PARAVIEW TUTORIAL

FOR THE VISUALIZATION OF EARTH- AND CLIMATE SCIENCE DATA

Stand vom: August 6, 2014

CONTENTS

1 INTRODUCTION AND OVERVIEW

PARAVIEW is a free available 3D visualization software that is based on the visualization toolkit (VTK) and is primarily developed and published by Kitware Inc. Paraview is known and used in many different communities to analyze and visualize scientific data sets. This tutorial is based on Paraview version 4.1. and aims to introduce Paraviews concept and user interface, as well as its functionality specifically to climate science researchers. The tutorial is accompanied by several self-guided workshops, in which the user follows a sequence of steps to accomplish a certain visualization goal. Although the focus within this tutorial is on using climate science data sets, the concepts and procedures taught can of course also be applied in other areas. Besides being a how to tutorial for using Paraview, a second focus in this tutorial is to provide an introduction to several basic and advanced visualization techniques. At first, Figure 1 shows a complex visualization of an ICON ocean data set to demonstrates some of the possibilities for the analysis and visualization of climate data sets.

The viewport of Figure 1 is divided into two parts, in which the left one visualizes salinity values at the ocean surface from an ICON ocean simulation. The different salinity values are color coded; higher values of salinity are here depicted in red. The lattice and cell structure is visualized as well. The area of

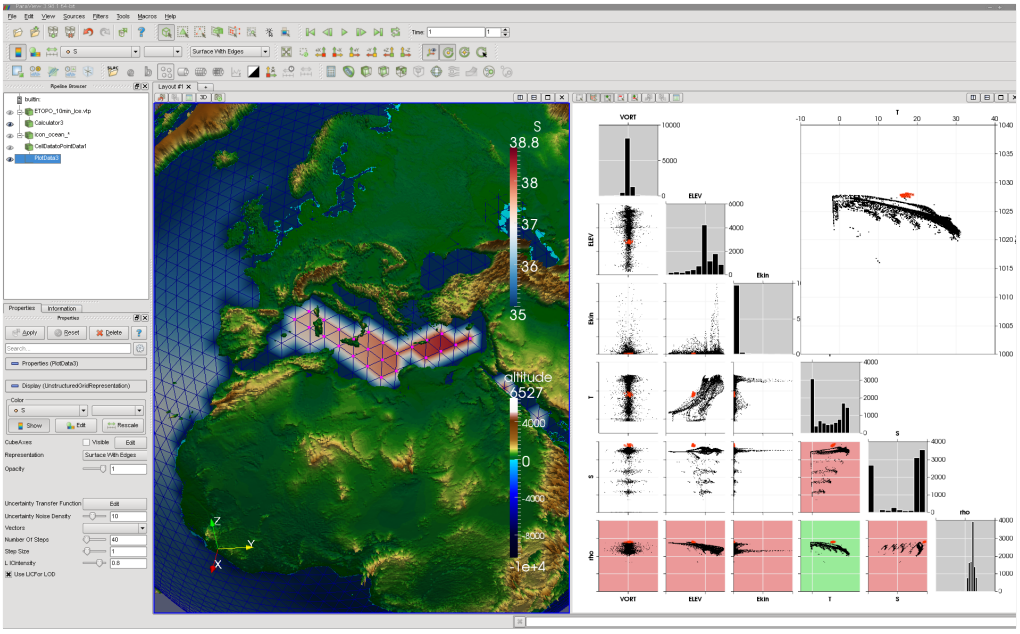

Figure 1: ICON Ocean Data in Paraview.

the Mediterranean Sea is select using the *Scatterplot Matrix* on the right hand side of the screen. In this scatterplot matrix, the dependencies between all variables within this data set are displayed using several different scatterplots. The largest scatterplot in the top right displays here the relationship between temperature and salinity. A selection has been made in this plot, which selects those values that have a both, a high salinity value as well as a high temperature. This selection is also visible within the other scatterplots and within the 3D view to visualize the position of the selected cells, as well as to highlight the relationship to other variables. All views can be linked together, which means that, if one changes either selection, both screens are updated accordingly. These features make Paraview a superb tool not only for data visualization, but especially for an interactive data exploration and analysis and model debugging.

Paraview itself is build upon the Visualization Toolkit (VTK), which is a large collection of C ++ classes used for 3D computer graphics, image processing and of course data visualization. The VTK supports a large variety of different data formats and grid structures, as well as implements the state of the art in algorithms for scalar, vector, tensor, volumetric and polygonal data processing and rendering, along with several advanced modeling techniques. Furthermore, the VTK comprises an information visualization framework, 3D gui elements for data selection and interaction, as well as supports parallel processing and rendering. Thereby the VTK is based on a visualization pipeline that is displayed in Figure 2[1].

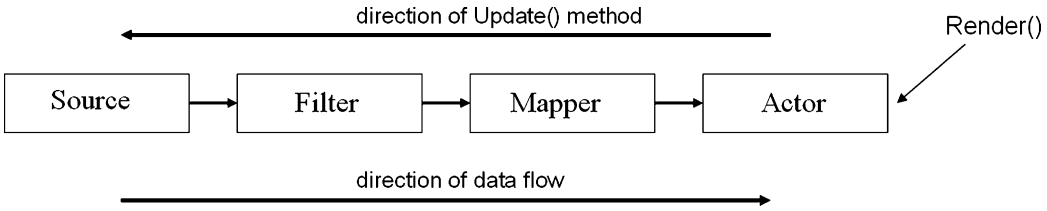

Figure 2: VTK Visualization Pipeline.

The visualization pipeline, as it is depicted in its simplified form in Figure 2, is THE essential structure of the visualization toolkit and Paraview. The data handling / flow is from left to right, the pipeline update commands are handled vice verse. A source can be a data reader or generator, from which the data is filtered and selected, then mapped into a geometric form for representation, and finally handled to an actor for interaction and visually rendered and displayed on the screen. This pipeline supports in its new realization also a demand driven update and rendering scheme, that is, only those parts of the pipeline / data are updated, that are really requested by the application. This allows a more efficient rendering of smaller subsets of large data sets.

1 http://openi.nlm.nih.gov/imgs/rescaled512/2039808_10278_2007_9062_Fig1_HTML.png

All these features and possibilities make Paraview an excellent choice for the visualization and analysis of climate science data sets. The outline of this tutorial is described in the next section.

1.1 OVERVIEW OF THIS TUTORIAL

This tutorial is divided into eight chapters:

INTRODUCTION AND OVERVIEW — This first Chapter 1 started with an overview of Paraview and briefly explained the underlying visualization pipeline. The second part of this chapter concentrates on an introduction of the user interface, some necessary data preparation, and creates a first simple 3D visualization by using a small ECHAM atmospheric data set. The visualization is further annotated by captions, a color bar, as well as an Earth's texture for reference.

SCALAR DATA — The second Chapter 2 discusses the available techniques for scalar data data visualization. Among these are parallel and oblique slicing, surface and iso-surface rendering, volume visualization, contouring and clipping, as well as general data filtering and processing techniques.

VECTOR DATA — After the visualization of scalar data, Chapter 3 demonstrates the available vector visualization methods, and explains how to combine two or three scalar fields to create one vector data set. This data is then mapped and displayed using stream- and streaklines, arrows & glyphs, as well as by using a line integrated convolution (LIC) texture.

INFO VIS — The following Chapter 4 diverts into the area of information visualization techniques, which are very powerful for an interactive data analysis and exploration. This includes scatterplot matrices, parallel coordinates, histograms and other data analysis tools. This chapter also discusses the principles of linking & brushing.

ICON DATA — Chapter 5 has a closer looks on loading and working with curvilinear and unstructured data sets.

ANIMATION — Visualizations are great to display results and to communicate ideas. Therefore, the following Chapter 6 discusses animation techniques, and explains the work flow to create annotated screenshots and animated movies.

PARALLEL RENDERING — Chapter 7 looks at possibilities for parallel data reading and visualization, in which Paraview is started in a Client/Server setting and connects to several Halo nodes for a more efficient parallel data reading, processing and rendering. Although the concepts are applicable anywhere, the examples in this chapter focus on DKRZ's infrastructure only.

IN-SITU VISUALIZATION — The last Chapter 8 examines an extension of Paraview (Catalyst), that allows a direct connection between simulation and visualization, and explores possibilities for in-situ data visualization.

1.2 THE USER INTERFACE OF PARAVIEW

The user interface of Paraview is depicted in Figure 3. It features a selection of menus and toolbars on top, the visualization pipeline on the top left, underneath the object inspector, and a big visualization window on the right hand side.

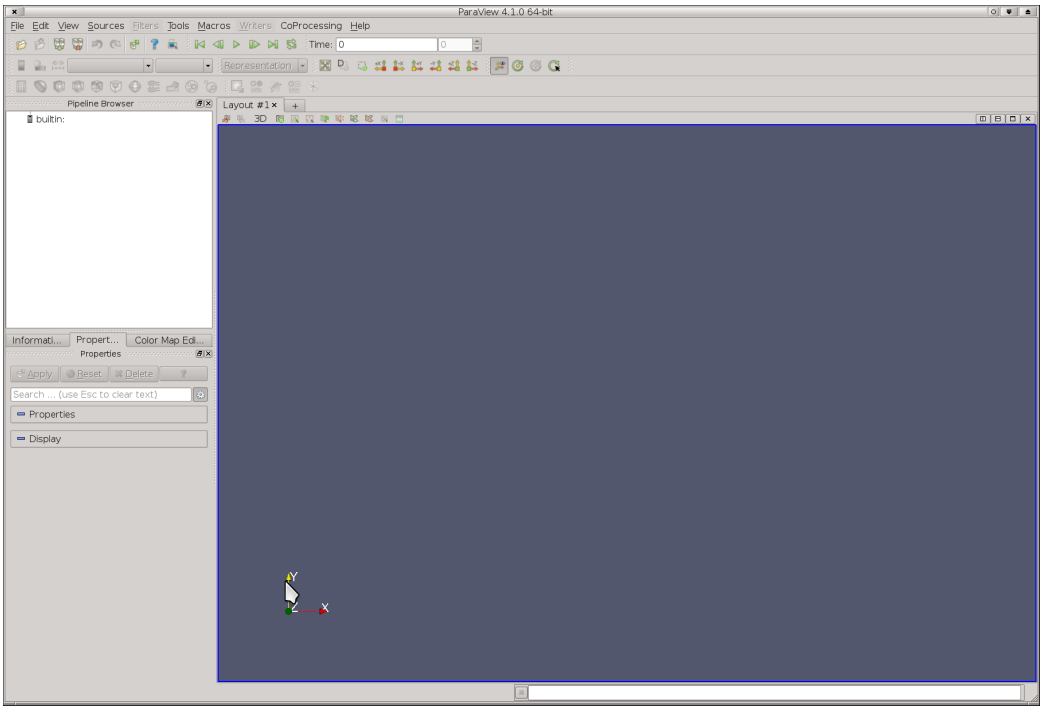

Figure 3: Screenshot of Paraview 4.1.

Paraview has in ten main menu items. These are: *File, Edit, View, Sources, Filters, Tools, Macros, Writers. CoProcessing* and *Help*. The File menu allows to open data objects (*Open* and *RecentFiles*), as well as to *Load States* and to *Save States*. A state, or session, within Paraview is the current setting of the visualization pipeline with all data sets loaded and modifications applied, as well as the status of all views. This is very convenient to reopen a saved session/state to continue working either on the same or a new data set. The File menu also allows to make and save a screenshot, as well as to connect and disconnect to/from a remote running Paraview server.

The Edit menu provides an *Undo* and *Redo* function, as well as a specification of general *Settings* and specific *View Settings*. The View menu allows one to enable/disable additional views, as well as to set the Toolbar. Paraview has a large variety of different toolbars and view. In this tutorial we will explore quite

a number of those, but are not able to discuss everything in the greatest detail. See here also the official Paraview tutorial on Kitwares website.

The Sources menu allows to add additional sources into the visualization pipeline, such as a sphere or a cone. The Filters menu lists all filters that are applicable to the current data set selected in bthe visualization pipeline. Filters that are grayed out can not be applied to the current data set, but often it is easy to convert the data by either resampling it to a different grid, or by remapping cell data to point variables or vice verse.

The Tools menu provides access to additional tools and plugins available, while Writers and CoProcessing add functionality for in-situ data analysis and visualization. The Help menu links to the help of Paraview.

1.3 CREATING A FIRST 3D VISUALIZATION

To familiarize yourself with the workflow of Paraview, we will now create a first, very simple, 3D visualization. For this, we will launch Paraview, load a small ECHAM netCDF data set, and display different variables in a straight forward manner. To start Paraview now, please log on to Halo and type in your console:

```
paraview
```

There are several different versions of Paraview installed in parallel, such as 3.98 or 4.01. A specific version can be selected by typing `paraview3.98`. Simply typing `paraview` will automatically execute and launch the latest, most recent, version of Paraview, which for now is Paraview 4.1.

If Paraview started correctly, you should see a graphical user interface, similar to Figure 3. Now go to the **File->Open** dialog to load a data set, and select from the tutorial data folder *ECHAMH_OM_A1B.nc*. In the following dialog you can choose between several different netCDF standards. Please choose here **NetCDF files generic and CF convention** and click on ok. After that, a new **Source** appears at the top of the visualization pipeline browser. Below this pipeline, you find the object inspector panel, which allows you to adjust some settings and provides information about the variables contained. Here you can choose to load the data in two different variations, either with a spherical representation, or with an equidistant cylindrical projection. Figure 4 shows both representations – parts of the spherical representation have been clipped away to illustrate the concept – , as well as the visualization pipeline and the object inspector dialog. In terms of dimensions and variables, you can choose between 2D (lat, lon) or 3D data variables (lev, lat, lon).

To continue with the tutorial, leave the checkbox **spherical representation** checked, select **lev, lat, lon** and click the button **Apply**. To get a first impression, switch in the menu bar of Paraview from an **Outline** to a **Surface** representation, and on the left select **temperature, t** as data variable, refer also to Figure 9. Now your visualization within Paraview should look similar to Figure 4. What you see is the temperature in the upper atmosphere. If you like, explore the different

variables and graphical representations that are available. In the later Chapters, we will show you how to *see inside* the sphere by using volumetric rendering and slicing techniques.

In the tutorial folder, we have saved this session as Paraview state *example_1a.pvsm*.

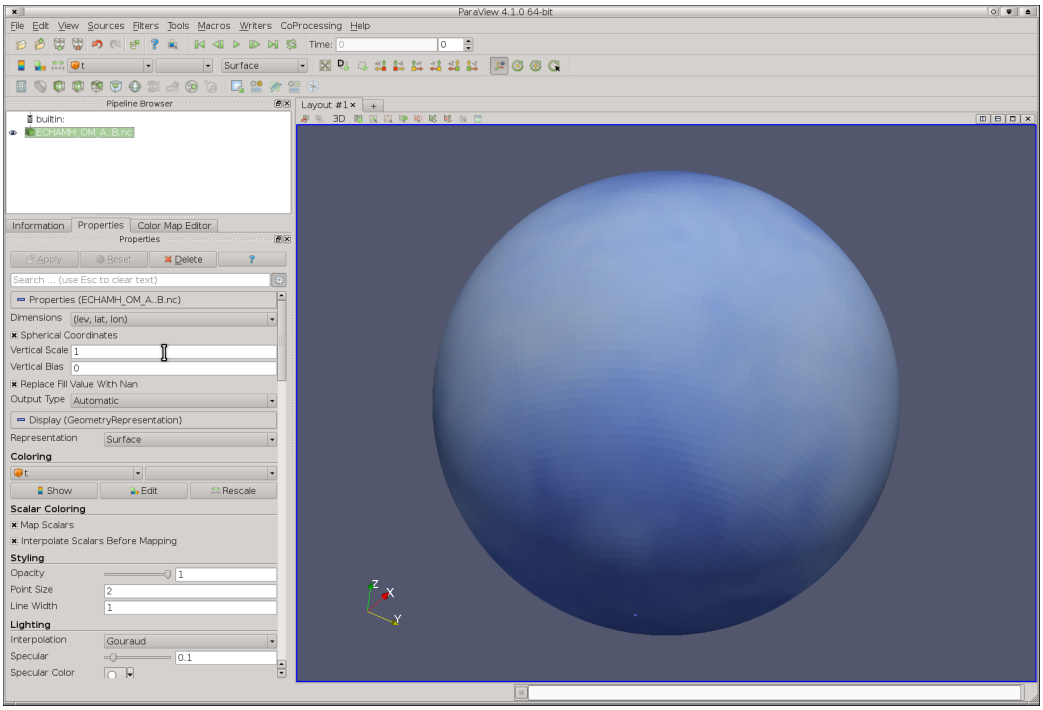

Figure 4: Paraview with ECHAM netCDF Data.

1.3.1 *Scaling Issues and Ordering*

The lattice description of the netCDF data is in general in latitude, longitude and meter. While lon/lat only ranges from 0 to 360, respective -90 to +90, the description of the z-axis is usually in meter or Pascal, and typically has quite high values. Unlike other visualization software, such as AvizoGreen, Paraview does not evaluate the dimensions of the axes, which results in a *wrongly* scaled z-axis if the netCDF data set is directly read in. Figure 5 illustrates the problem.

Here the left 3D view illustrates the problem, while the right 3D view shows the solution. To account for this problem, a different scaling factor for the z-axis has to be specified within Paraview. For this, we will apply our first filter to the data set. With the netCDF data in the pipeline selected, go the the Filter menu and scroll down to alphabetical selection. Select here the Filter **Transform** and change the scaling for the z-axis to 0.001, as is displayed in Figure 5. Click on the Apply button, and the problem is solved. This module also has handles for interactive scaling of all three axes. We do not need this now, but it might become useful

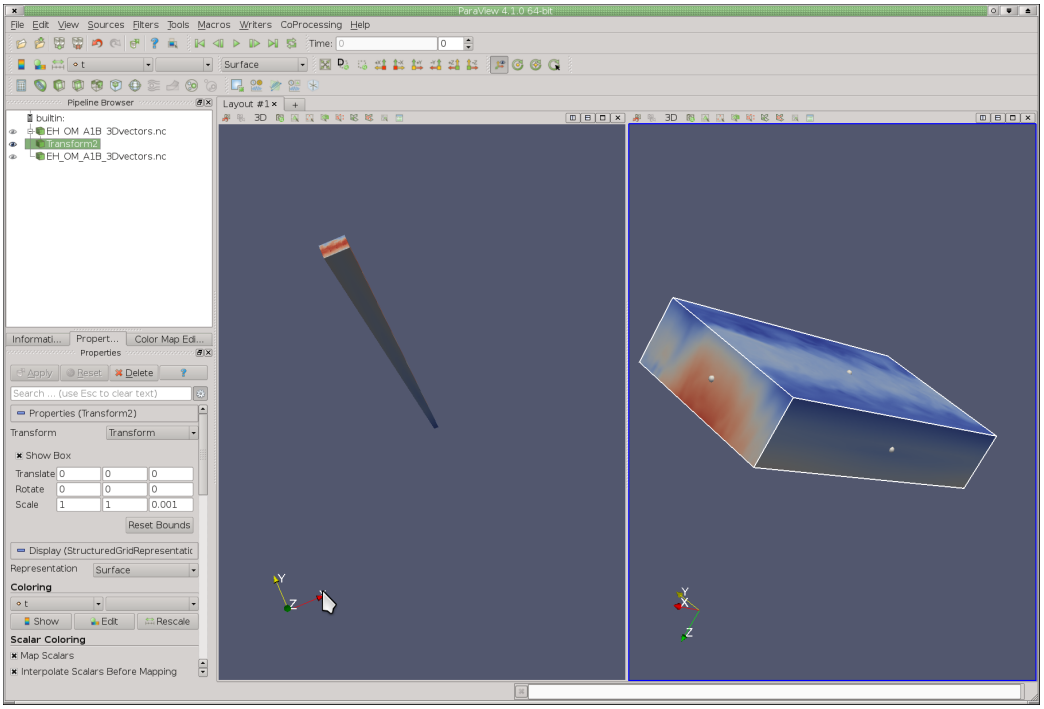

Figure 5: Paraview Z-Axis Scaling Issue.

later on. To disable and switch off these interaction handles, simply uncheck the box labeled **Show Box**. For your convenience, we have saved the Paraview state as *example_1b.pvsm* in the tutorial folder.

