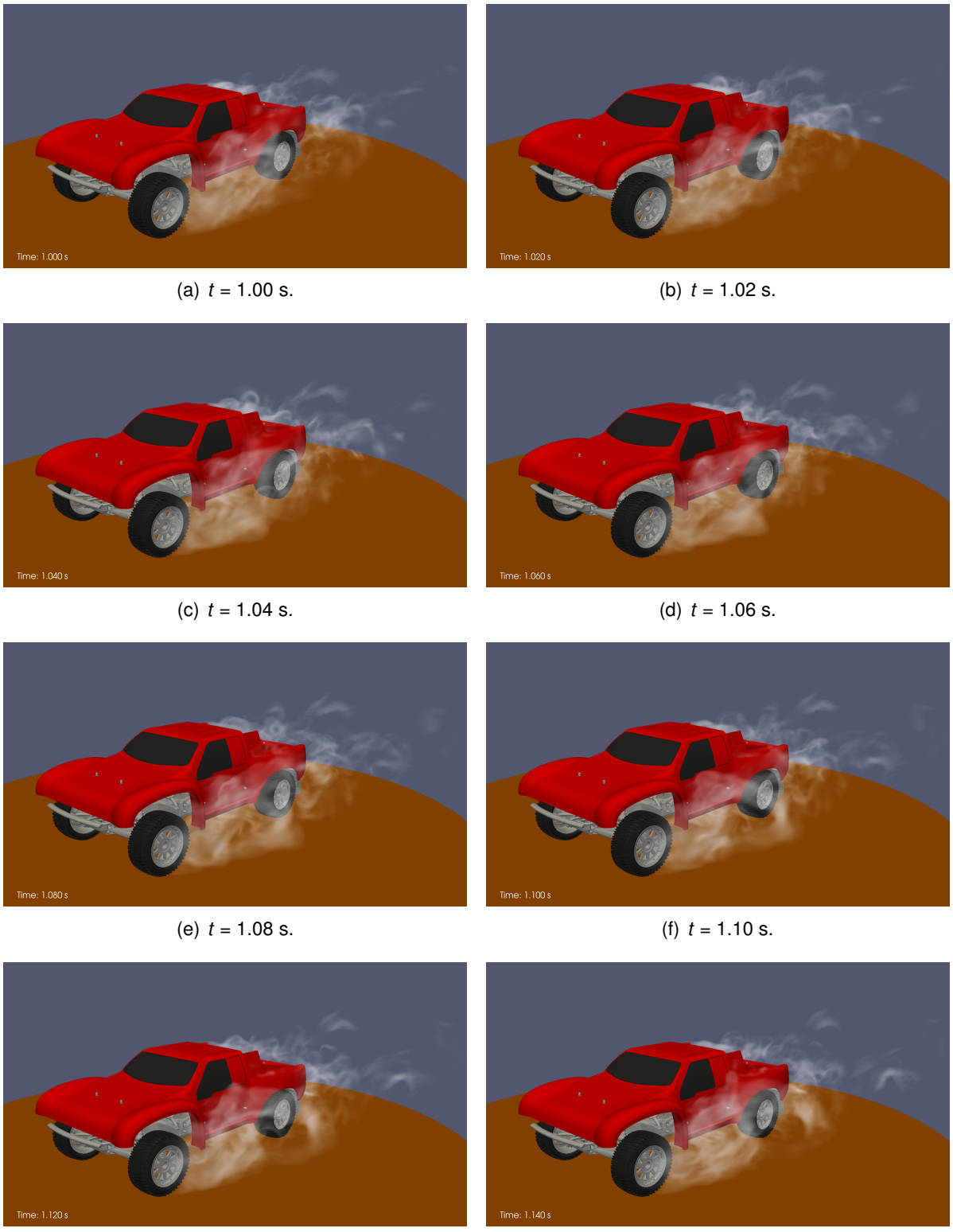


(g) t = 1.12 s.



(h) t = 1.14 s.

Figure 11. Snapshots of velocity field at the midplane using Paraview.



(a) t = 1.00 s. (b) t = 1.02 s.

(c) t = 1.04 s. (d) t = 1.06 s.

(e) t = 1.08 s. (f) t = 1.10 s.

(g) t = 1.12 s. (h) t = 1.14 s.

Figure 12. Visualisation using turbulent kinetic energy using Paraview.

