WaterVis: Water Measurement

Project documentation

Petra Eibl, Thomas Schneider

02.06.2009

Overview

The water supply company of a city in China has commissioned KSC to build a remotely controlled system for monitoring of the water quality. The water quality is measured by measuring probes supplied by the company s::can.

The main tasks of the system are retrieving, storing, visualization and delivery of data from s::can measuring probes. The following individual components are used:

- SCADA system
- Interface server
- database
- s::can server
- Visualization client

The SCADA system will show real time data from the water group, for example the water level of a water tank. The SCADA system also receives data from another SCADA and the water quality measurements from the s::can probes. For this purpose, an interface application is needed. It serves as the interface between other components and the SCADA system.

The actual measurements are stored in the database. In the present project, a Microsoft SQL Server 2008 is used for this task. The s::can server writes the measurements into a database and the visualization client reads from the database. The main task of the client is the visualization, using the so-called fingerprints from the s::can system.

All programming was conducted in C# with Microsoft Visual Studio 2008.

Interface server

How to write into the SCADA system?

A request for the project was to write values to the SCADA system. The destination for the data in the SCADA system is the so-called PVs (process variables). We have to write data to PVs from two data sources:

- From the s::can measurement system: A parameter file is generated for every fingerprint file. The values have to be transferred to the SCADA system.
- An interface has to be written from the legacy SCADA system to the new system. The old SCADA system is needed for the transition period until all stations from the legacy SCADA system have been migrated to the new system.

The first task in the project was thus to find a way to write a value to a PV in the RESY-PMC system. The problem was that the documentation for this part of the system is very poor and those employees that had sufficient knowledge of the system are no longer with the company. As a result, the only way to find out how to write data to system was to keep on trying until it worked. We found two ways of writing into SCADA: OPC and ODBC.

OPC

OPC stands for OLE (object linking and enabling) for process control. OPC is based on OLE, COM and DCOM, which were developed by Microsoft for Windows. It consists of standardized objects, interfaces, and methods for use in process control and automation applications.

There are some descriptions in the internet on how to write to an OPC interface, even for .NET environments, but this description did not work for the RESY-PMC system.

ODBC

We received a hint from KSC, who said that it should also be possible to write to SCADA via ODBC. ODBC stands for open database connectivity. ODBC provides a standard interface for using database systems, independent of the database system, programming language or operating system. The ODBC interface of the SCADA system provides read and write access to the table COM_PV. The actual value of a given PV is stored in this table. When you write to such a PV, the SCADA system takes over all the functions needed (for storing the historical values of a PV).

The program that writes the values to the system has to run on the system where the client installation of the system is installed and where a user with sufficient credentials has logged on.

.NET remoting

The server that writes data from s::can into the database has to run on a different machine. The parameter values must be written to the SCADA system too, so a solution is needed that lets the server that writes the s::can measurements into the database communicate with the ODBC interface. The solution used for this task in the project is .NET remoting.

.NET remoting is an interface for programming inter-process communication. .NET remoting enables you to build widely distributed applications easily, whether application components are all on one computer or spread out across the entire world (Microsoft, 2009). It is very well integrated into the

.NET framework and the C# language, so it was the easiest, fastest and most maintainable method compared with other possibilities, such as Web Service or OLE.