You can also apply this scaling factor in the netCDF module itself, but this method only works for spherical representations, not for an equidistant cylindrical projection. If the checkbox **spherical representation** is checked, you change the scaling factor from 1 to 0.001 directly underneath this checkbox to rescale the sphere. This feature will be used at the end of this Chapter. Please also note that the ordering of the z-axes is important and needs to be in the correct way. That is, the lowest level comes first and the highest last. If you encounter a problem, check if:

```
ncdump -v lev netcdf-file.nc
```

shows you a different ordering, e.g. the highest level first and the lowest last. If so, then use:

```
cdo invertlev netcdf-file_in.nc netcdf-file_out.nc
```

to reverse the ordering of the z-axis, and read in the file netcdf-file_out.nc.

1.3.2 *Color Tables*

A good and carefully chosen color table is of paramount importance for any good visualization, as a wrong color table might lead to artifacts and shows features

that are actually not contained in the data. Refer here to a nice article[2] by Maureen Stone, which is still up to date and starts with a famous quote by Edward Tufte regarding the importance of choosing an appropriate color table:

> *"... avoiding catastrophe becomes the first principle in bringing color to information. Above all, do no harm."*
> [Edward Tufte, Envisioning Information, Graphics Press, 1990]

Paraview automatically applies a default cold / warm (blue-white-red) color table to your data, but a different color table can easily be selected and / or created. For many variables, this color table is not a too bad choice. To change the color table of the currently displayed variable, scroll down to the **Coloring** section in the NetCDF object inspector of the last module. First we would like to add an annotation of our color table to our visualization. To do this, simply click on the button **Show**. If you don't like the position and size, feel free to drag this color bar around. To change the current color table, click on the button labeled **Edit**. A new dialog appears, in which you can change and adjust the color table. If you click on **Choose Preset** – the little icon with the heart symbol – , you see a wide variety of different color tables that are available, but for now, we keep the existing color table and simply adjust it a little bit.

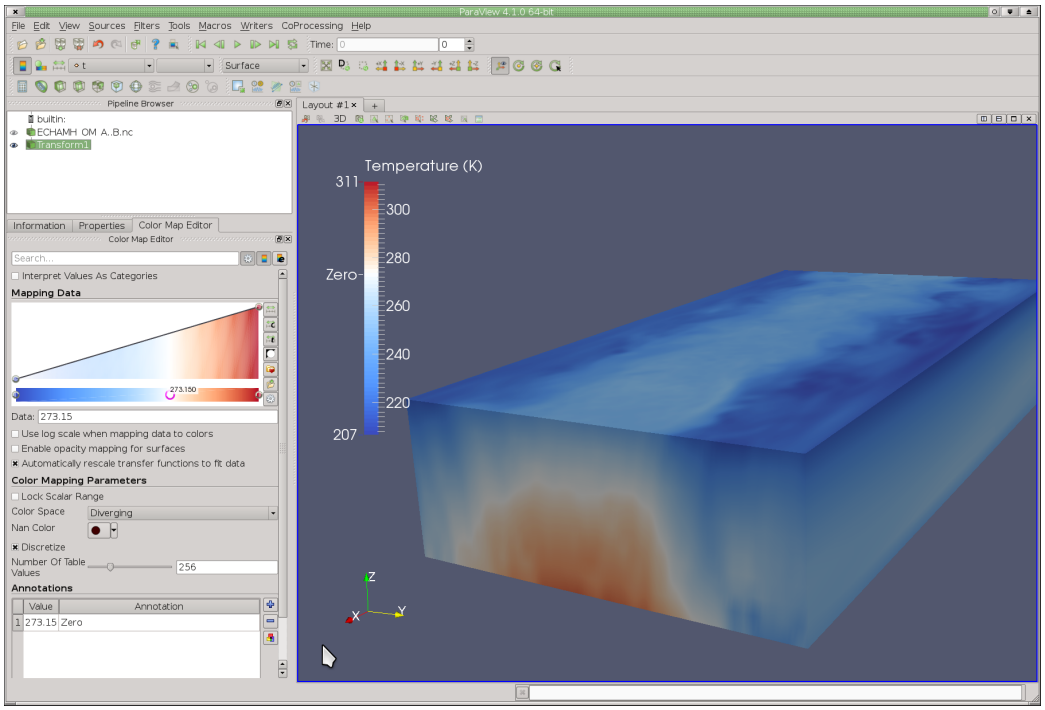

Figure 6: Adjusting the Color Table in Paraview.

Directly below the color bar, you notice several small circles. Using these, you can select a certain color and move it around. You can also add and delete colors

2 http://www.perceptualedge.com/articles/b-eye/choosing_colors.pdf

here. As we have temperature data displayed, we would like to highlight the zero degree C value, that is in your example 273.15 Kelvin. Therefore select the middle circle and drag it to this value. You can also use the text space to enter 273.15 for a more precise positioning. Now we chose pure white as color, so double click the circle, and change the color to pure white. Interact with the 3D view to explore the changes you have made. Now the areas which are cooler / warmer than zero degree C are clearly visible. Compare your visualization with Figure 6.

We can also highlight the zero degree C value in our color bar visualization. To do this, scroll down to *Annotations* and add here for 273.15 Kelvin the Text *"Zero"*. As the description of the color bar is not very expressive, we would like to change it. Therefore, click the little icon that says *Edit color legend properties.* This is the right most icon at the top of the color map editor. Here you can replace "t" with *"Temperature (K)"*. Compare your visualization with Figure 6. You can also specify and change opacity values within the color map editor, but we do not need this for this visualization. However, this becomes more important for other techniques, such as volume rendering, that are discussed in later chapters.

Compare your results with the example Paraview state *example_1c.pvsm*.

1.4 ANNOTATION WITH COASTLINES AND TOPOGRAPHY

The majority of climate simulations represent georeferenced data sampled over the globe. Therefore, it would be helpful to display the data using coastal outlines and / or the Earth's topography and bathymetry. As noted at the beginning of this Chapter, Paraview supports two different mappings / representations of the data. During the loading of the netCDF data set, you can decide whether you prefer a spherical, or an equidistant cylindrical representation.

1.4.1 *Equidistant Cylindrical Representation*

First we will have a look at an equidistant cylindrical projection. Either continue from the last session, or restart Paraview and load the state file *example_1c.pvsm*. To annotate this visualization with an background image of the Earth's surface, simply load the Paraview state file *earth.pvsm*. A dialog appears that asks you for the correct location of the file *earth.png*, which you will find in the Paraview tutorial folder as well. Browse to this location and click ok. After loading, you see a flat image of the earth, but the data set disappeared.

The visualization pipeline shows all objects that are part of this scene. Some of these are only tools to modify variables, but the majority of them can also be displayed in different views. If a module, along the data variables it contains, can be displayed, then there is a small eye left to its name. If the variable is currently displayed, such as in our case *earth.png* then the eye is black. If the module can be displayed but is currently switched off, then the eye is grayed out, such as for the *Transform* module. Simply click on the eye to display this module. Now

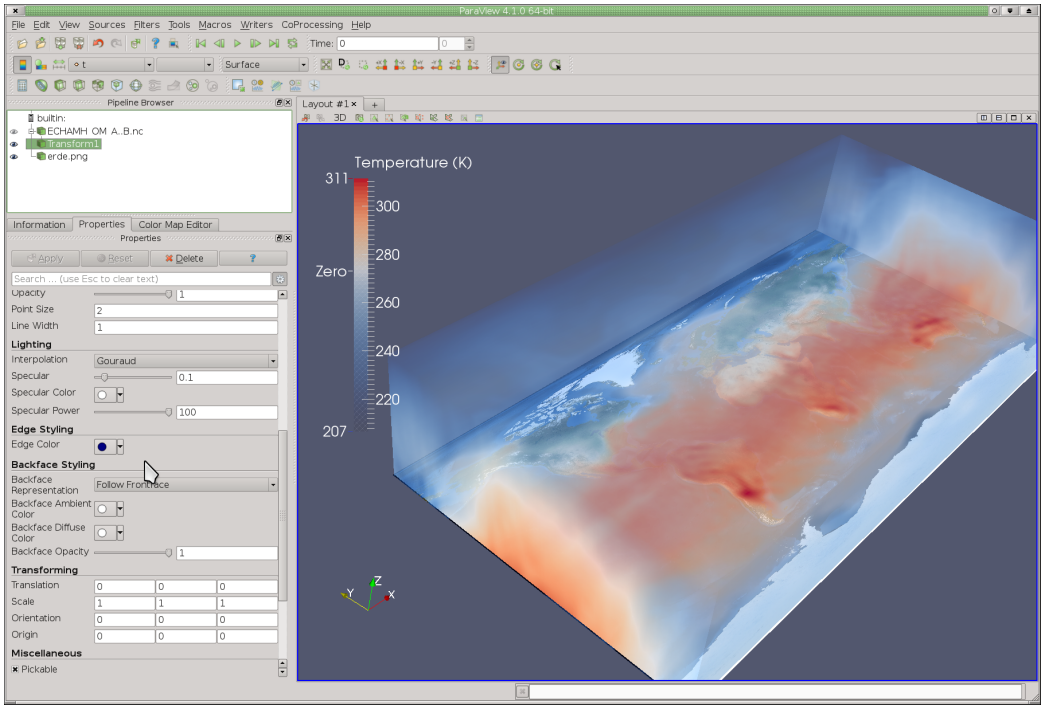

Figure 7: Equidistant Cylindrical Projection with Earth Surface.

the Earth is hidden by the data, so we would like the data to be displayed using transparency. Simply select the data (*Transform*) and go to the color map editor and here enable transparency by checkmarking the box *Enable opacity mapping for surfaces*. Then your visualization should look similar to the one in Figure 7. Please note that in this visualization, only the outer box of your date is displayed. This is not yet a volumetric visualization of your data.

Compare your results with and the Paraview state *example_1d.pvsm*.

Figure 8: Paraview Viewport Menu.

In order to make the work with the 3D viewports a little easier, Paraview has several quick access menus, such as the one displayed in Figure 8. The icons are more or less self explanatory, the first ones allow you to zoom out to show everything and to zoom fit to the data selected, while the latter ones orient the data along the positive / negative coordinate axes and allow a more precise rotation around a specified angle.

Another important menu is depicted in Figure 9. The first three icons are related to the color table and give you quick access to this menu and its submenus, while the last three drop down menus lat you select a variable for display and the method of representation. So far we have worked with *Outline* and *Surface*,

but feel free to explore the other options as well. We will have a closer look at the other options in the next chapter.

Figure 9: Paraview Data Representation Menu.

1.4.2 *Spherical Projection*

There are several possibilities to annotate the visualization for spherical projections. Unfortunately, the module that allows to texture map a sphere is a little bugy, but it is nevertheless possible to use it as well. Therefore, reset Paraview, go to the **Source** menu and create a higher resolution sphere. Change here the default settings from 8 to around 50 and click the **Apply** button. You have created a sphere. Now navigate to the Filters menu, and select under alphabetical the Filter *TextureMapToSphere*. In the properties dialog of this module, scroll down a bit, and chose under **Texture** to Load tge file erde_flipped.png. As the texture coordinates are calculated in a wrong way, we need a flipped earth for a correct display. Additionally, we have to adjust the scaling a little bit, therefore, scroll down to the **Transform** section and change the scaling from $1,\ 1,\ 1$ to $-2,\ -2,\ 2$. Compare your results with Figure 10 and also with the included example Par-

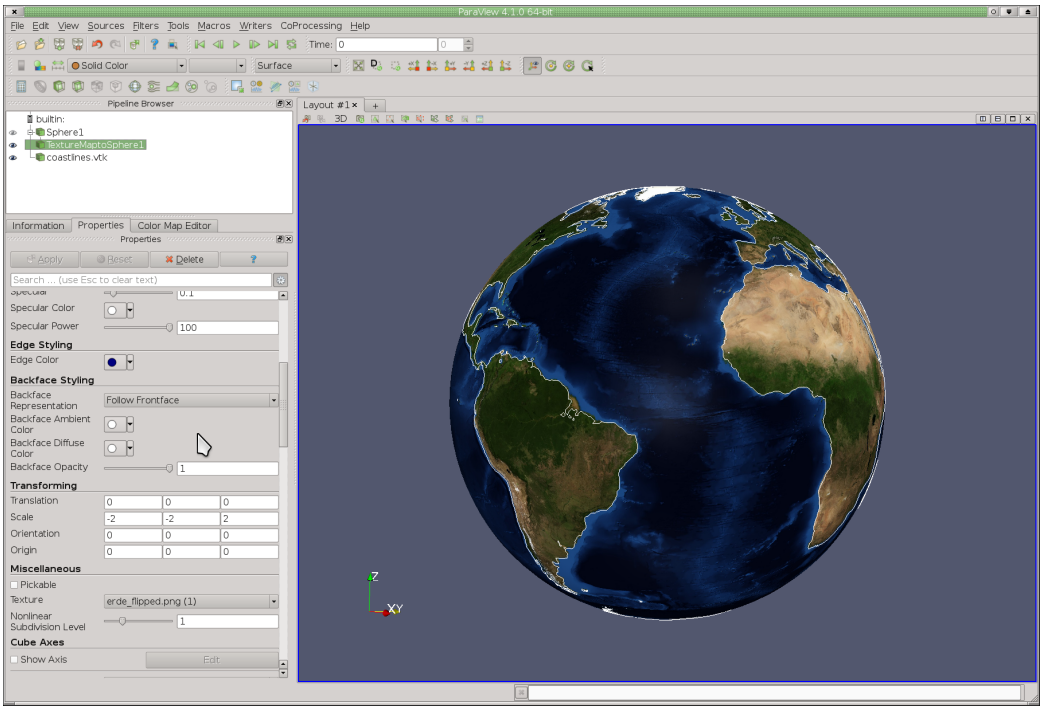

Figure 10: Spherical Projection with Orography, a height-dependent a color coding and coastal Outlines.

aview state *example_1e.pvsm*. The visualization in Figure 10 has also loaded and displayed coastal outlines.

1.4.3 *Including Orography & Bathymetry*

If you would like to include the orography of the earth in your visualization, then there is an alternative approach, compare with Figure 11. Please reset Paraview and load the file *orography.vtp*. This file contains a large texture that encodes the height and depth of the earth surface. After clicking on apply, the texture is loaded and displayed using Paraview's default color table. Now we need to extrude the surface using the variable *altitude*. This is accomplished by using the calculator module. Please select this module from the toolbar, and also click on the gear symbol to bring up the advanced properties for this module. Here you select the checkbox **Coordinate Results**, and in the calculator field you enter the following formula:

$$(1 + (\text{altitude} \;/\; 6370000) * 100) * (\text{iHat} * \cos(\text{asin}(\text{coordsZ})) * \cos(\text{atan}(\text{coordsY} \;/\; \text{coordsX})) * \text{coordsX} \;/\; \text{abs}(\text{coordsX}) + \text{jHat} * \cos(\text{asin}(\text{coordsZ})) * \sin(\text{atan}(\text{coordsY} \;/\; \text{coordsX})) * \text{coordsX} \;/\; \text{abs}(\text{coordsX}) + \text{kHat} * \text{coordsZ}) \tag{1.1}$$

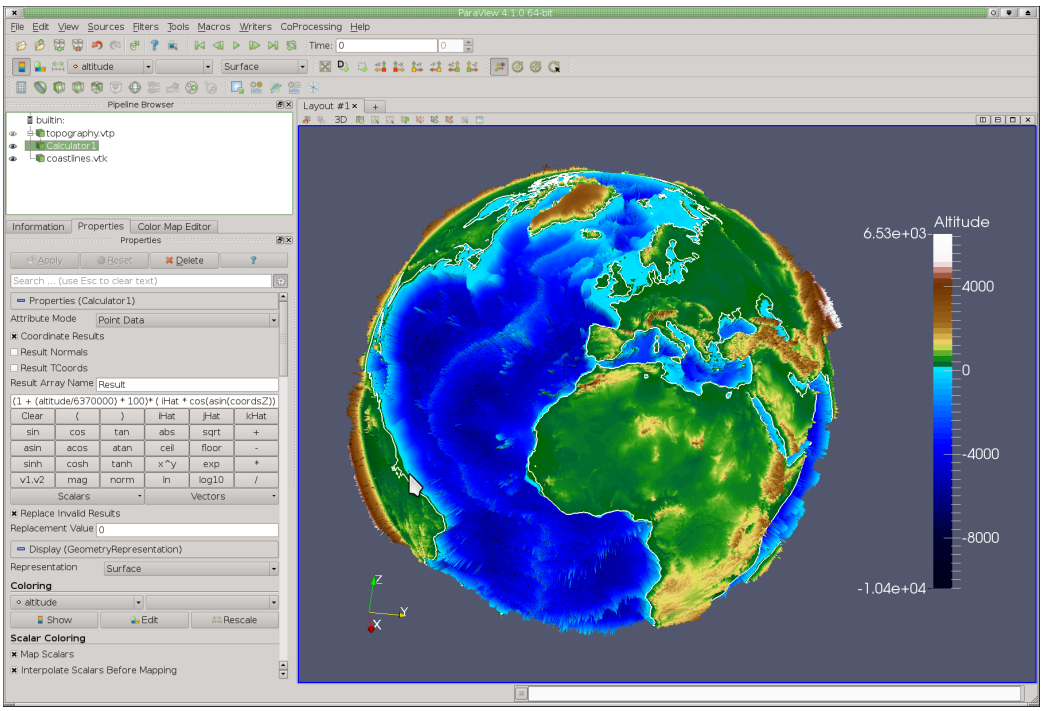

Figure 11: Spherical Projection with Orography, a height-dependent a color coding and coastal Outlines.

and click on the button **Apply**. Equation 5.1 simply performs a displacement mapping, and re-arranges the mesh to accommodate the correct height given by altitude. See also the left viewport of Figure 11 for comparison. Although we can not yet use a texture to color this earth, we can use a height scaling to at least approximate this. Therefore, go the the coloring section of the module, click on the button **Edit**, and load the color table *colormap_orography.xml* from the tutorial folder. If you also add an annotation for the color bar chosen, your visualization should look similar to the one in Figure 11.

It is now relatively simple to further enhance the visualization by adding coastlines. To add coastlines to your view, simply load the shape file *coastlines.vtk* from the tutorial folder and click on **Apply**. If you'd like, you can increase the line with of the rendering to 2 or 3. Compare your results with the included example Paraview state *example_1f.pvsm*.

For a last example, and to demonstrate how easily visualizations are created within Paraview, we provide a short glimpse into the next Chapter. Either continue from the previous example, or load the state file *example_1f.pvsm*. Now load the netCDF file *ECHAMH_OM_A1B.nc*. Leave the checkmark for spherical projection, and change the scaling underneath from 1 to 0.00004 and click on the button **Apply** to load the data as before. Now a scaling has been applied in the z-direction to accommodate the size of the earth orography. Now we select the variable specific humidity (q), but this time, we change the color table a bit to

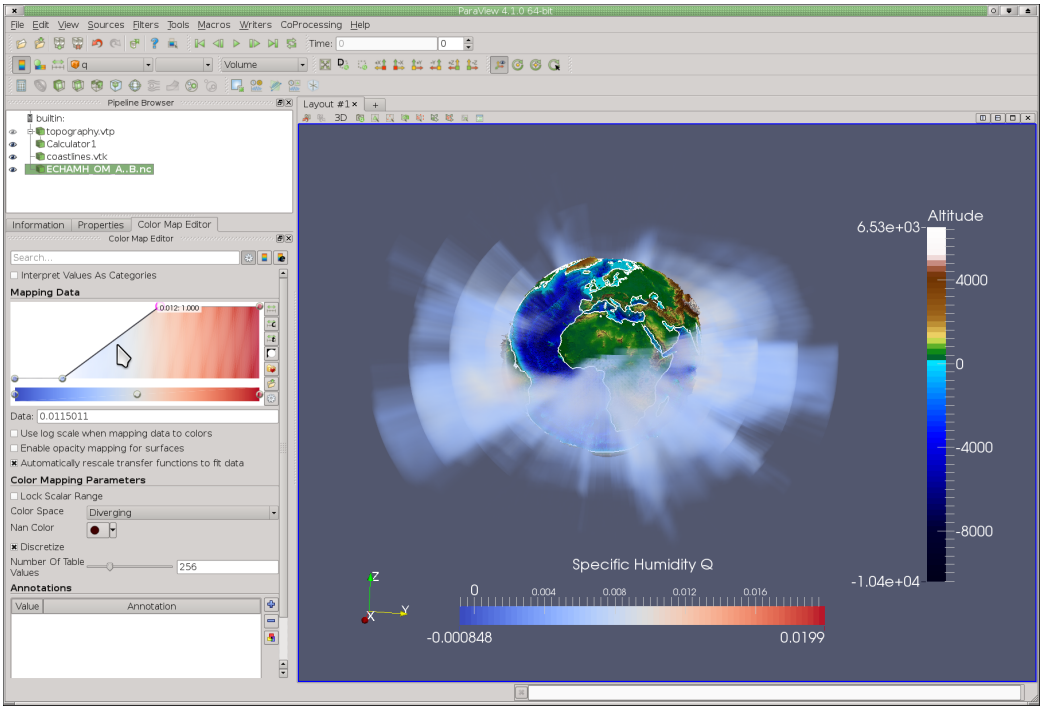

Figure 12: Spherical Projection with Orography, a height-dependent a color coding and coastal Outlines.

what is seen in Figure 12. As we would like to add transparency, we blend out the lowest values, and make the higher ones a little more opaque.