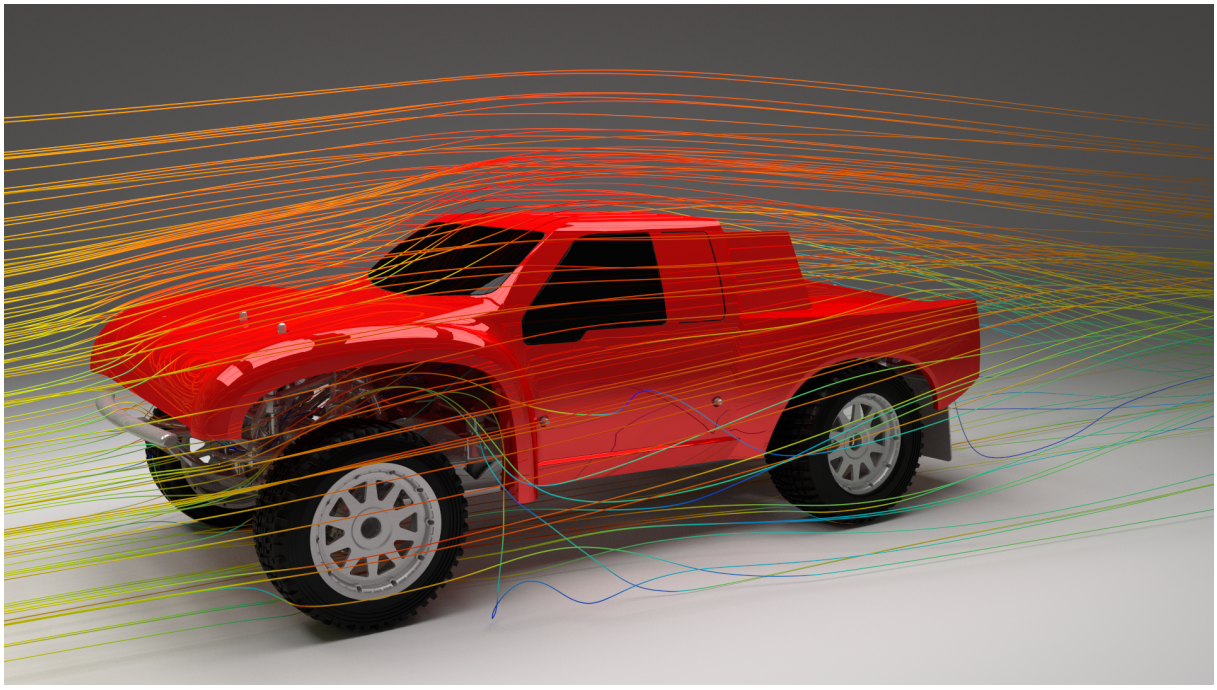


Figure 13. Streamlines (actually small diameter tubes are used to make them more visible) colored by velocity magnitude and rendered using Blender.