Now we change the representation from surface to volume, to show the entire data set, refer to Figure 12. What is shown now are the data points with a high specific humidity. The artifacts that are seen result from the low resolution of this example data. Compare your results with the included example Paraview state *example_1g.pvsm*.

2

SCALAR DATA VISUALIZATION

THE majority of climate data has either or a mixture of 2D / 3D scalar and / or vector variables. This Chapter looks at the visualization of 2D and 3D time-varying scalar data quantities. As examples we will use ECHAM climate simulation data, emanating directly from simulations run at DKRZ.

The data is usually stored and sampled at different locations. Most of the data is sampled directly at the cell center, whereas other variables might also be sampled at the cell vertices, or – in the case of velocities – at the center of the cell edges or cell sides. Paraview supports data that is sampled at the cell center (cell data) and at the cell corner (point data). It is easily possible to convert the data between those two directly within Paraview. The sampling distance for the z-axis has to be the same though, and if not, the data can to be interpolated before using them in Paraview with cdo. This might be the case, in which the u and v velocities are at one level, and the vertical velocity is sampled half way in between. For pre-processing of rotated grids it is also required to de-rotate the values prior to the visualization. For a more detailed discussion about these,

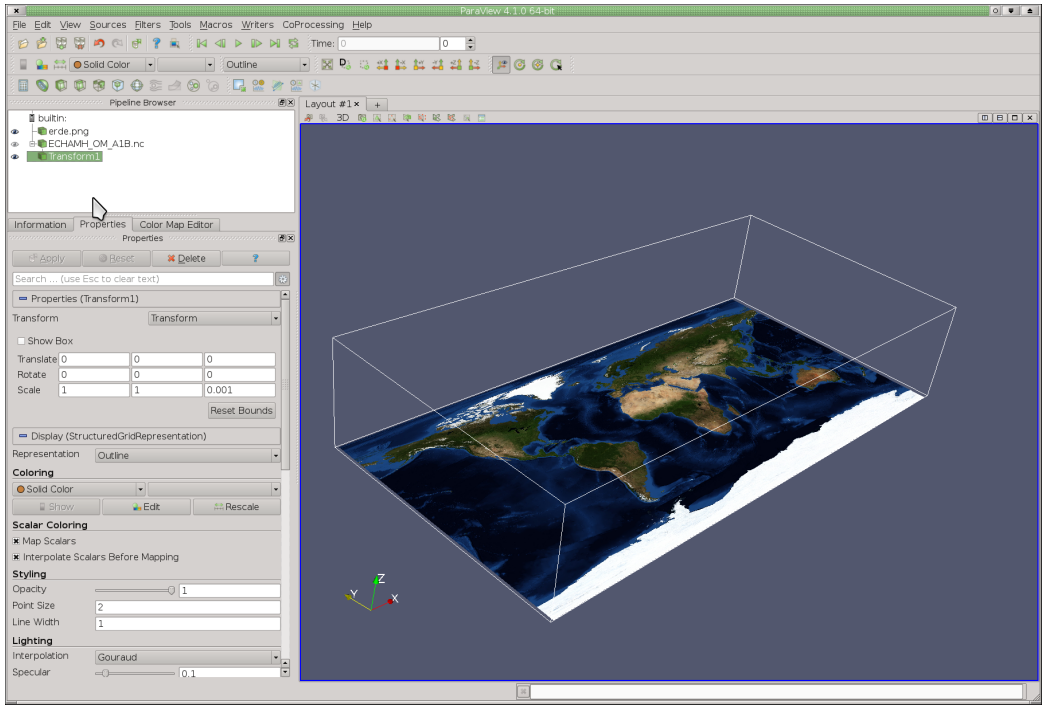

Figure 13: Paraview with NetCDF ECHAM Data, rendered as Outline.

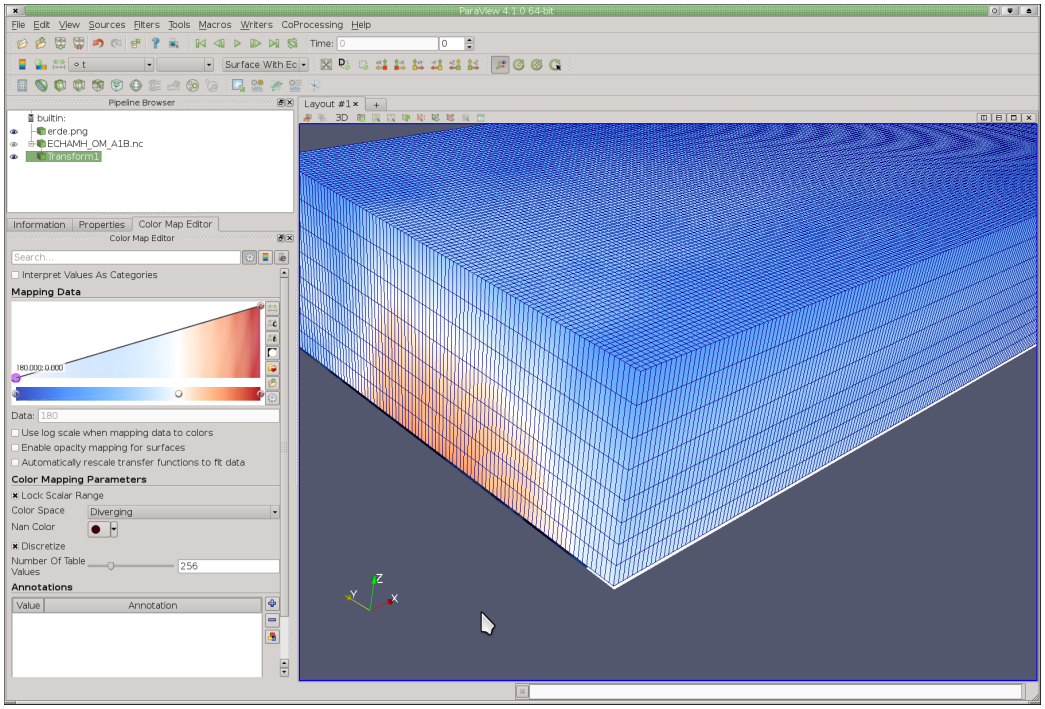

Figure 14: Paraview with NetCDF ECHAM Data, rendered as points.

we refer to the CDO tutorial hosted by ZMAW, as well as the Avizo Green and SimVis tutorials, which are hosted on the DKRZ website[1].

To get started with this tutorial, simply start Paraview and select from the menu **File -> LoadState** and select the Paraview session *earth.pvsm*. Please also load, similar to the last Chapter, the netCDF data set *ECHAM_OM_A1B.nc*. Please load the data as equidistant cylindrical, i.e. not spherical, and load the 3D variables, i.e. lev, lon, lat, and apply a Transform filter with a scale if the z-axis of 0.001, see the last Chapter if you are unsure about this procedure. After this, Paraview should look similar to Figure 13. Compare your results with the included example state *example_2a.pvsm*.

In the last Chapter you have already seen and worked with many different rendering methods, such as Surface and Volume Rendering, or the Outline as is depicted in Figure 13. Sometimes it is also desirable to visualize the grid at the same time. This can be done by changing the representation to a Wireframe rendering or by using the option Surface with Edges, as is displayed in Figure 14.

2.1 SLICING AND CLIPPING

The visualization of scalar data often displays only one variable at a time, such as temperature or humidity, using a slicing plane that can interactively be moved through the 3D data set by the user. This technique is especially well suited if one

1 http://www.dkrz.de/Nutzerportal-en/doku/halo/sw

has a large 3D data structure, and one would like to see the inner areas. Except for some advanced rendering techniques, Paraview often only visualizes the outer boundaries, such as for Surface, Point or Wireframe Rendering, compare with Figure 13 and Figure 14. To see the inner structure, one can either use a slicing plane to cut through the data, or one can clip portions off the data and visualize the remaining data points. In the following, we will do both.

2.1.1 *Slicing*

Please either continue the session from above, or load the included example Paraview state *example_2a.pvsm*. For slicing the data, we use a 3D plane that cuts through the data, and visualizes the data points at this position. Slicing is often performed along one of the major axes, but sometimes it is also required to slice the data arbitrarily using an oblique slice.

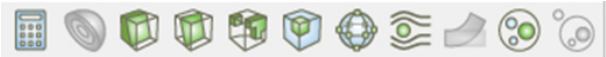

Figure 15: Paraview Visualization Modules.

If required, please change the representation of the data back to surface rendering. Now go to the main toolbar, and add a slice filter to the pipeline, and click on ok. The slicing module is the fourth icon from the left of Figure 15.

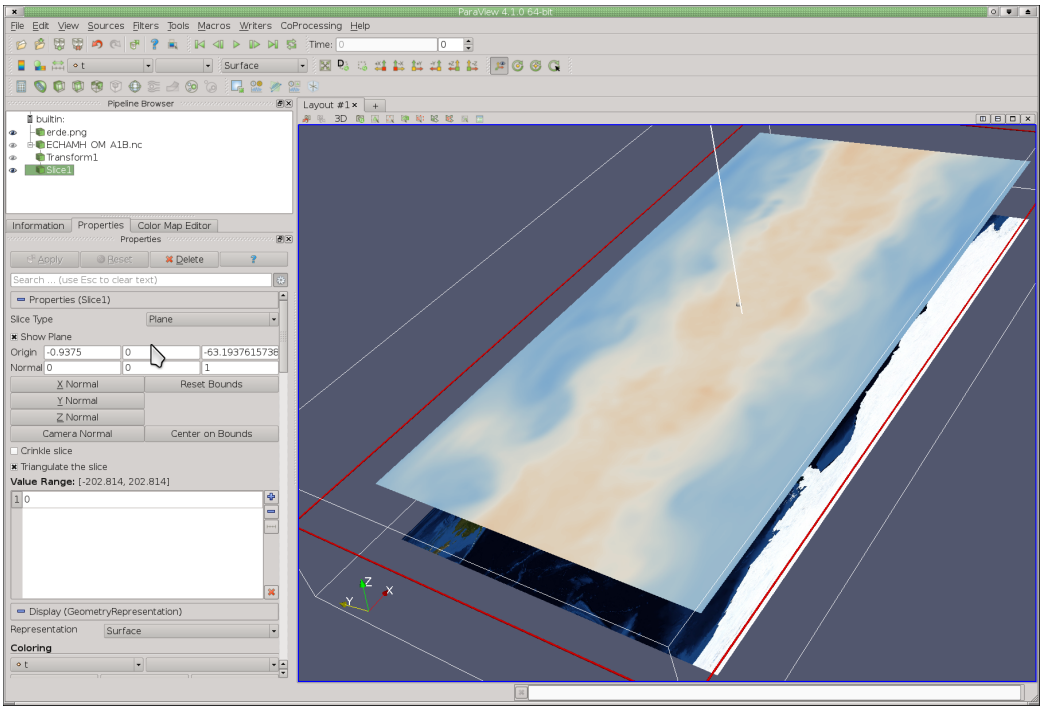

Figure 16: Slicing ECHAM data in Paraview.

The other icons are: Calculator, Contouring, Clipping, Slicing, Threshold, Subset, Glyph, Streamlines, Warp by Vector, Group Datasets and Extract Level. The resulting visualization is depicted in Figure 17. The object inspector displays several parameters that can be changed, for instance, the direction of slicing. Please click on **Z-Normal**, then click on the red box, move it a little bit, and click on apply.

Please play a little bit with all slicing directions and move the slicing plane to different positions. For an oblique slice, grab the handle of the long red line and move this around. That way, you can specify an arbitrary slicing angle. Also evaluate the function of the checkbox **Crinkle Slice**. Here the data is not interpolated, only those voxels are visualized directly that are sliced by the cutting plane. Compare your results with the included example Paraview state *example_2b.pvsm*.

In Paraview it is also possible to create several slicing planes at the same time. Now remove all but one slicing plane. In order to slice the domain at a specific interval, scroll down a little bit in the object inspector of the slicing module, and click on the button **Add Range**. Leave the values for this example as they are, and click ok and on the button **Apply**. Now you have created 10 slicing planes which slice through the data at a given user specified interval.

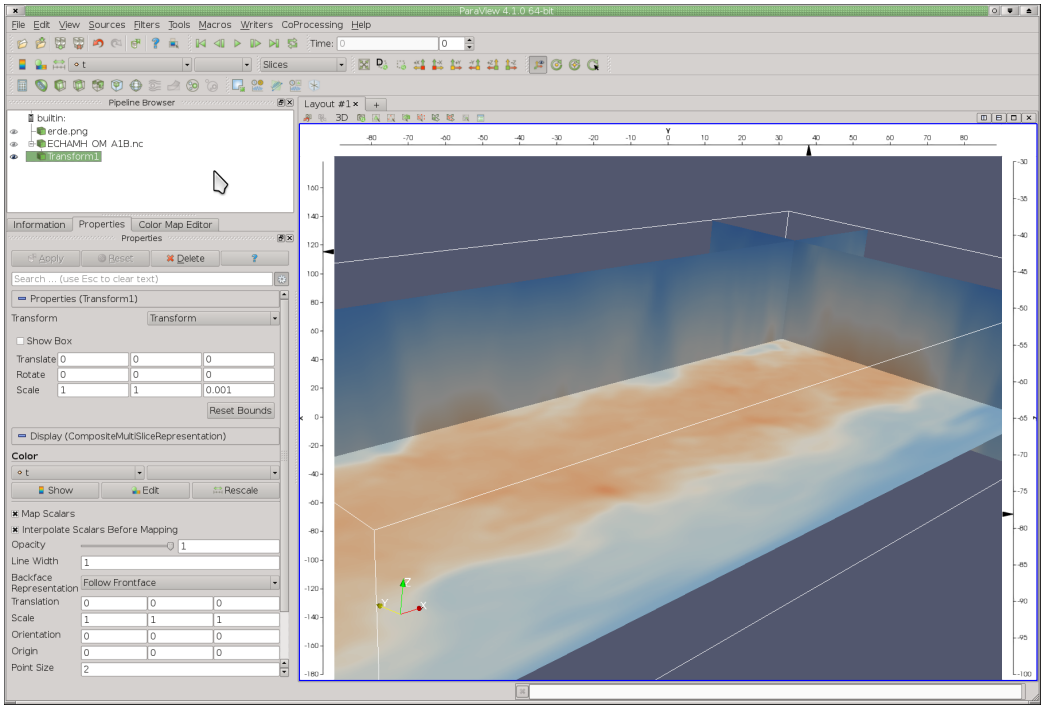

Figure 17: Slicing ECHAM data in Paraview.

So far we have only worked with the 3D viewport of Paraview, but Paraview offers many more interesting views to look at the data. One of these is the slicing view. Remove the slice module from the pipeline, and also close the 3D view at

its right upper corner. Now you have a selection of different views to open. Click on Slice View, and a 3D view with three slicing planes and axes along the side will open, as can be seen in Figure 17. You might have to switch on the Transform module (click on the grayed out eye left of it) and select a quantity for display. Now you can use the handles in the axes to move the slices around and can also rotate, pan and zoom the visualization. For you convenience, we have saved our session as Paraview state *example_2c.pvsm*.

2.1.2 *Clipping*

Another useful technique is Clipping. This method allows to clip away certain portions from the data. Please reset Paraview and continue with loading the example state *example_2a.pvsm*. Change the representation of the data back to surface rendering.

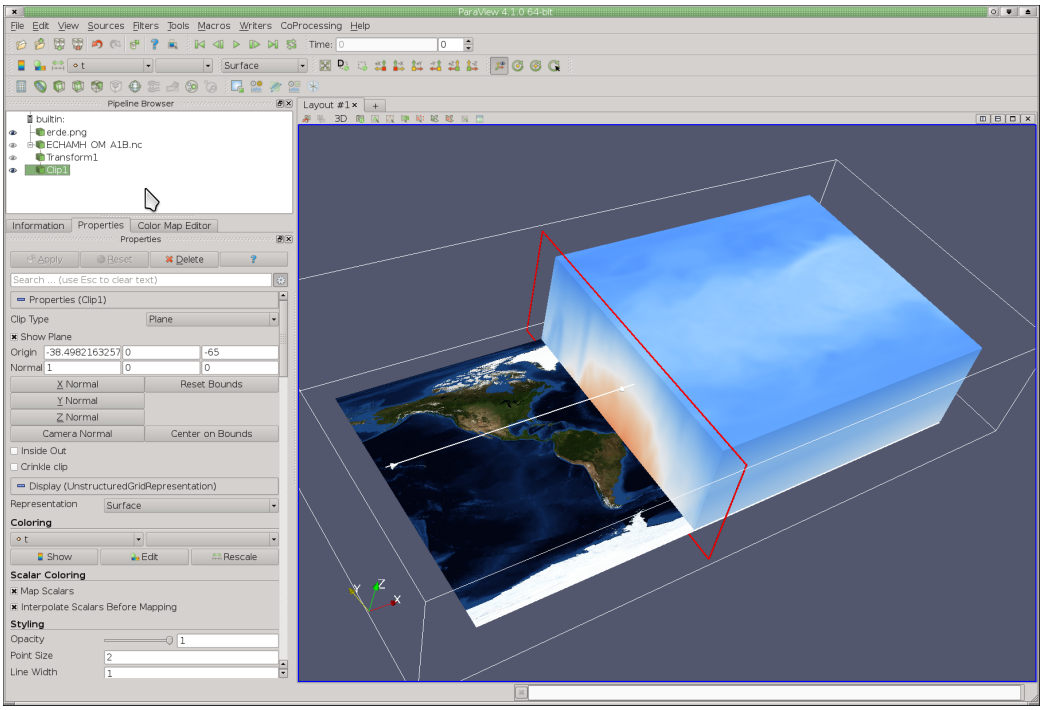

Figure 18: Cutting ECHAM data in Paraview.

Now add a Clip module, refer to Figure 15 and click on the button **Apply**. The parameter specification and interaction is very similar to slicing, except that this time, the cutting plane is used to clip away portions of the data. The resulting visualization should look similar to Figure 18.

If you select the checkbox **InsideOut**, then the cutting plane is inverted, and the other half is displayed vice verse. If you select the option *Crincle Clip*, then the slice is not interpolated, instead, it shows the real voxels. Slicing and clipping can also be combined as often as required to create complex visualizations. Try to

play around with both modules: cut away several parts of the data, and also use the slice module to interact with the remaining portions.

Compare your results with the included example Paraview state *example_2d.pvsm*.

2.1.3 *Extract Subset*

Creating a subset of a variable is similar than clipping. It allows to create one or more subvolumes for a separate processing. Instead of using up to six clipping planes, one Extract Subset module is sufficient. Furthermore, it allows to change the sampling distance for the new data set. Figure 19 shows the two subsets. The one in front has the original resolution, while the sampling along the x-axes from the one in the back was decreased by a factor of 10. Please play around with this module a bit. You can find the Paraview example state in file *example_2e.pvsm*.

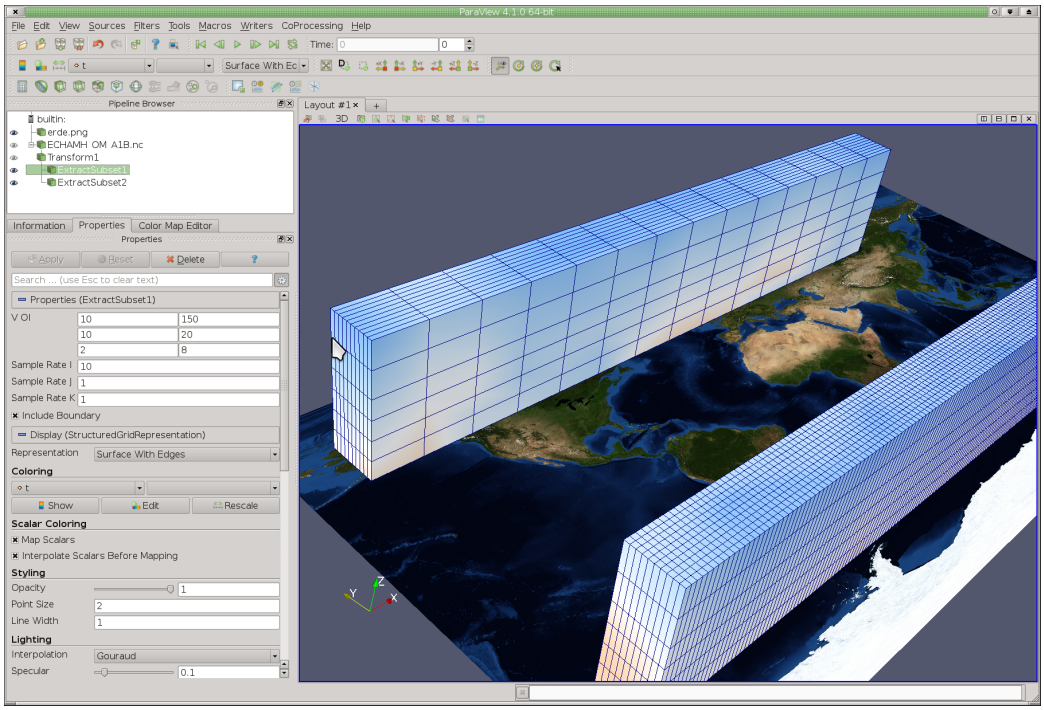

Figure 19: Extracting Subsets in Paraview.

2.2 DATA CONTOURING

Data contouring is a great technique to find structures in the data and to highlight those structures. Contours can be created from 2D slices using lines, or from 3D volumes with 3D iso-surfaces. The most important step while contouring the data is to find and select appropriate thresholds and contour values. Fortunately,

Paraview allows to create several contours / iso-surfaces simultaneously by specifying ranges.

2.2.1 *2D Contours*

First we will look how to create contours from 2D data sets. Therefore, please reset Paraview and load the Paraview state file *example_2a.pvsm*. As we have 3D data, we first create a slice around sea surface height. Please also use the variable temperature and add a contour module just below the slice. Compare your visualization pipeline with the one depicted in Figure 20. In this contour module, check if for contouring the correct variable (t) is selected and scroll down to the parameter settings. Click here on **Add Range**, change the lower value to 280 and leave the other values as they are. Click on ok and on the button **Apply**. Now we used the slice to apply a 2D line contouring algorithm for the variable temperature with 10 distinct iso-contours, see also Figure 20. Compare your results with the included example Paraview state file *example_2f.pvsm*. In this visualization we have also changed the line width to 2 for a better readability.

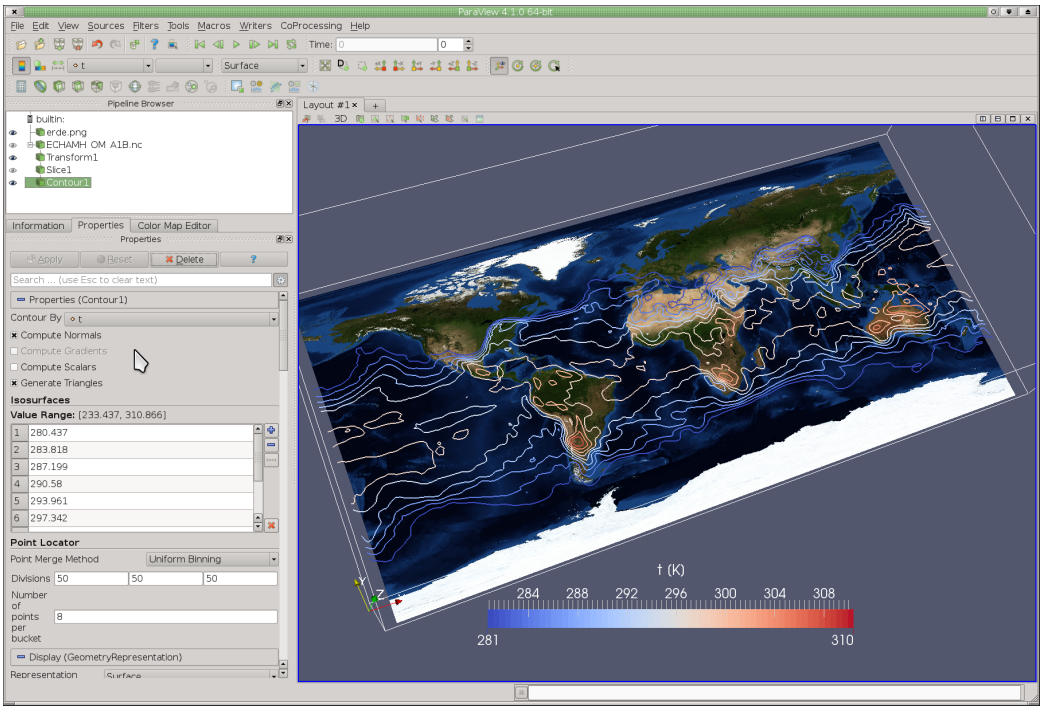

Figure 20: 2D Contours along a Slice.

2.2.2 *3D Iso Surfaces*

Where a 2D contour is represented through several lines, a 3D contour is represented by one or several iso-surfaces. All points that lie on an iso-surface have

the same value, i.e. the iso-surface is constructed by laying a surface through all points that have the same value. This value is defined as threshold, which separates the data set by using a 3D contour.

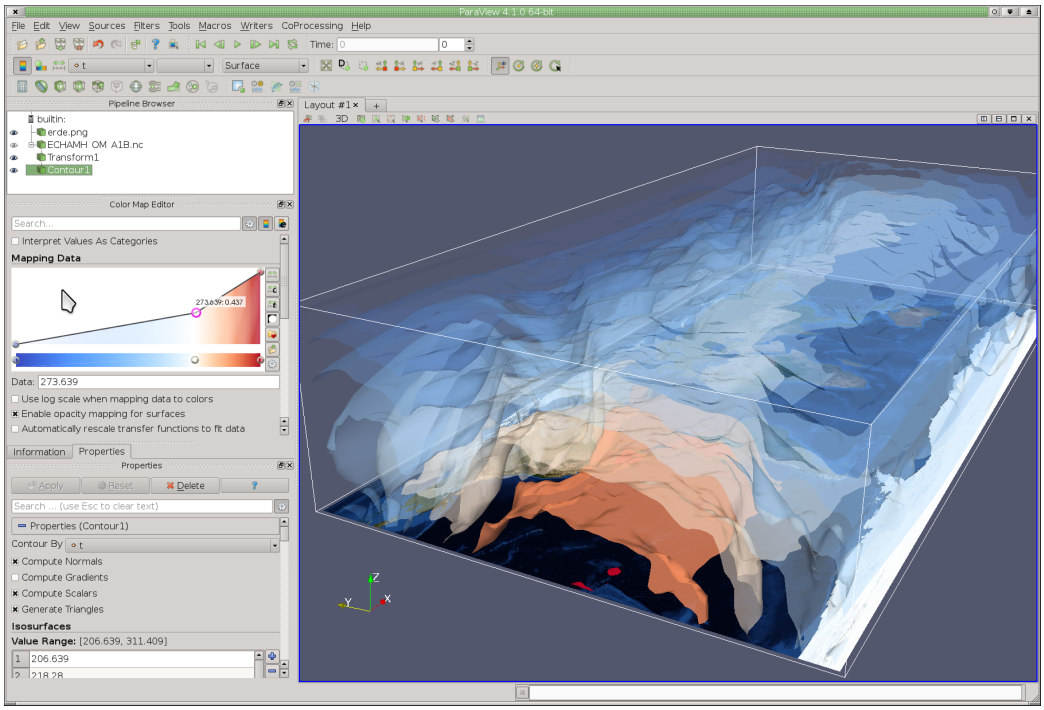

Figure 21: 3D Contouring using 10 Iso-Surfaces.

If you continue from the example above, please delete the contour and the slice module and add a new contour module, refer also to the visualization pipeline depicted in Figure 21, and click on apply. This uses a default threshold to extract a 3D iso-surface of the mean value for the currently selected variable. This threshold can be changed and set to a new threshold, but it is also possible to generate a larger set of iso-surfaces that range through the entire data spectrum. Similar to the example before, except this is now in 3D. To do so, make sure that within the module the right variable, i.e. temperature (t) is selected. Then click on **Add Range**, accept the default values, and a set of 10 iso-surfaces is created. Make sure to enable the checkbox *Compute Scalars*, as this allows you to color the iso-surfaces using the same variable that is used for contouring. Alternatively, you can also color the surfaces by using a second variable, for instance humidity. In order to see through the various iso-surfaces, we need to make them transparent. Therefore, please change the transparency of your color table similar to Figure 21, and make sure that within the to color map editor also the checkbox *Enable Opacity Mapping for Surfaces* is activated.

Compare your own results with the included example Paraview state file *example_2g.pvsm*.

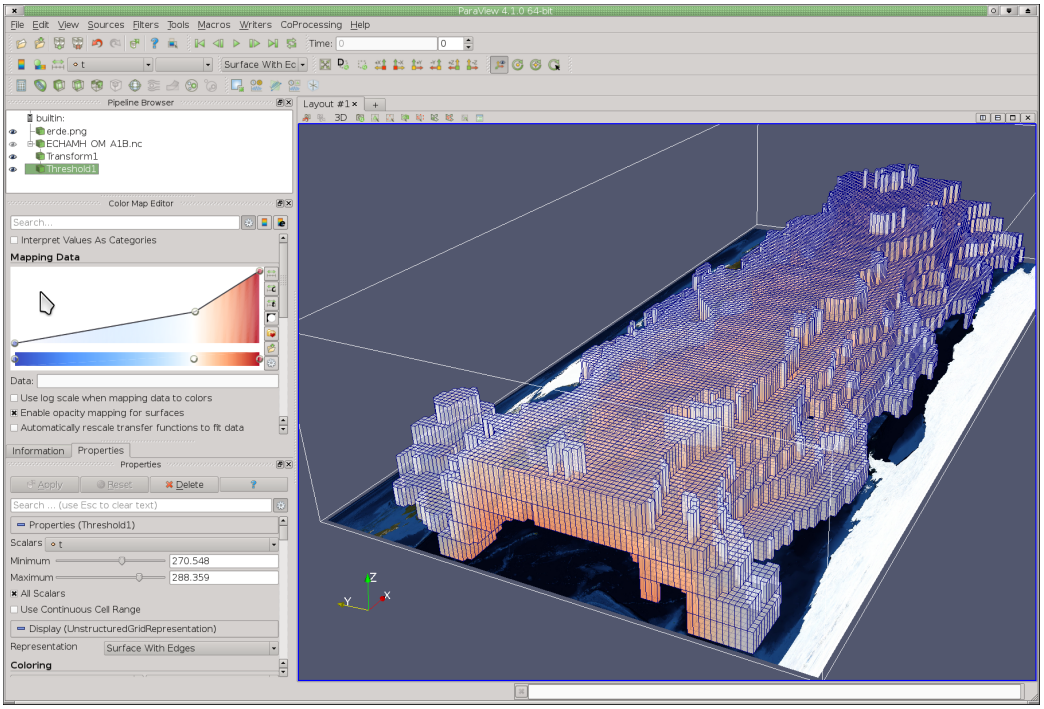

Figure 22: Using Thresholds for Temperature.

2.2.3 *Threshold*

Thresholds define lower and upper boundaries. Sometimes one is only interested in a specific data range, or in data points that lie below or above a certain threshold. For these cases, the *Threshold* module is the appropriate one. If you continue from the previous example, please delete the contour module, or reset Paraview and load the example state file *example_2a.pvsm*.

Now add a *Threshold* module, and in here chose the variable temperature (t) for scalars. Now for the lower value chose something around 270 and for the upper threshold a value of around 290. Click on the button **Apply** and chose for the rendering either *Surface with Edges* or *Volume*. If the color table does not render the surface with transparency, please enable it. Play a little bit around with the various settings, and compare your own results with the included example Paraview state file *example_2h.pvsm*.

2.3 VOLUME RENDERING

Volume rendering is a classic scalar visualization technique that visualizes an entire 3D data set, by rendering all data points with a transfer function based not only on color, but also on opacity. You have seen this technique already at the end of the last Chapter. A requirement for volume rendering though, is that the data is explicitly loaded as structured or as unstructured data set. So far, the netCDF

importer did this automatically for us, but in cases when it does not work, try to load your data as structured or unstructured data set.

To get started, simply continue from the last example, or load the Paraview state file *example_2h.pvsm*. Leave the Treshold module as it is, and simply select as representation **Volume**. If the transfer function is not suitable, click on **Edit**, and chose a better transfer function, and also adjust the opacity for the various values. Play around a little with different settings and different colortables.

Depending on the underlying data representation, Paraview offers different algorithms for volume rendering, and also supports an LoD based volume rendering that is used when interacting with the data set. The default algorithm is based on projected tetrahedra, if you need high quality drawings, you may want to switch to raycast or z sweep, although the rendering takes now much longer. If you happen to have data sampled on an equidistant rectilinear grid, then you can also switch to a GPU based volume rendering, which is not only very fast, but also produces a very good image quality. Compare your own results with the included example Paraview state file *example_2i.pvsm*.

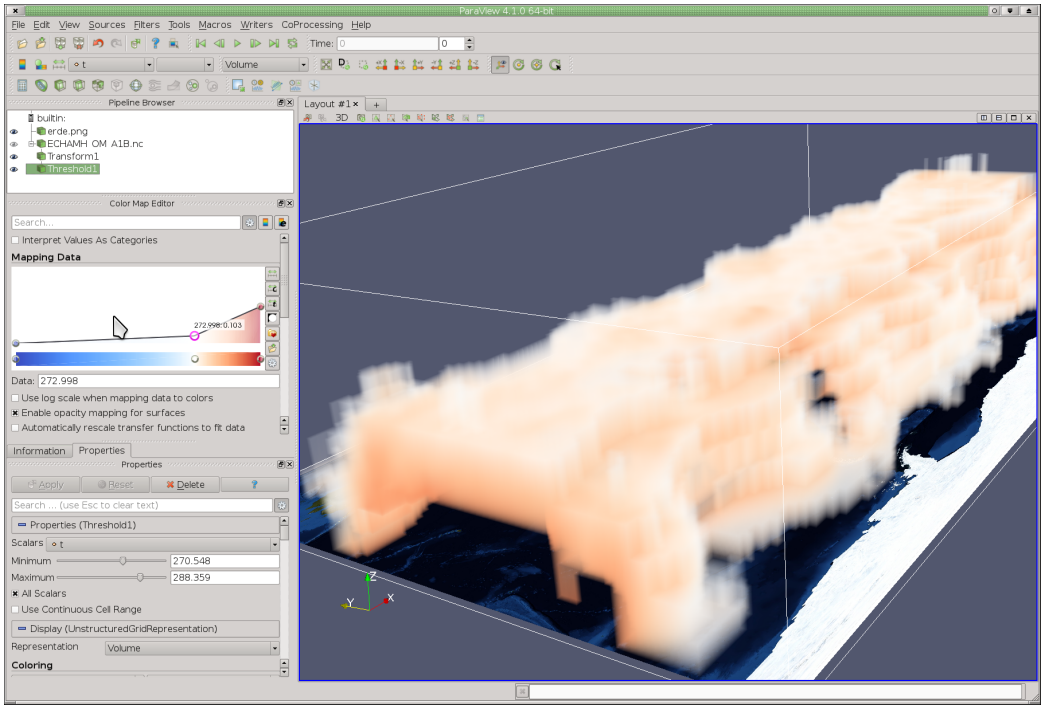

Figure 23: Volume Rendering in Paraview.

For the second volume rendering example, we would like to render some *clouds*. Therefore, delete the threshold module, and chose as variable relative humidity and as representation Volume. Please also create a color table similar to the one in Figure 23, which uses blue and white colors and high levels of transparency. Play a bit around with the scene and the color table, until you found a good setting and good point of view. Now, to see the visualization in even better

image quality, switch the mapper for volume rendering to *Bunky Raycast*. The rendering takes a little longer. For your reference, please find the current session also in the Paraview state file *example_2k.pvsm*.

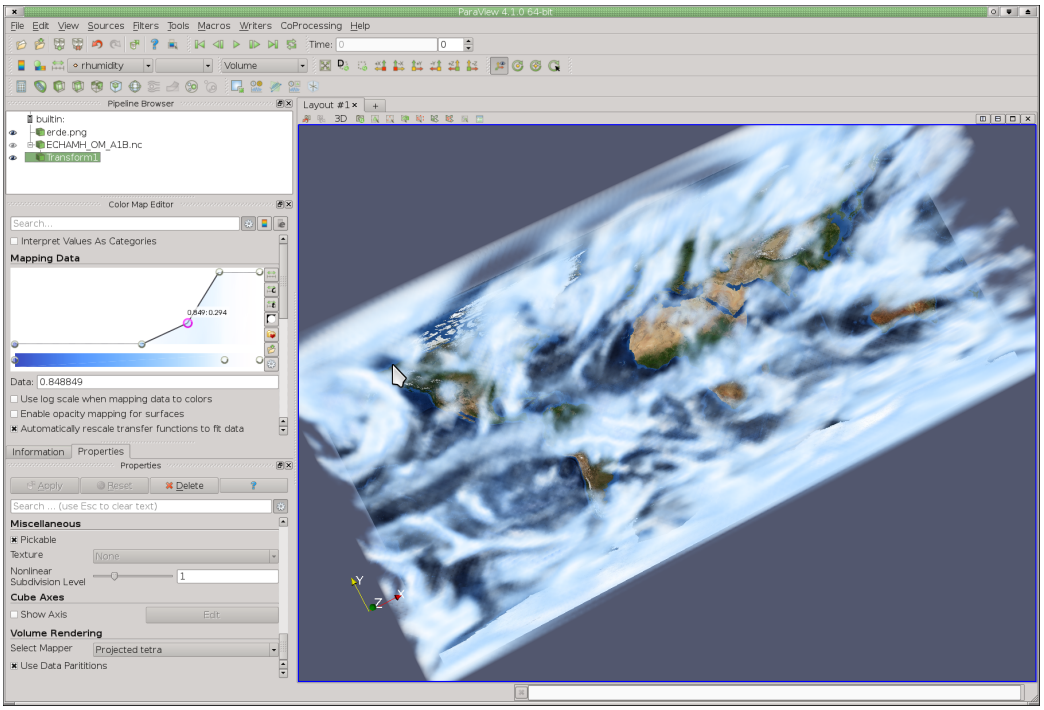

Figure 24: Volume Rendering of relative Humidity.

2.4 UNCERTAINTY RENDERING

As open source software with the VTK as visualization pipeline, extensions and additions to Paraview in the form of plugins are easy to create. Therefore, several additional plugins are available for Paraview, to support additional data formats, as well es for data processing and rendering. Two interesting plugins for rendering uncertain scalar (and also vector) data are the Point Sprite Rendering Plugin and the Uncertainty Surface. In order to use these plugins, there must be loaded into Paraview manually. To add a plugin to Paraview, browse to **Tools->ManagePlugins**, and load in the upcoming dialog the **Point Sprite Plugin**, as well as the **Uncertainty Rendering Plugin**. To make sure that both plugins are loaded automatically from now, please also check the box that is labeled *Autoload*.

Although simulations appear to have exact values and correct data, all simulations are inherently afflicted with some degree of uncertainty. Sometimes it is possible to specify the level of uncertainty, sometimes not. The two plugins that are discussed in the following are for the first case. Data with a specific uncertainty is for instance created through ensemble runs. Here one has the mean value, as well as the statistics to describe the deviations in the individual simu-

lation runs. In the following examples, we use data from ... to demonstrate the applicability of both plugins.

2.4.1 *Uncertainty Surface*

The following example will be a bit more complex, and uses some techniques and procedures that we have learned thus far. To start, please reset Paraview, and load the sate file *earth.pvsm*, which simply loads an earth texture for annotation into the background. Now load the netCDF file *cmip5_uncert.nc* as a regular netCDF file with an equidistant cylindrical mapping into your visualization pipeline, and explore the data variables that it contains. This netCDF data is comprised of two variables, temp and skill. Temp represents the change of temperature at 2m height. Positive values thereby mean that the temperature increases, while negative values imply that the average temperature at this position sinks. The variable skill describes the certainty of this information, in which positive values represent a high certainty, while negative values remain more untrustworthy. Originally, an ensemble simulation with 10 ensemble members has been performed, from which the mean temperature value (temp), as well as the certainty of this mean value – i.e. how much the ensemble differed at this position – were derived. So far, we can visualize either one, but not a combination of both.