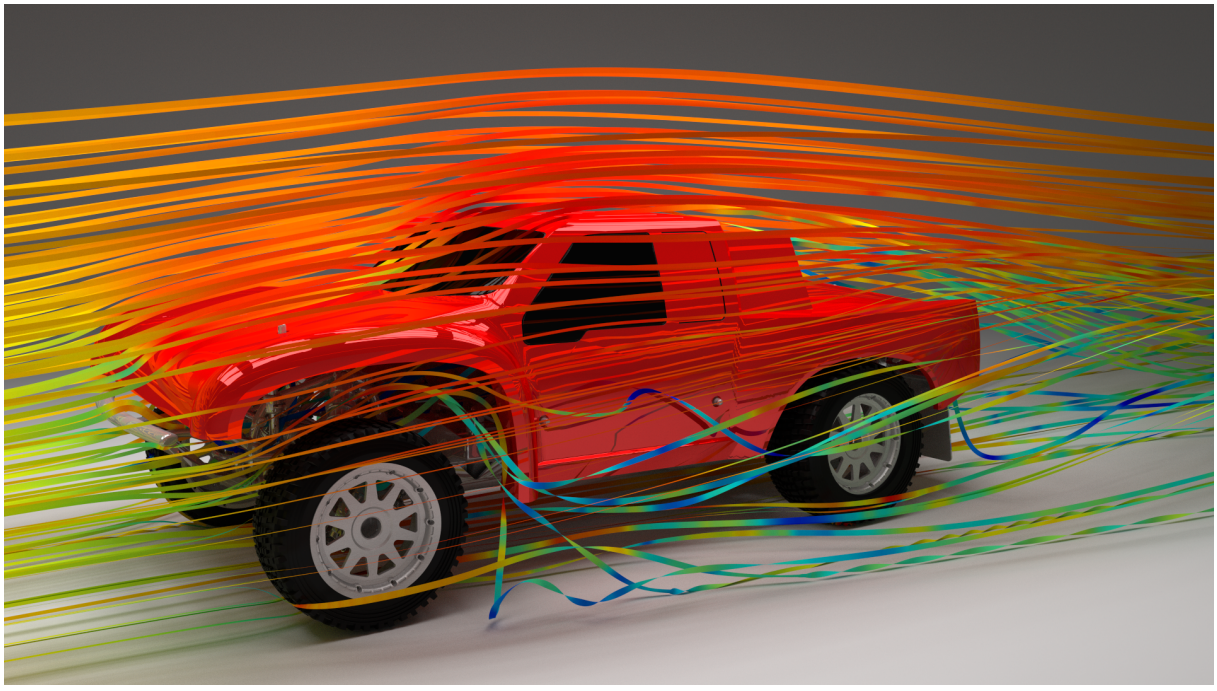


Figure 14. Ribbons colored by velocity magnitude and rendered using Blender.

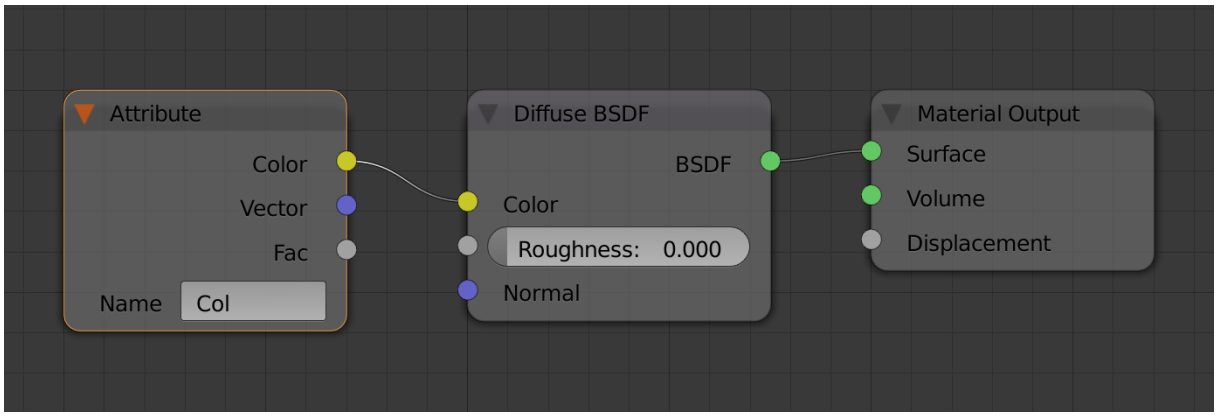


Figure 15. Attribute in the streamline and ribbon rendering in Blender.

4. Conclusions

Salome-platform – OpenFOAM – ParaView – Blender is very capable open source tool chain for parallel computational fluid dynamics simulation and post-processing and can produce photorealistic results. The rendering quality of Cycles Render rendering engine in Blender is on par with commercial software and produced images can be used for example in advertisement material.

The ability to use parallel post-processing in a server-client in ParaView is an important feature especially in large-scale simulations. Using server-client mode also minimises the amount of required data transfer between the computing cluster and workstation environment.

References

[1] Kitware, Inc. *VTK User's Guide*. Kitware, Inc., 11th edition, 2010.

[2] William J. Schroeder, Kenneth M. Martin, and William E. Lorensen. *Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. Kitware, Inc., 4th edition, 2006.

[3] Utkarsh Ayachit. *The Paraview Guide: A Parallel Visualization Application*. Kitware, Inc., 2015.

[4] Roland Hess. *Blender Foundations: The Essential Guide to Learning Blender 2.6*. Focal Press, 2010.

[5] John M. Blain. *The Complete Guide to Blender Graphics: Computer Modeling and Animation*. A K Peters/CRC Press, 2012.

[6] Y. Egorov and F. Menter. *Development and Application of SST-SAS Turbulence Model in the DESIDER Project*, volume 97, chapter Advances in Hybrid RANS-LES Modelling, pages 261–270. Springer Berlin Heidelberg, 2008.

[7] F. R. Menter and Y. Egorov. The scale-adaptive simulation method for unsteady turbulent flow predictions. part 1: Theory and model description. *Flow, Turbulence and Combustion*, 35(1):113–138, 2010.

A. Used open source software applications

- Salome Platfrom (www.salome-platform.org/)
- OpenFOAM (www.openfoam.org)
- ParaView (www.paraview.org)
- Blender (www.blender.org)
- Utilities
  - LibreOffice (www.libreoffice.org)
  - GIMP (www.gimp.org)
  - Libav (in video accompanying this report) (libav.org)
- Linux
  - Rocks Cluster Distribution (www.rockscluster.org)
  - CentOS (www.centos.org)
  - Ubuntu (www.ubuntu.com)