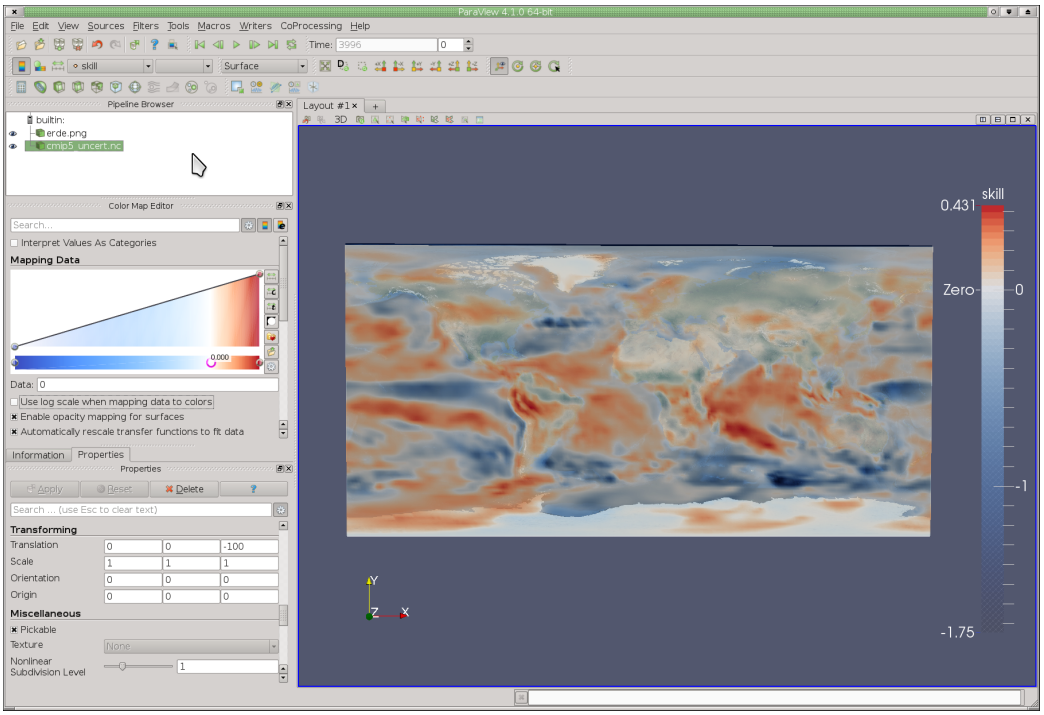

Figure 25: Uncertainty Variable in CMIP5 (Temperature at 2m).

Figure 25 shows a visualization of the skill variable. To create this visualization, we have to translate the slice closer to the textured surface, so move it

in the negative z-direction by -100, see also Figure 25. Next select the skill variable for display and a Surface representation for rendering. We agree with the cold / warm color table selected, but move the middle color to 0.0, change it to pure white, and also enable the checkbox *Enable opacity mapping for surfaces*. The opaque and red colored areas in your visualization represent data values with a high certainty, while the less reliable information is now blended out and rendered translucent. Compare your own results with the included example Paraview state file *example_2m.pvsm*. But this only visualizes the certainty, and not the actual temperature data. So how can we combine both?

One answer to this question is the *Uncertainty Surface*, which is an optional plugin for Paraview, and allows a visualization of uncertain data sets. Uncertainty Rendering needs one additional scalar value – which represents certainty – and perturbs the color mapping of the original data using Perlin noise, depending on the (un)certainty of the data. An example visualization can be seen in the lower half of Figure 26.

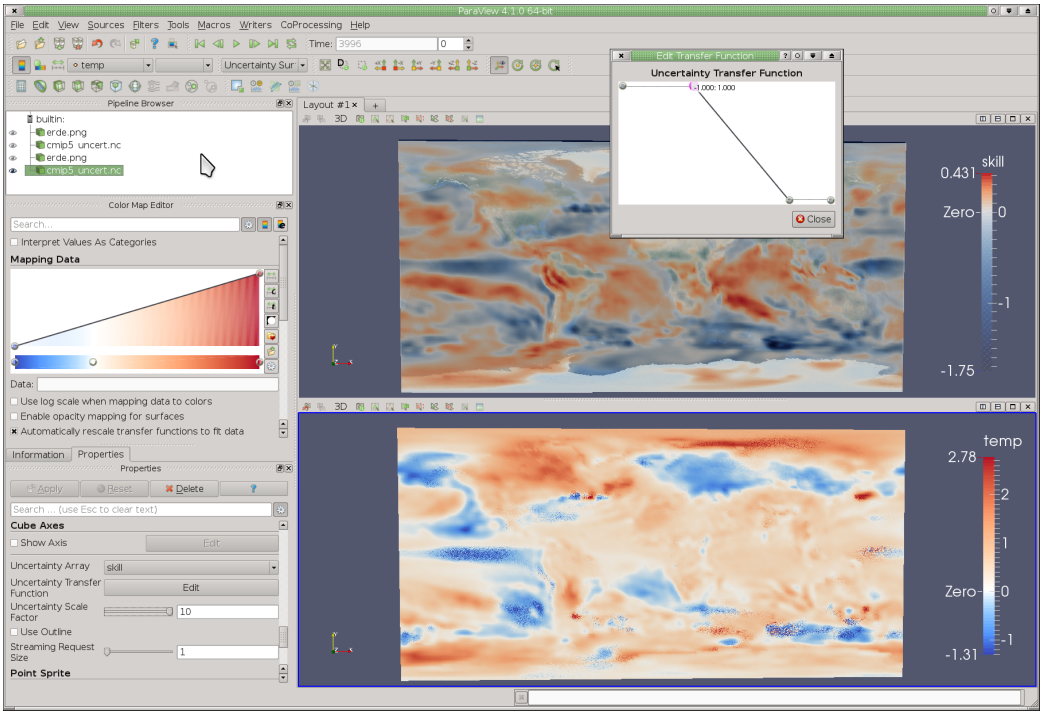

Figure 26: Uncertainty Surface Rendering in Paraview.

To create the visualization, perform a vertical split of the main viewport, and create a second 3D view. In this one, enable the netCDF data for display by clicking on the gray eye, and zoom in a little closer. We do not need the Earth texture this time. Also, select temperature (temp) as data variable, and chose *Uncertainty Surface* for representation and rendering. Adjust the colortable by moving the middle color to zero, and by changing it as well to pure white. Now we really visualize the changes in temperature at 2m height. Please also annotate

the viewport with a colortable and highlight the value Zero. Now scroll down a bit in the Properties menu of the netCDF module, until you find the place in which you can change the *Uncertainty Array*. Please change it to the variable skill, and also adjust the Uncertainty Transfer Function to something similar as can be seen in Figure 26. We want a high uncertainty for all values below -1 and a high certainty for all values above 0, and a gradual change in between. Please also move the *Uncertainty Scale* factor higher to a value about 10 to better observe what the technique actually does to your visualization.

If you compare both visualizations in Figure 26, you will notice that the areas in which we have a high certainty, are colored uniformly, as we know it from previous surface visualizations before. But areas that exhibit a higher degree of uncertainty $\leqslant -0.5$ have some color bleeding with red and blue values, depending on the level of uncertainty. This shall visualize that the certainty of the information depicted in this area can be either positive or negative, and with the data used, no clear answer can be given.

Compare your own results with the included example Paraview state file *example_2n.pvsm*.

2.4.2 *Point Sprite Rendering*

The optional *Point Sprite Rendering* plugin is also able to visualize uncertainty data, however, at the moment the plugin seems not to be fully stable and reliable. In contrast to the Uncertain Surface plugin, it visualizes each data point using a textured sphere, in which two other scalar data sets, such as uncertainty, can be used to define the size of the sphere and its opacity. For now, feel free to explore the possibilities of this module on your own.

3

VECTOR DATA VISUALIZATION

MANY scientific simulations not only contain scalar fields, but also vector data. These variables are comprised of two or more individual scalar fields, which, when combined, describe a 2D or 3D vector array. Vector fields are characterized by direction and magnitude. Generally, one would like to visualize both at the same time, and / or combine it with a second or third additional scalar field. This chapter describes some of the basic and advanced visualization techniques available within Paraview to visualize vector fields, as well as how to import vector data and how to combine regular scalar fields to form a new 2D or 3D vector array.

First, we need to start or reset Paraview, after which we load the state file *earth.pvsm*, which simply loads an Earth texture in the background to later annotate the data displayed. As the transform module, which we we have used so far to scale our data sets, is unfortunately not aware of additional representation and rendering plugins, we had to find an alternative way to scale our data. One alternative was to use CDO – the climate data operators – to write out a description of the z-axis (`cdo zaxisdes input.nc > zaxis.txt`) and store it into a text file. Now we can use a regular text editor to manually adjust the scaling of the z-axis, and use CDO again to set the new z-axis (`cdo setzaxis,zaxis.txt input.nc output.nc`).

We have already done this, and therefore, we simply load the netCDF file *EH_A1B_3Dvectors.nc* into our visualization pipeline. In the netCDF module, make sure that the *Output Type* is set to Structured and that the data is loaded as equidistant cylindrical representation, i.e. not using spherical coordinates. Please click on the **Apply** button. If you check the information tab about the variables contained in the data set, you will find the following 3D variables for (lev, lat, lon):

- q – Specific humidity,
- rhumidity – Relative humidity,
- t – Temperature,
- u – Horizontal velocity
- u – Horizontal velocity
- W – Vertical velocity

As it appears, our vector data set consists of the three scalar fields u, v and W that build up a vector. Now we use the Paraview module *Calculator* to combine these three scalar fields into one single vector field. Therefore, click on

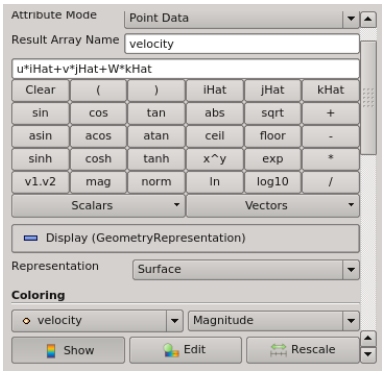

Figure 27: Constructing a 3D vector field.

the Calculator icon, and within the module add **velocity** as the *Result Array Name*, and add the following line below:

```
u*iHat+v*jHat+W*kHat
```

This expression uses the individual velocities u, v and W, and combines them with their corresponding unit vectors iHat, jHat and kHat to form the new 3D vector field **velocity**, compare also with Figure 27. Other than that, the Calculator module serves now as source for data variables, and passes on all other variables

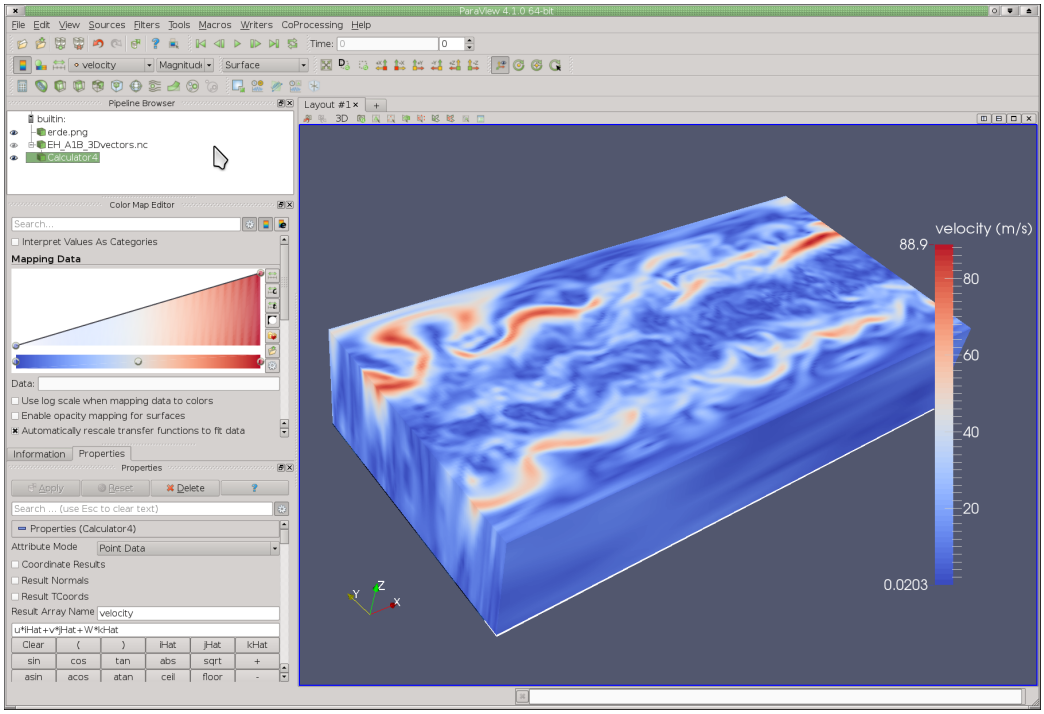

Figure 28: Visualization of the vector field Magnitude.

from the original netCDF module. This module can also be used to assign a color table for data coloration, as well as to add a color bar annotation.

After hitting the **Apply** button, the new vector is computed, and can be selected from the variables available, refer also to Figure 28. If selected, automatically the magnitude is derived and displayed, see Figure 28, but also the individual vector components X, Y and Z can be used for display. The following sections in this chapter will have a closer look at dedicated vector visualization techniques. Compare your own results with the included example Paraview state *example_3a.pvsm*.

3.1 SURFACE LIC

LIC, or line integral convolution, is a well know vector visualization method that is able to visualize the vector field, or a slice, in its entirety. Although glyphs provide a good impression about the topology of a vector field, the real structure becomes much more apparent by using the LIC technique. The LIC rendering technique is implemented as an optional plugin, which has to be loaded by Paraview during startup. Therefore, please make sure that the **SurfaceLIC** plugin is loaded, and check under **Tools->ManagePlugins** if the SurfaceLIC Plugin is present and loaded. If not, please load it, and also place a checkmark at the autoload option.

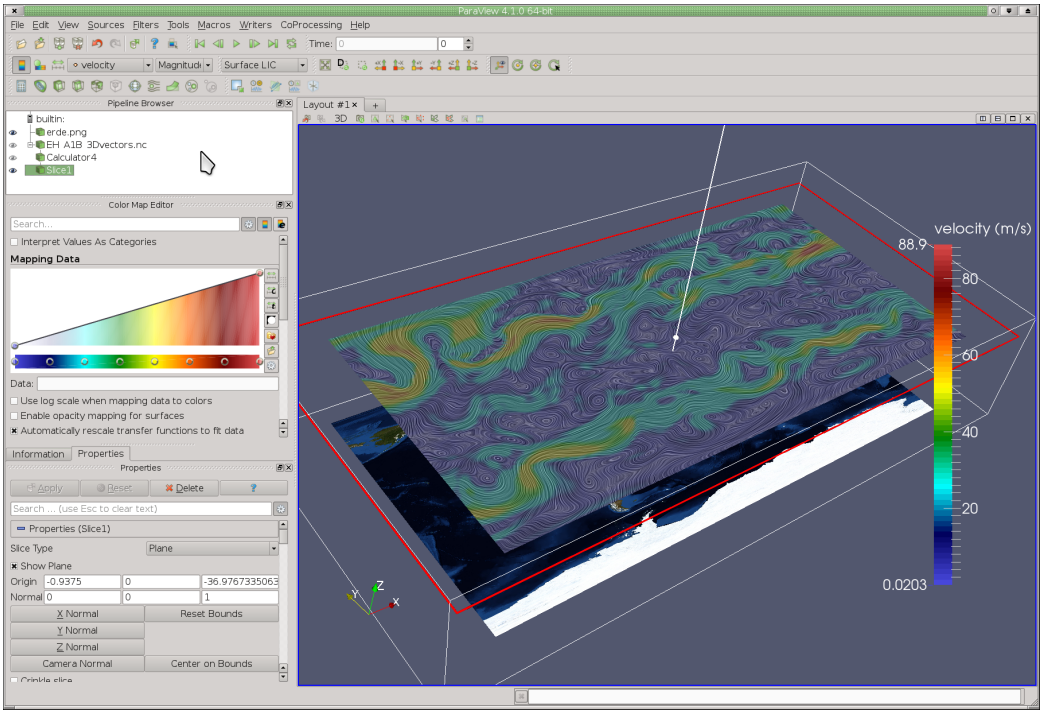

Figure 29: Vector Visualization with Surface LIC.

Just continue from the previous example, or start by loading the Paraview state *example_3a.pvsm*. Now we are adding a slice module, which will be used to render our LIC texture. Therefore, add a **Slice** module to the visualization pipeline, select the z-axis for slicing, and select a plane in the upper atmosphere, as in here we can expect the highest velocities. Now press the **Apply** button, and choose **SurfaceLIC** as representation instead of just Surface. A visualization, such as depicted in Figure 29, appears. To make the higher velocities more apparent, we chose the colortable *Rainbow Desaturated* as main transfer function. Please slice through the data to explore the vector field, and also try out other colortables if you like.

Compare your own results with the included example Paraview state *example_3b.pvsm*.

3.2 GLYPHS

One of the easiest and most expressive ways for the visualization of vector data is to use glyphs. A glyph is a geometric object with a specific size, a direction and a color, which is drawn at certain positions within the vector field. The geometry of these glyphs is usually aligned with the vector field to describe the direction of the flow, while its size and color are usually mapped to other quantities, such as the vector magnitude. Figure 30 shows a visualizations using arrow glyphs,

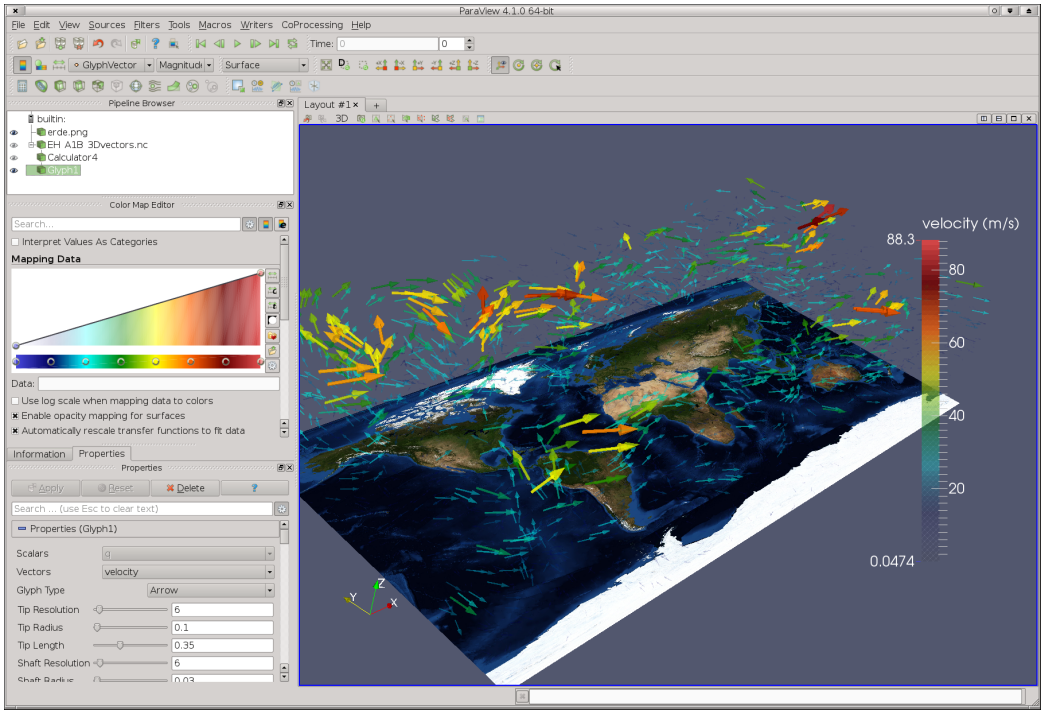

Figure 30: Vector Visualization with Glyphs.

which characterizes the main direction and magnitude of the vector field quite well.

To recreate this visualization, either continue from the previous example, or load the Paraview state file *example_3b.pvsm*. Now delete the **Slice** module, as we would like to continue with the entire 3D data set. You can, however, also apply this technique just onto a 2D slice. In that case, keep the slice, and simply continue with the tutorial.

Now click on the icon **Glyph** in the main toolbar and add it to your visualization pipeline. In the properties tab, select **Arrow** as glyph type and hit apply. Other glyph types that can be used are cone, line, box, sphere and a 2D glyph. Play around with different glyphs if you like, and see how the individual parameters change the appearance of the glyphs displayed. In the end, select **Arrow** as glyph type, and use the **Glyph Vector** for coloration. This is simply the magnitude of our vector field. Please also choose a suitable color table, such as *Rainbow Desaturated*, and check the box labeled *Enable Opacity Mapping for Surfaces*. That way, we use transparency in our visualization, and the lowest velocities are simply blended out. Compare your own results with Figure 30 and the included example Paraview state *example_3c.pvsm*. The strongest velocities appear in the upper atmosphere, and can be easily identified as the well known jet streams.

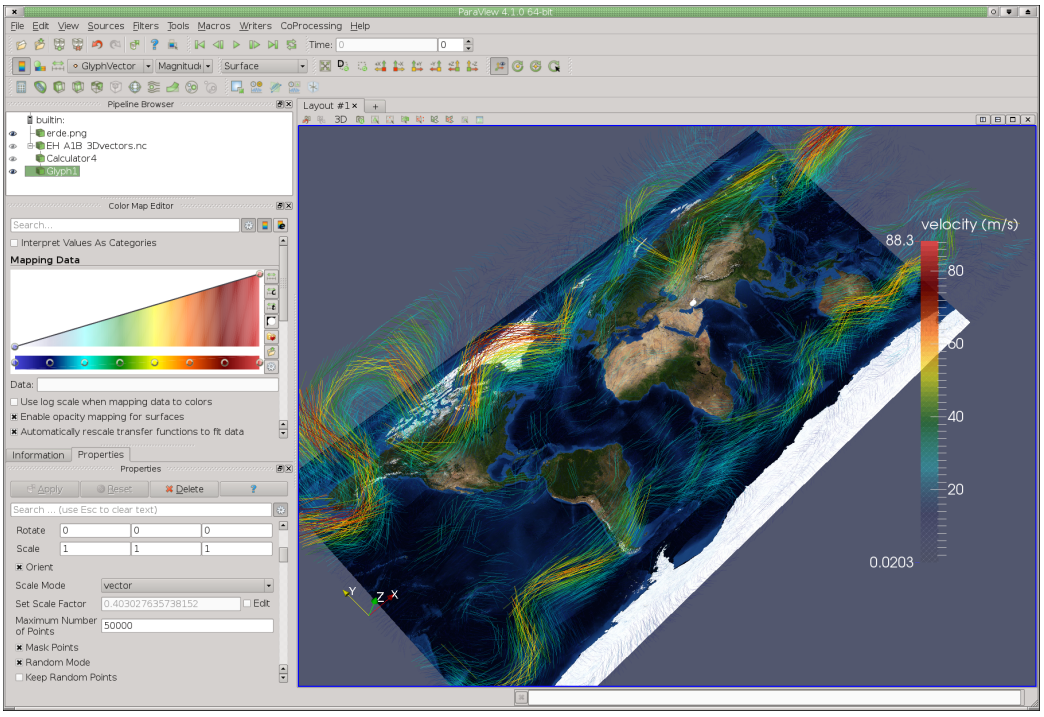

Figure 31: Hedgehog Vector Visualization with Line Glyphs.

3.2.1 *Hedgehogs*

The glyph module is in itself very powerful due to its large variety and many settings. Not only allows it various shapes to represent scalar and vector quantities, also 3rd variables can be used for scaling the glyph. The next example includes a very dense vector field representation using many line segments. Therefore just continue from the previous example, or simply load the Paraview state *example_3c.pvsm*.

Now we switch in the glyph module from arrow to a line representation, and as we want many lines to be drawn, we scroll down in the properties section of the glyph module and change the **Number of Points** from 5.000 to 50.000. We keep the colortable, and if we hit apply, the lines are computed and displayed, compare also with Figure 31.

Just a little above the setting for the maximum number of points, we can switch the scaling from vector to scalar, and can choose a scalar variable to influence the length of the line segments. Currently, they are drawn according to the vector magnitude, and are longest in the higher atmosphere. If we switch to humidity or temperature, the visualization changes, as here the higher values are closer to the Earth surface. That way, we can create very complex, yet also difficult to read and interpret visualizations. Whenever creating a visualization that is not just for yourself, think about the target audience, and include as much information as necessary, for instance captions and color bar descriptions.

Compare your own results with the included example Paraview state *example_3d.pvsm*.

3.3 WARP BY VECTOR

Sometimes one would like to distort the visualization of one quantity by using another one. The module **Warp by Vector** does just that, it warps a slice, or a complete 3D surface, depending on the underlying vector field. To do this, either continue from the previous example, or start fresh by loading the Paraview state file *example_3d.pvsm*.

Now, we need a new slice module, so we just add one and create a slice in the upper z-direction, as we have done many times before. To continue, we also add a **Warp by Vector** module from the menu, and press the **Apply** button. You see that the slice is somewhat distorted, but it is now difficult to see by how mach. Therefore, for the slice, decrease the **Scale Factor** to 0.25, switch the rendering to **Surface with Edges**, and color the edges with a much lighter color than dark blue, compare with Figure 32. You can change the color for the wireframe in the properties section of the Warp by Vector module. Just scroll downwards, until you reach the section **Edge Styling**. Here you can adjust the edges color.

The warp by vector module might be useful if one wishes to visualize the extent of a directional error, for instance, or simply to visualize uncertainty. The more distorted the mesh is, the higher the uncertainty. The mesh is distorted

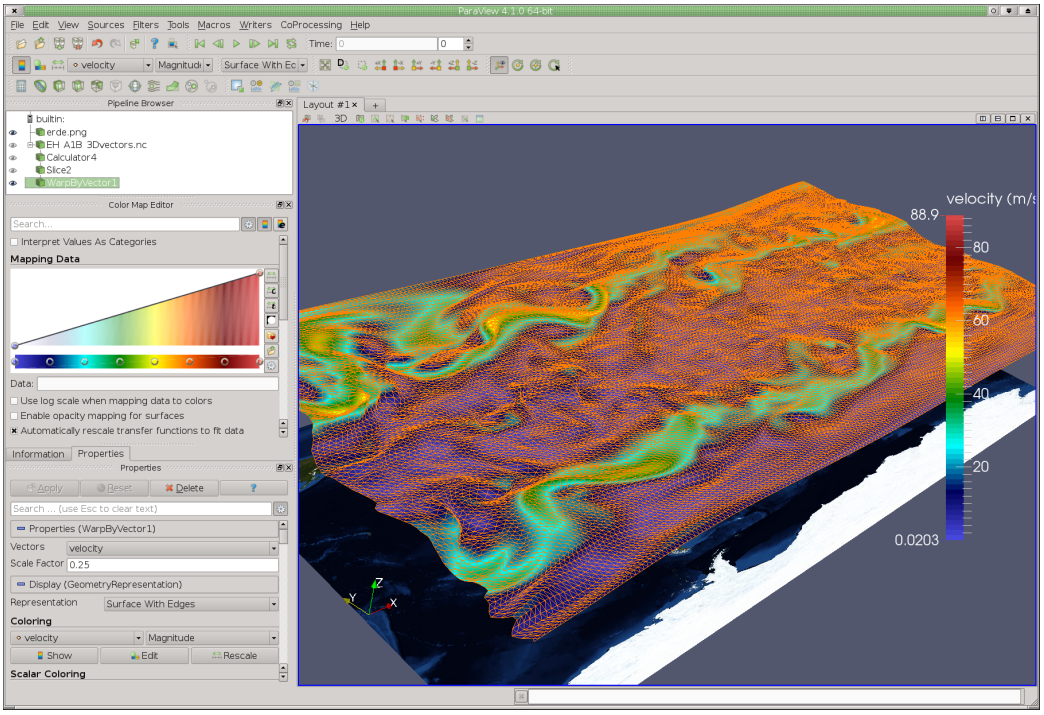

Figure 32: Vector Visualization with Warp by Vector.

according to the vectors (errors) length and direction. Compare your own results with the included example Paraview state *example_3e.pvsm*.

3.4 STREAMLINES

Another powerful and often employed method for vector and flow visualization are streamlines. Streamlines are visualizations of tracked massless particles placed in the vector field for one single time step. Streamlines are not pathlines or trace particles through the velocity field over time, instead, streamlines are computed over one single time step. They describe the flow field for any given point in time, and are very useful to visualize what is flowing from where and in which direction. Streamlines are also suitable for finding vortices, and during the streamline computation in Paraview, a new vorticity variable is derived and can be used in the visualization

To continue with the example, either continue from the last session, or load the Paraview example state file *example_3e.pvsm*. As we no longer require the modules *Warp by Vector* and *Slice*, please delete both of them, and add a *Stream Tracer* module from the main toolbar, compare with the visualization pipeline depicted in Figure 33. The Stream tracer module offers quite many different settings, however, we will only discuss a couple of them. Leave the integration and streamline parameters as they are, and scroll down to the *Seed* section. As seed type we select point source, and we increase the number of points to 1000

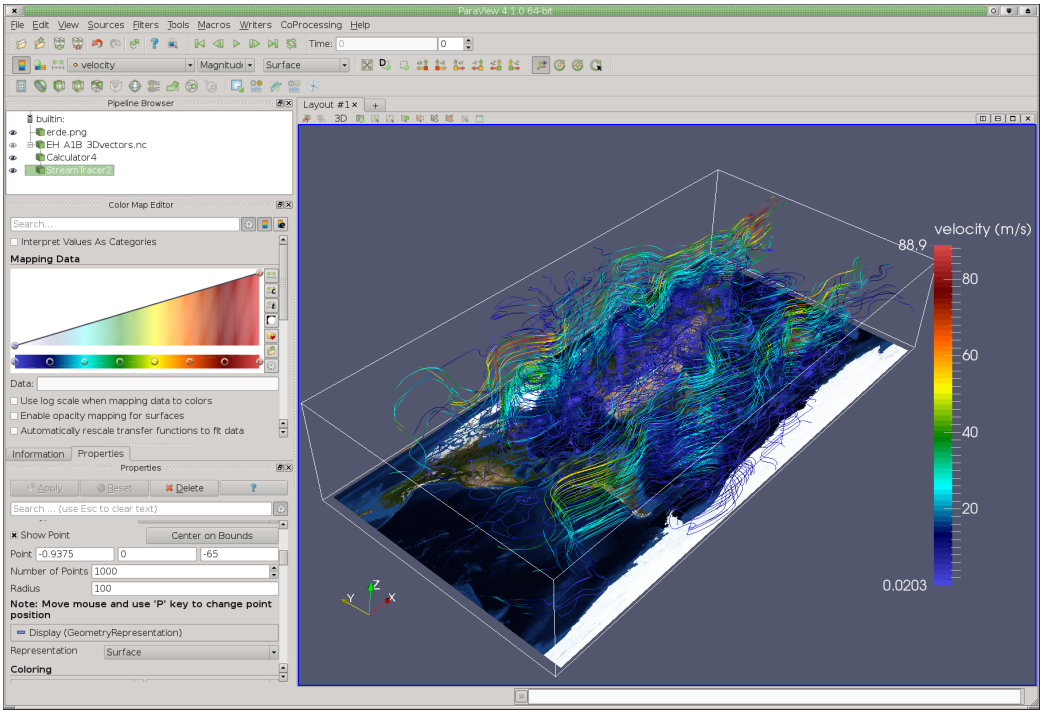

Figure 33: Vector Visualization with Streamlines.

and also the radius to 100. After this, press the **Apply** button and wait a while, while the streamlines are computed. Your visualization should now look similar to Figure 33. Compare your own results with the included example Paraview state *example_3f.pvsm*.

Please play a bit with the different settings, and also explore the additionally derived variables like vorticity and angular velocity. In order to compute a very dense field of streamlines, change the number of points to 2000, and also increase the radius to 150.

3.5 BUMP SLICE

The last example is an extension to the regular slice.

4

INTERACTIVE DATA ANALYSIS TECHNIQUES

SIMULATIONS are not only performed to proof an existing hypotheses, but often also to learn and to understand unfamiliar processes and to find unknown features and structures. In these cases, one only has a vague idea of what information the data set contains, and how the single variables are related and possibly dependent on each other. Besides the design of intuitive to interpret and good looking animations for presentation and communicating purposes, visualization is especially well suited for an interactive data mining and data analysis. Several specially developed techniques are available to assist the user to interactively explore the variables contained and to derive more specific information about the nature of the data. One area that is especially concerned with the development of such techniques is called *Information Visualization*. This chapter presents and discusses several techniques, and shows how they can be applied to analyzing climate science data sets.

The majority of the techniques discussed in this Chapter deal with point, but not with cell data. Therefore, sometimes cell data has to be converted into point data, which is relatively easy to do in Paraview, and only requires the use of one additional filter (CellDataToPointData). For the first example in this Chapter this is not necessary, but you should keep this in mind if you would like to apply some of these techniques to your own data sets.

4.1 HISTOGRAM AND BASIC STATISTICAL INFORMATION

Please start Paraview and load from the tutorial folder the state file **earth.pvsm**. This state only contains a correctly scaled and positioned earth texture, which we will use to annotate the visualization. After this, load the data set **Aeropt_2010.nc** from the tutorial folder. This data set contains five 2D variables that show the concentration of various aerosols and particles in the Earth atmosphere:

- BC – Biological Carbon
- DU – Dust
- OC – Organic Carbon
- SS – Sea Salt
- WASO – Sulfur and Nitrate

If you load the data set, please make sure that you select the correct netCDF module (netCDF files generic and CD conventions), and that you deselect **Spherical Coordinates** and select **Replace Fill Values with NANs**. We also have to

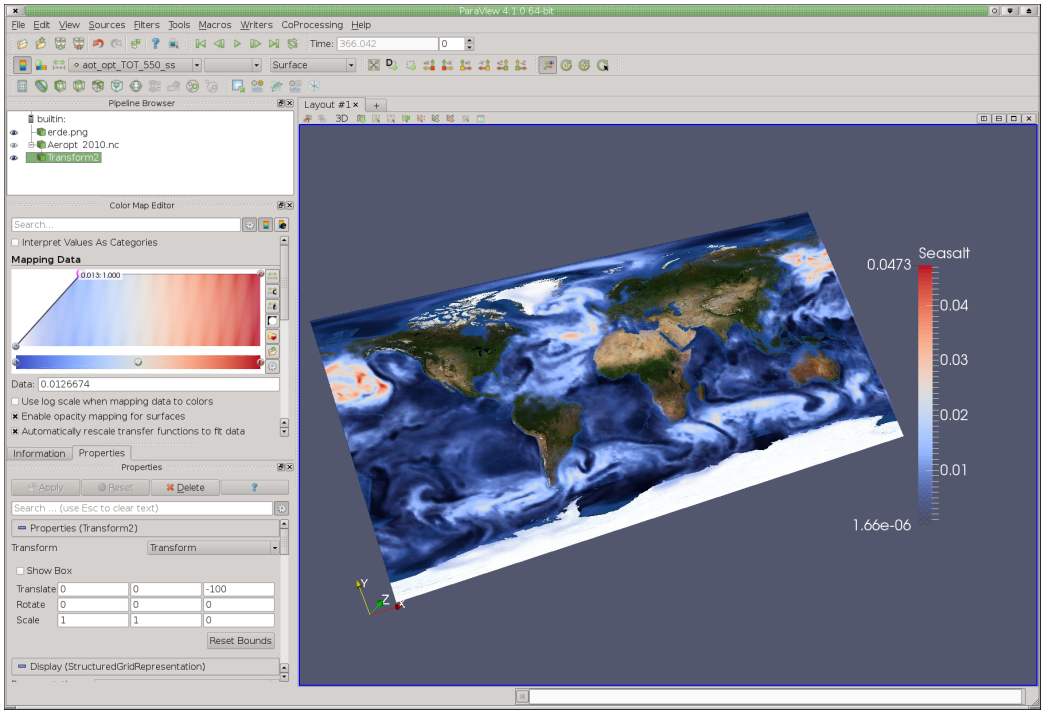

Figure 34: Paraview with Aerosols Data.

translate our data a bit, therefore, as you know from previous sessions, please apply a *Transform* filter and translate the data in the z-direction by -100. Now, if you enable any variable and draw it as surface, the earth texture will be hidden, and as annotation not be useful anymore. Therefore, we have to use transparency. To do yo, go to the color map editor, and activate the checkbox *Enable Opacity Mapping for Surfaces*.

If you have chosen the variable *aot_opt_TOT_550_SS* – which shows concentration of sea salt particles in the atmosphere – your visualization should look similar to Figure 34. To plot a Histogram for one or for all data variables, one only has to select a certain filter from the **Filter** Menu. To do so, please choose **Filter->Data Analysis->Histogram**, and a new histogram filter appears in the visualization pipeline, refer also to Figure 35.

The new module automatically splits the view in half, and opens in the right a new Bar Chart View, which displays the Histogram. Please make sure that you choose the correct *Input Array* for the histogram binning. After clicking on **Apply** the histogram is computed and displayed as is shown in Figure 35. As the concentration follows an inverse exponential rate, the histogram is zoomed in. You can also increase the number of bins for a more precise display and also chose other variables for plotting.

Compare your own visualization with the included example Paraview state *example_4a.pvsm*.

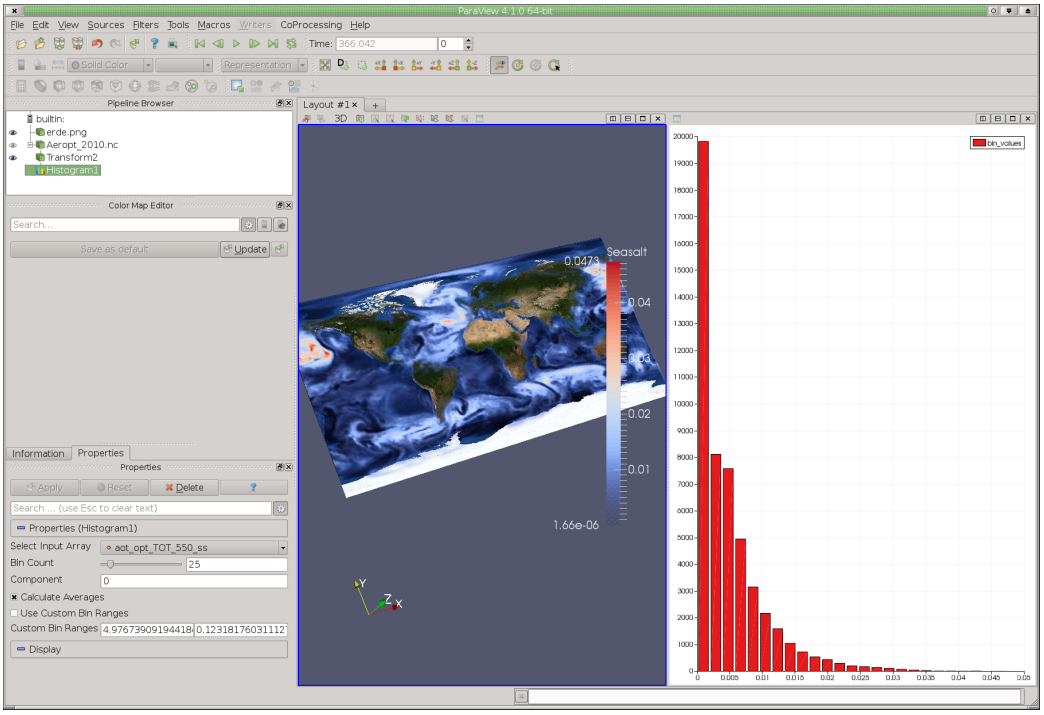

Figure 35: Plotting a Histogram for Sea Salt Concentration.

4.2 SPREADSHEET VIEW

Another very useful view is the spreadsheet view, as can be seen in Figure 36. The spreadsheet view shows for all data points the point, cell and field data variables. These can also be sorted and selected, and if there is something wrong within the data, either missing, too large or too small values, they will show up here. Therefore, the spreadsheet view is an ideal tool to find abnormal values that are either artifacts, or result from a bad simulation model.

To open the spreadsheet view, close the bar chart view and delete the Histogram module. Now create a second view by clicking on the vertical split icon, and select a spreadsheet view (at the very bottom) as new view. Paraview should now look similar to Figure 36. In this spreadsheet view, from the top left, one can select the data source, the data type (i.e. point, cell or field data), and the precision, with which the data shall be displayed. The other three icons allow to select data values, to show only selected data values (very useful), as well as to enable and disable certain variables. To sort a data array, simply click on the name, and it will be sorted either ascending or descending for this variable. For now, find the variable sea salt and order it from high values to low values. Now select the highest 5 to 10 percent, and as you select the variables, you immediately see the variables also becoming selected in the other views. That means that these two views are *linked* together, which is of great assistance for the data exploration.

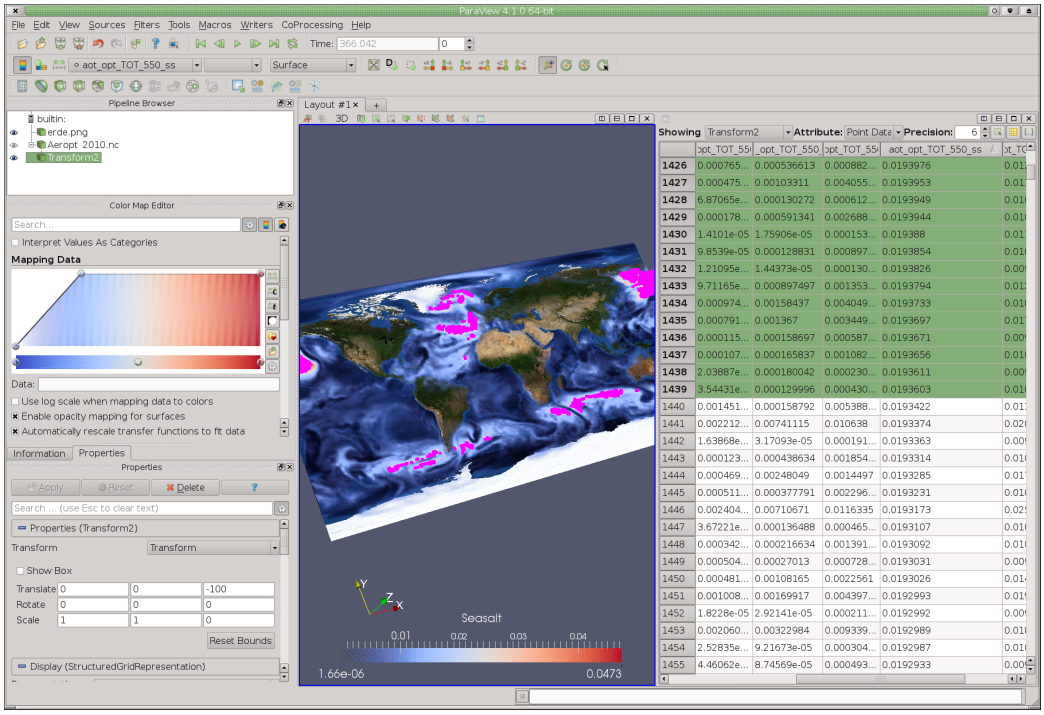

Figure 36: Paraview's Spreadsheet View.

Compare your own visualization with the included example Paraview state *example_4b.pvsm*.

If you are interested in the entire amount of all data values of a variable accumulated together, you can use the filter **Filter->Data Analysis->IntegrateVariables** to perform this operation. A spreadsheet shows you the accumulated data values after applying this filter.

4.3 TEMPORAL STATISTICS

The majority of climate simulations has more than one time step, and sometimes, it is desirable to easily find the maximum, the minimum, or to derive some statistics, like the average or the standard deviation. In Paraview, all this can be performed with one module called *Temporal Statistics*. Please close the spreadsheet view. Maybe you remember that in previous Chapters we not able use the Transform filter together with additional representation types, such as the Uncertainty Surface plugin. To circumvent this problem, we disable the earth.png, as well as the Transform filter, and we select the netCDF data module (which is in the visualization pipeline just before the Transform filter). Now browse in the Menu to **Filters -> Temporal -> Temporal Statistics**, and add this module to your visualization pipeline. If you click on the button **Apply** you will find 4 new variables for each existing variable. These represent the minimum, maximum, average and standard deviation per point.

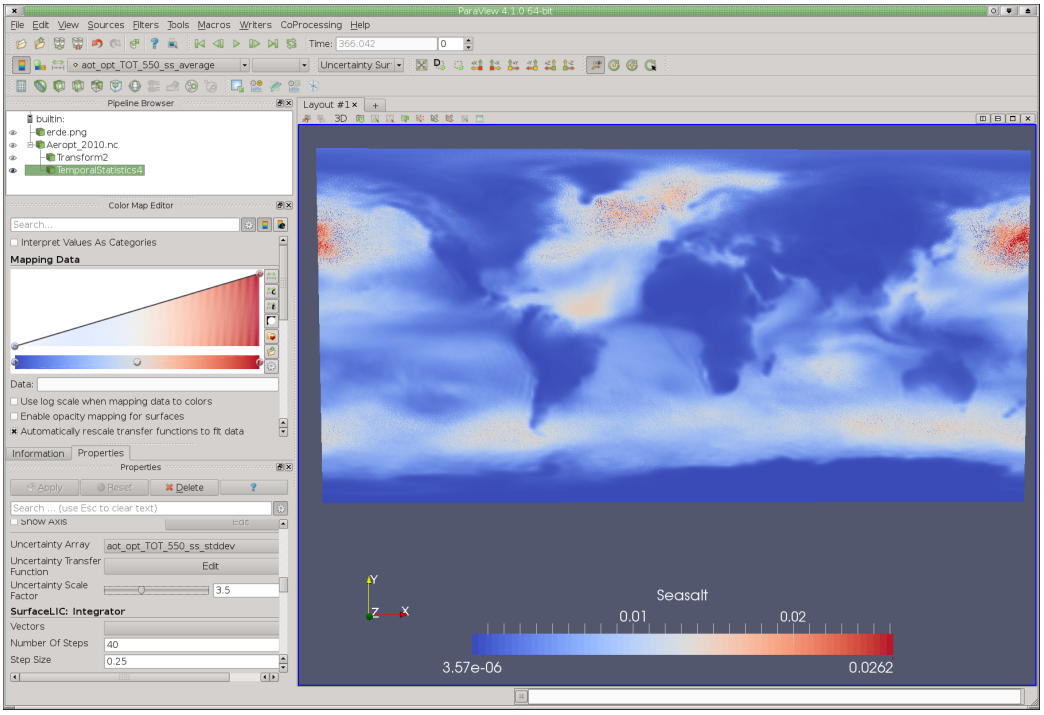

Figure 37: Temporal Statistics and Uncertainty Visualization.

Now you can use your knowledge to visualize this data using the uncertainty mapping technique discussed earlier. Choose here the average for sea salt and an *Uncertainty Surface* for representation. Furthermore, chose the sea salt standard deviation as uncertainty array and change the transfer function in a way that only high values are classified as uncertain, and you receive a visualization such as depicted in Figure 37.

Compare your visualization with the included example Paraview state *example_4c.pvsm*.

4.4 LINE AND BAR CHART VIEW

A Line Chart View plots the data in a different way, and allows thereby a deeper insight. To see the distribution of data values, one has to add the filter **PlotData** from the Filter menu to the visualization pipeline. If you continue from the previous example, please delete the *Temporal Statistics* module and switch on the *earth.png* as well as the *Transform* module, and add a *Plot Data* module right after the last module. Clicking on the **Apply** button automatically splits the visualization in half and opens an additional line chart view that shows the distribution for all five aerosols. Please disable the four other concentrations, and only visualize the distribution of SS (sea salt), refer to Figure 38.

Now we can use this distribution to select a subset of the data, for instance, only some larger values, to see in which areas of the earth the majority of sea

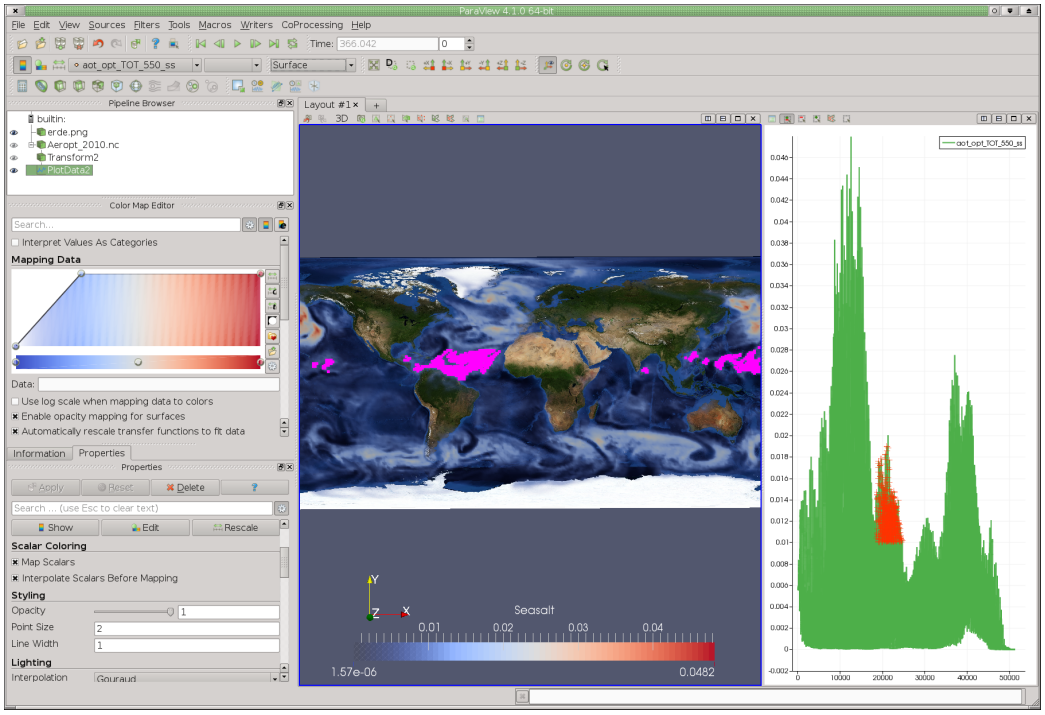

Figure 38: Plotting Data Distribution for sea salt and selecting certain data values.

salt emanates. In the module make sure that only sea salt (SS) is enabled in the line chart view. Figure 39 shows a close up of the data selection tool available in the line chart view. The first icon lets you adjust the settings for this viewport, while the remaining five are for data selection and de-selection: Toggle Selection, Remove Selection, Add Selection, Draw Selection Polygon, Rectangle Selection.

Figure 39: Paraview Data Selection Menu.

To link both views together, select Plot Data as data source for both views. Select *Add Selection* and *Rectangular Selection* and make a selection in the line chart view. The y-axis shows the concentration of sea salt, while the x-axis is simply the cell id number. Please make a selection, and compare your visualization with Figure 38, as well as with the included example Paraview state file *example_4d.pvsm*.

Sometimes it is also important to plot and visualize the values along an intersection curve. In order to do this, select the *Transform* module in the visualization pipeline and add the filter from the menu **Filter -> Data Analysis -> PlotOnIntersectionCurves**, and use the mouse to move the slice to an interesting position, in our example we plot a line over Nort America to Europe. After hitting **Apply**, the Line Chart View changes its visualization, and now plots the data values along this slicing curve, refer also to Figure 40. The slicing curve can be arbitrary,

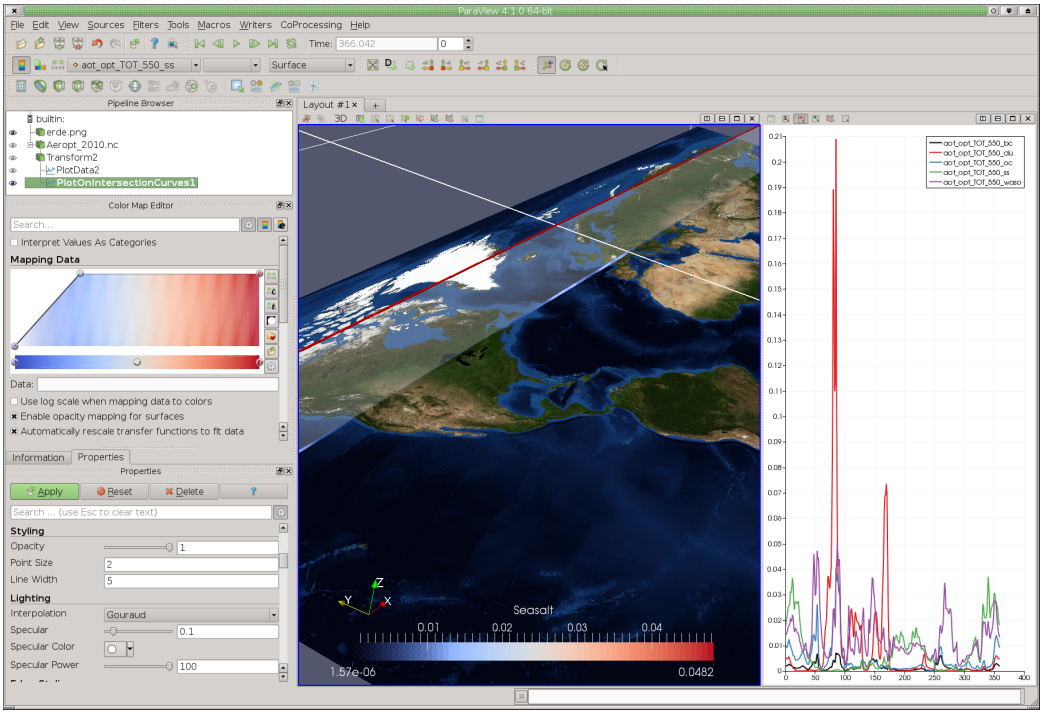

Figure 40: Plotting Data along an Intersection Curve.

and is easy to readjust. It now shows the data for all five aerosols. If you enable in the menu the auto apply option, you can simply slide through your data more intuitively, while the plot is always updated according to the intersection curve chosen. Compare your visualization with Figure 40, as well as with the included example Paraview state file *example_4e.pvsm*.

You can also select a subset of the data, either through the spreadsheet, the 3D user interface, or by using a scatterplot or line chart, and plot this selection of data values over time using the filter **Filter -> Data Analysis -> PlotSelectionOverTime**. This allows you to inspect the development of the selected area over time.

4.5 PLOT MATRIX VIEW

Scatterplots are a great tool to explore and visualize dependencies and relationships between two variables. Paraviews Plot Matrix View goes one step further, and visualizes the scatterplots and histograms of several variables at the same time, refer also to Figure 41. In this visualization, the scatterplots and histograms for biological and organic carbon, dust, sea salt and sulfur / nitrate are displayed. In total there are 5 histograms and 10 scatterplots. A scatterplot visualizes the relationship between two variables, as in our example for biological and organic carbon, and draws a point for each matching pair. This way, one intuitively sees the data distribution. The size of the points drawn can be adjusted in the proper-

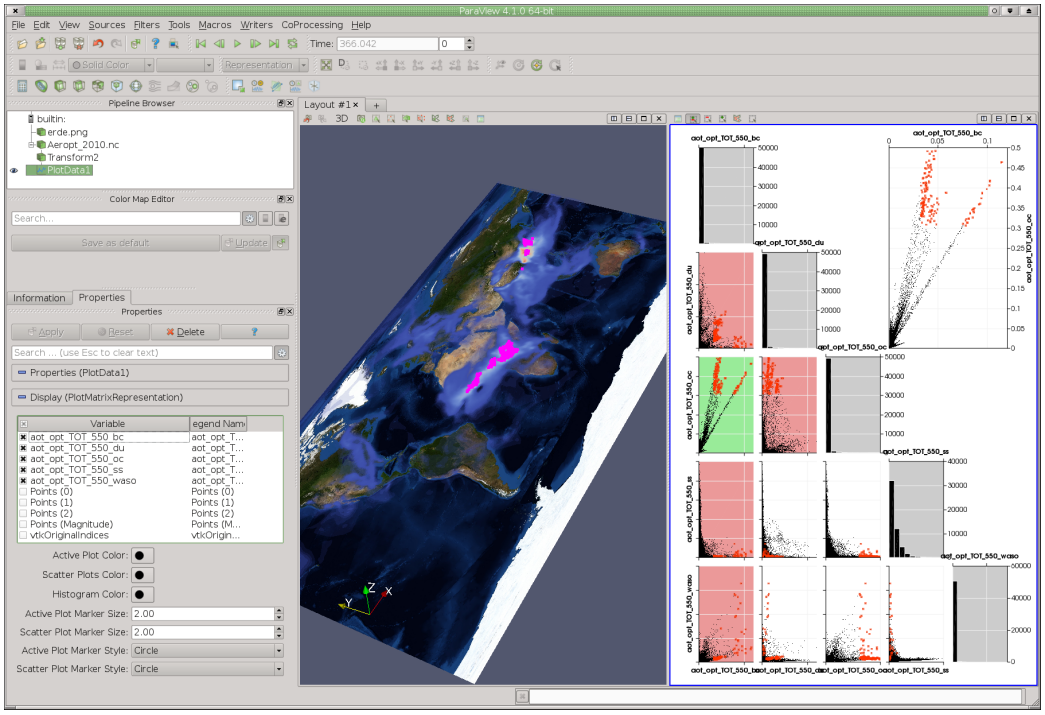

Figure 41: Plot Matrix View in Paraview.

ties window, and it is recommended to use a point size smaller than 5 for larger data sets, as otherwise, the points are too bulky. The active scatterplot has a green background, while the active row/columns are drawn in red.

In order to continue with the example, either reset Paraview and start with example Paraview state file *example_4d.pvsm*, and / or continue from the last session, and delete everything after *Plot Data*, and also delete all Views, except the 3D viewport. Now vertically slit the viewport and add a new view a *Plot Matrix View*, and enable *Plot Data* for display in this viewport, as well as for the 3D view. Please chose all five variables for display in the *Plot Matrix View*, and reduce the *Plot Marker Size* to about 2, refer also to Figure 41.

To make a selection, the six little icons on the top left of the viewport can be used, refer to Figure 39. We have used these selection tools previously, and would now like to make a polygonal selection of the higher biological/organic carbon concentration, see also Figure 41. Our selection is also visible in the other scatterplots, as well as in the 3D view – if *Plot Data* is selected – , and allows one to easily compare and visualize dependencies. As we made a selection for high carbon values, chose either biological or organic carbon as variable for display in the 3D view. It appears that high values of biological carbon, dust and organic carbon are linked together. This type of visualization is also called Linking & Brushing, as the different views and data representations are linked together, i.e. changes in one view also appear as changes in another view. Brushing means that

one selects a subset of data points, and uses a third variable for data visualization at these positions.

Compare your visualization with Figure 41, as well as with the included example Paraview state file *example_4f.pvsm*.

4.6 PARALLEL COORDINATES VIEW

Equally powerful and important is the *Parallel Coordinates View* in Paraview, which also visualizes the interdependencies between various data variables. To continue, close the Plot Matrix View and – this time – split the screen horizontally, not vertically, and add a *Parallel Coordinates View* to the visualization, and enable the *Plot Data* module as data source for this display, refer to Figure 42. Again, enable all 5 data variable, and deselect (if selected automatically) the vtkOriginalIndeces variable. The last selection is still available, and the points selected using the plot matrix view are also selected in this view. But as we would like to select something else, deselect all points, and make a selection of the highest dust values only. To deselect a range, click on the selection, and then, on they appearing gray box. To make a selection, activate the add and the rectangular selection boxes and simply draw with the left mouse button over a range of one variable.

In this parallel coordinates view, a line is drawn between each data point, and visualizes the value for the individual data points at these locations. For instance, if a point has a value of 0.01 for BC and a value of 0.6 for dust, then a line is drawn

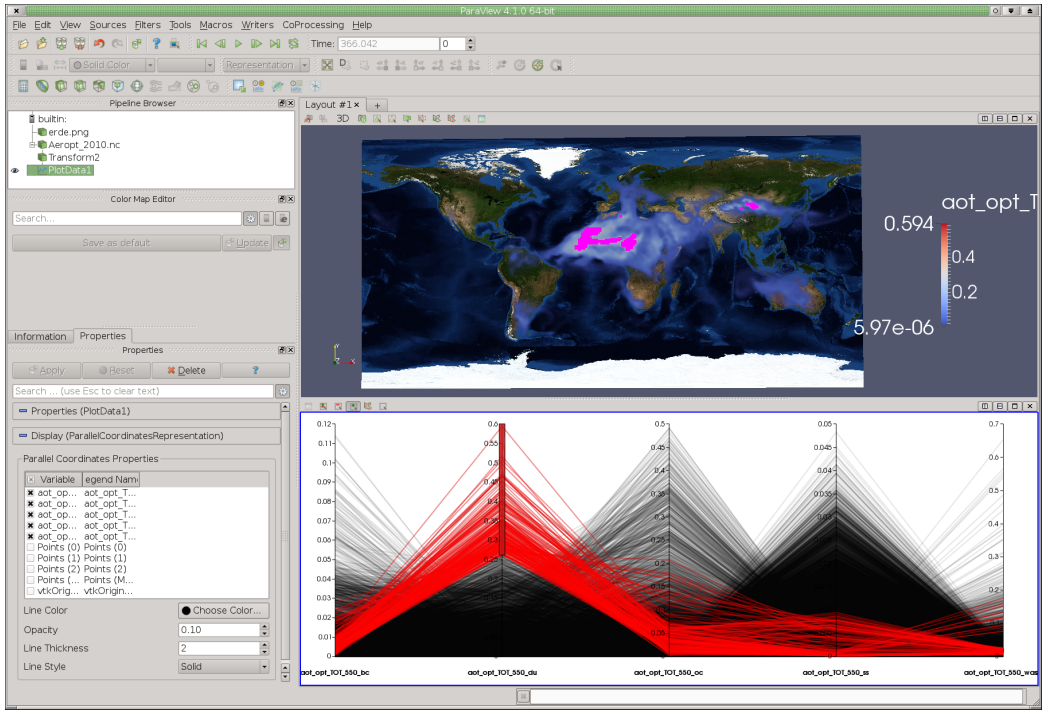

Figure 42: Paraviews Parallel Coordinates View.

between those values. The darker the plot, the more connections – hence similar data points – there are. Using the mouse, one can draw a selection rectangle, to select a range of a certain variable, as in Figure 42 high concentrations of dust are selected. The red lines visualize the selected data, and clearly show that high values for dust do not overlap with high values for sea salt, carbon or sulfates, i.e. this variable has no direct relation to any other variable.

Play around with different selections and variables and try to learn more about the data provided. Compare your visualization with Figure 42, as well as with the included example Paraview state file *example_4g.pvsm*.

5 CURVILINEAR AND UNSTRUCTURED DATA

CLIMATE science uses a variety of different lattices and grids, depending on the simulation model and the research quests asked. One strength of Paraview is that it is not only well equipped to read and work with data sampled on a regular Cartesian grid, but it also handles unstructured data sets quite effectively. To read netCDF data, there are a variety of plugins available to import these data sets into Paraview. So far, we have worked with the standard netCDF CF 1.0 reader, that reads regular and rectilinear netCDF data sets. All currently available netCDF formats for Paraview 4.1 are:

- A regular netCDF CF 1.0 reader
- A netCDF MPAS reader for MPAS data
- A netCDF POP reader netCDF parallel ocean program data
- A netCDF CAM reader for reading CAM data
- An ICON netCDF reader for unstructured ICON data sets

In the following we will have a closer look on the ICON model, as well as on the curvilinear tri-polar ocean model MPI-OM.

5.1 UNSTRUCTURED ICON DATA

The ICON grid and model, as is depicted in Figure 43, is a joint development between the Max Planck Institute for Meteorology[1] in Hamburg and the German Weather Service DWD. Two models, one for the Atmosphere and one for the Ocean, are being developed, and besides several other benefits, the main positive aspects are that the new model does not have any poles, allows an easy refinement of the grid for a detailed weather forecast, as well as allows a much easier coupling between ocean and atmosphere, as both models share the same grid.

To get started with Paraview and ICON, please start or reset Paraview, and load the data sets *icon_ocean.nc*, which is oceanic test data with several cell and one point variable. In the following dialog, please tell Paraview that the data is on the ICON netCDF format and continue. In the following dialog, you have to specify the correct dimensions for the data set. If you started Paraview from a directory in which you have write access, then in the lower half, there will be a text window with *ncdump -h* in it, to help you set the correct values. Please make the specification as it is depicted in Figure 44 and click on **OK**. If you can't close

1 http://www.mpimet.mpg.de/en/science/models/icon.html

Figure 43: ICON Grid with refinement over Europe.

the dialog, then some of your settings are not correct. Paraview only continues if the dimensions are specified correctly.

Now an ICON netCDF module has been added to the pipeline, but no data has yet been read in. We need to further specify in which representation we woul like our data, as well as which variables should be read in. Figure 45 shows some more details of the possibilities. Top left is a equidistant cylindrical representation, while the top right shows a Cassini projection. The later is well suited for an undistorted visualization of both poles, such as for an ice or snow visualization. Both lower views show a spherical projection, with the right one having the missing values (landmass) cut off.

The ICON plugin is developed at DKRZ and provided as is. However, we still work on completing, extending and perfecting this plugin.

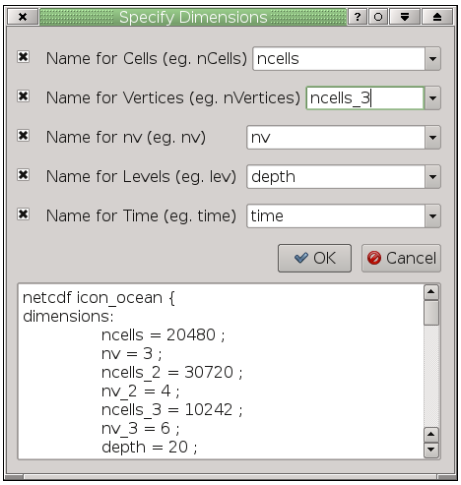

Figure 44: Paraview ICON Dimensions Dialog.

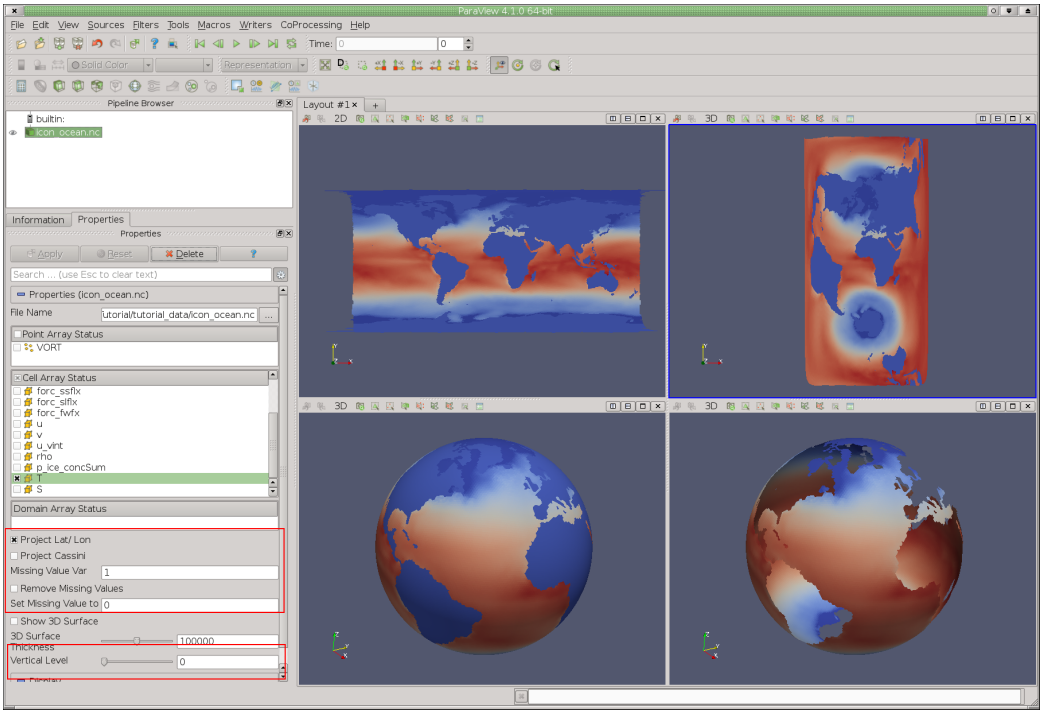

Figure 45: Paraview ICON Dimensions Dialog.

5.1.1 *Spherical Projection with Orography*

After this short introduction, we would also like to work a little bit with some ICON data and use some of the knowledge that we acquired in the previous Chapters. To start, please reset Paraview and read in the example Paraview state file *earth_orography.pvsm* from the tutorial folder. This provides the setting for our visualization with the Earth Orography. Now load the ICON data set *icon_ocean.nc* as ICON netCDF data into Paraview. Make sure that all dimension names are properly defined, refer to Figure 44. Before clicking on the **Apply** button to read in the data, move the *Vertical Level* to 0, leave the projection as is (i.e. spherical), check the box *Remove Missing Values*, and select all point and cell variables. After clicking apply, you should see the reconstructed surface, with the landmasses cut out.

The data set contains one point variable (vorticity) and many cell variables: salinity, temperature, velocities, ice concentration and several more. Play a little bit around and explore the individual variables and try to find suitable color tables. Visualize the underlying grid by changing the representation to *Surface with Edges*, and use the calculator to compute two vectors:

- Water Velocity (`u_acc*iHat+v_acc*jHat+0*kHat`)
- Ice Velocity (`ice_u_acc*iHat+ice_v_acc*jHat+0*kHat`)

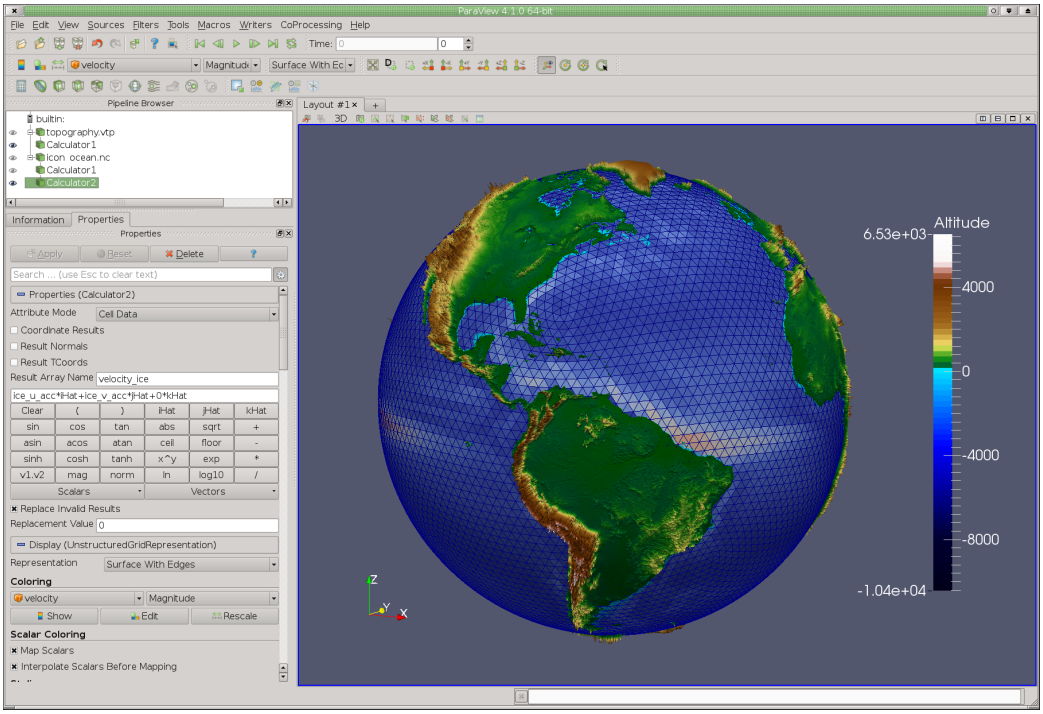

Figure 46: Paraview ICON Dimensions Dialog.

Compare your visualization with Figure 46, as well as with the included example Paraview state file *example_5a.pvsm*.

5.1.2 *Analyzing and Exploring the Data*

Now we can continue and look into the data using the techniques discussed in the last Chapter. However, as we have not only point variables, we have to convert our cell data into point data. This can be done very easily using a filter module.

Either continue from the last session, or load the example Paraview state file *example_5a.pvsm*. Now go to the Menu and add **Filters -> Alphabetical -> Cell Data To Point Data** to the visualization pipeline, check the option *Pass Cell Data* and click on **Apply**. Now all your Cell variables are also available as point variables. Now go to the Menu and add **Filters -> Alphabetical -> Plot Data** to plot the data. Horizontally split the viewport, and add an *Parallel Coordinates View* at the bottom, and enable for this view Plot Data as display Source. You may notice that there are really to many variables being displayed at the moment to see anything meaningful, therefore go to the Plot Data module and disable all, except salinity (s_acc), temperature (s_acc), velocity (magnitude), ice_velocity (magnitude) and sea surface height (h_acc), refer to Figure 47.

We would like to identify really heavy water, therefore, in the parallel coordinates view, make to selections: low temperatures and high salinity. You immediately see the connections to the other data values, especially for sea sur-

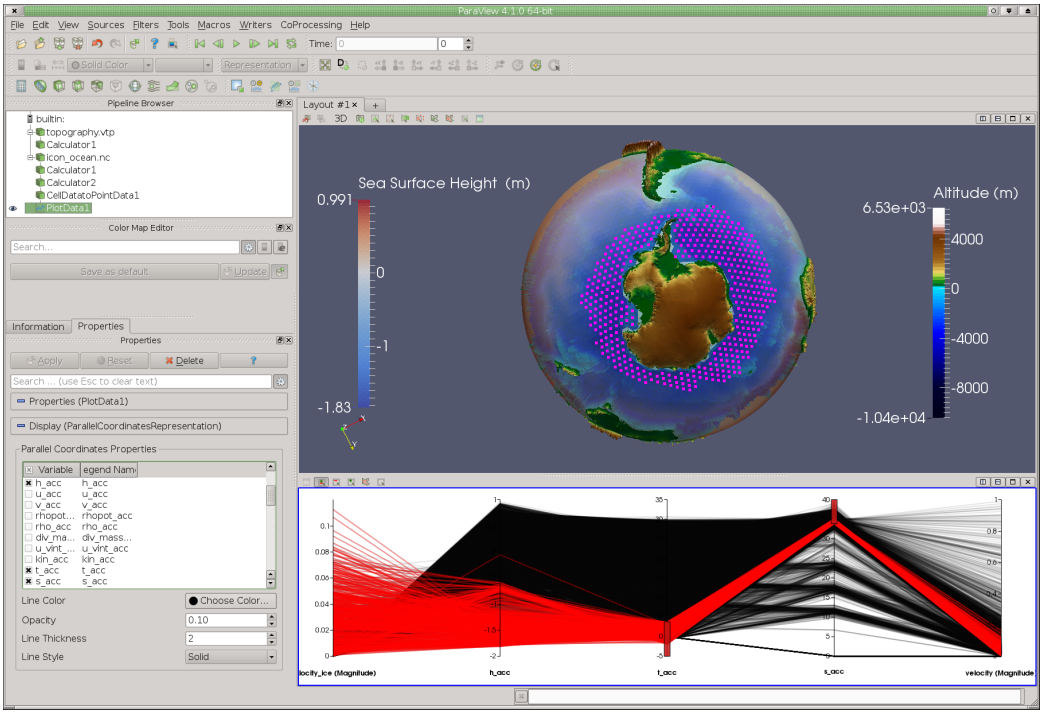

Figure 47: Plotting ICON Data.

face height. In the 3D viewport, also make the module *Plot Data* the source for display, and choose the variable sea surface height (h_acc) for display. The areas that have been selected by us are the arctic areas, in which we find cold and salty, heavy water. This becomes even more prominent if we compare our selection with the visualization of sea surface height. In our visualization, we have also made the Surface slice transparent to see a bit from the structure of the ocean. Compare your visualization with Figure 47, as well as with the included example Paraview state file *example_5b.pvsm*.

Play around with different selections and different visualizations.

5.2 CURVILINEAR MPI-OM DATA

Paraview is also able to read curvilinear grids, which are often employed in ocean models, such as MPI-OM. This data can be read with the regular netCDF reader, however, the data is displayed best using a spherical projection, as otherwise artifacts may occur. For our last example, please reset Paraview, and browse to the tutorial folder to read the netCDF file *mpi-om.nc*. This data emanates from a very high resolution 3D ocean simulation, which depicts the north Atlantic. To further reduce the data size, we restrict ourselves to one time step only, and only the variables temperature and salinity. While reading the data, please make sure the the checkbox *Spherical* is checked, and also activate *Replace Fill Values with NANs*. As the data is quite large, the reading and grid reconstruction may take

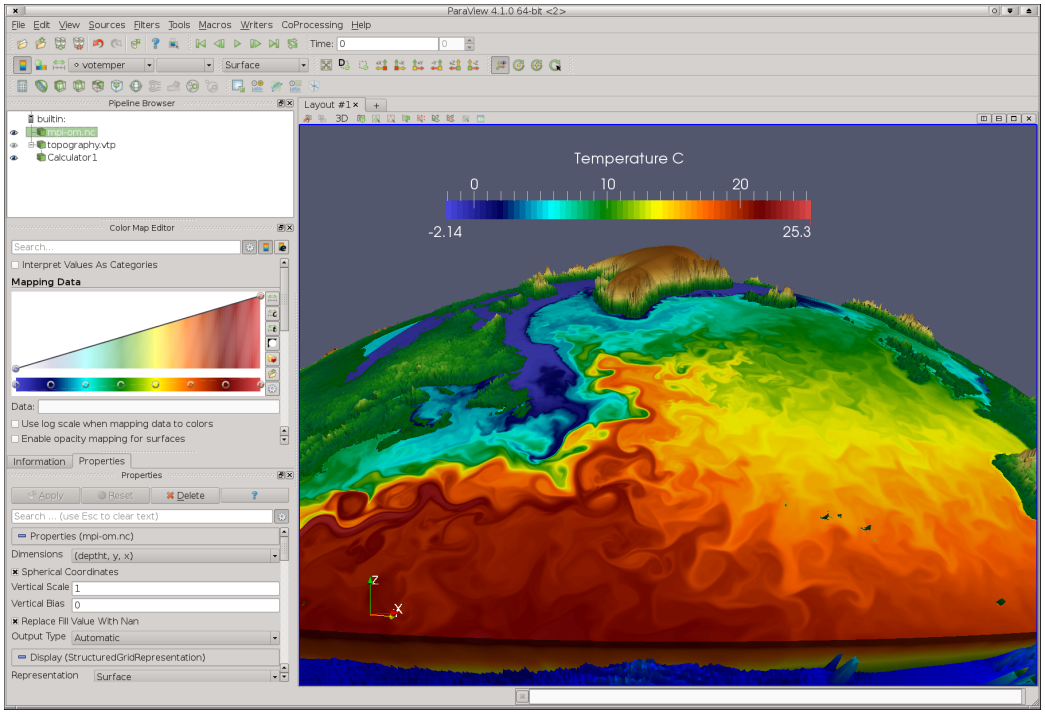

Figure 48: Visualizing curvilinear MPI-OM data.

a while. If the data is loaded, switch from Outline to Surface, and explore the variables. To enhance the color contrast, you can select *Rainbow Desaturated* as color table for the temperature data (votemper).

To annotate the visualization, we will load the topography data. Do not worry, if you won't see anything from the beginning. We first need to adjust the scale a bit. Therefore, load the data *topography.vtp*, and the therein contained variable altitude. Use the *Calculator* module to perform a displacement mapping: Add a Calculator, select the checkbox **Coordinate Results**, and in the calculator field you enter the following formula:

$$(1 + (\text{altitude} \;/\; 6370000) * 100) * (\text{iHat} * \cos(\text{asin}(\text{coordsZ})) * \cos(\text{atan}(\text{coordsY} \;/\; \text{coordsX})) * \text{coordsX} \;/\; \text{abs}(\text{coordsX}) + \text{jHat} * \cos(\text{asin}(\text{coordsZ})) * \sin(\text{atan}(\text{coordsY} \;/\; \text{coordsX})) * \text{coordsX} \;/\; \text{abs}(\text{coordsX}) + \text{kHat} * \text{coordsZ}) \tag{5.1}$$

Now click on the **Apply** button, and in the advanced settings, you enter a scaling factor of 5872 for all three axes. Please also choose the color table *colormap_orography.xml* from the tutorial folder, for a more realistic coloration. If you add a color bar annotation to the viewport, your visualization should look similar as depicted in Figure 48. Compare your visualization with the included example Paraview state file *example_5c.pvsm*.

6 CREATING ANIMATIONS

VISUALIZATION is often employed to analyze unknown data sets in the search for interesting structures, to derive qualitative and quantitative information hidden between the data values. If this data analysis is complete, another goal moves into focus: visualization to communicate and to represent the results. This involves the creation of screenshots and animations, which shall be discussed within this Chapter.

6.1 CREATING IMAGES AND SCREENSHOTS

6.2 CREATING ANIMATIONS

7

PARALLEL AND REMOTE DATA VISUALIZATION

ONE of the great advantages of Paraview is the possibility to be run in a client-server setting. That means the GUI and interaction input runs and is handled by a small client on the users system, possibly a local, machine, while all the data and graphics intense work is loaded off to one or more servers working in parallel. Here we will discuss how to use this feature within the visualization environment of the DKRZ and on the HALO system. Paraview allows three different server settings, a data server, a graphics server and a combined data-graphics server. We will only look at the latter example.

8

IN-SITU VISUALIZATION WITH CATALYST

CATALYST is a special extension to Paraview, that allows an in-situ visualization of data while the simulation is still running. This chapter explains the idea and principles of catalyst, as well as explains how to use catalyst together with the ICON simulation model.