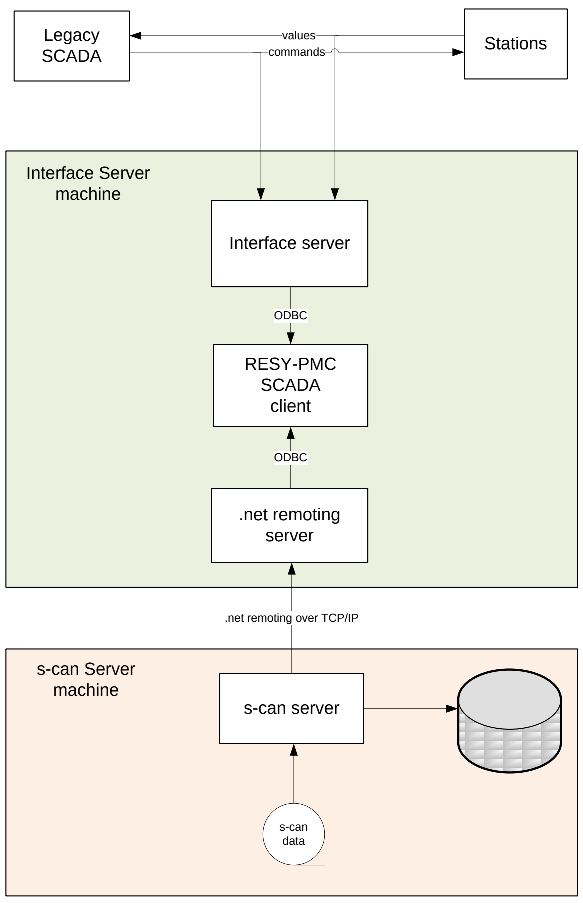

Figure 1: Overview of the interface components to the SCADA system

All components involved are shown in Figure 1. The machine borders are marked by the green and orange boxes. The smaller boxes in the colored boxes are the software components that are needed for the given task.

How to read data from the legacy SCADA system?

The old legacy SCADA system communicates with its measurement station via two serial ports. The commands to the stations are sent at one serial port and the answer from the station is received at the second serial port.

Our interface server listens at both serial ports. All commands to the stations are stored in a data structure (hash table) with a time stamp. There are two possibilities for the answer from the station:

- If the answer is not received within a given time interval, an alarm should be sent to the new SCADA system. Otherwise,
- if the answer is received within the interval, the value is interpreted as a decimal value (the station always sends four byte values that have to be interpreted as a decimal value within a given interval) and sent to the new SCADA via ODBC interface. Afterwards the command is deleted in the hash table.

Three concurrent running threads have to be implemented in the server process:

- A thread to listen for all commands from the legacy SCADA system to its stations. All commands have to be stored in the hash table.
- The second thread checks periodically whether the time stamps on all entries in the hash table are older than a defined value. If they are older, an alarm is sent to the new SCADA.
- The third thread listens to all communication from the stations to the legacy SCADA system.

We chose a synchronized data structure to avoid problems with concurrent accessing elements in the data structure.

All tasks by the server are recorded in a log file. To avoid concurrency problems when writing to the log file, a mutex (mutual exclusion) was used. In (mutex, 2009) a mutex is described as a program object that allows multiple program threads to share the same resource, such as file access, but not simultaneously. A mutex is a global variable that can only be zero or one. It is used to prevent more than one process entering a critical section in a program. If a thread wants to enter the critical section while another thread is in the critical section, it has to wait until the mutex is released.

s::can Server

The data from the s::can system is provided as an ASCII-file in the file system. The main task for the s::can server is to read the files and store the data in a database.

The database used is a Microsoft SQL Server 2008 Express. The advantage of the Microsoft SQL Server is its perfect integration into the development environment and programming language. With LINQ, Microsoft developed database access with full syntax checking. The data can be accessed via objects. Results of queries can be enumerated using the “foreach” statement. The Microsoft SQL Server 2008 Express edition is a free edition of the SQL server. The limit for the database size of 4GB can be handled by creating more than one database. In this project, we created one database for each s::can probe. The name of each database is the serial number of the probe.

The tables created and the relations between them are shown in Figure 2. “FPrintRow” is the abbreviation for fingerprintrow. It stores the information for the header of one line of the fingerprintfile from the s::can probe. “FPrintcol” stores the information for every column of the row. The values from the parameter file are stored in “Parameter”. “ParameterTypePV” contains the configuration of which parameter should be sent to which PV in the SCADA system.

Furthermore, a database with general configuration information is created. The parameter types and the stations of the system are stored in this database.

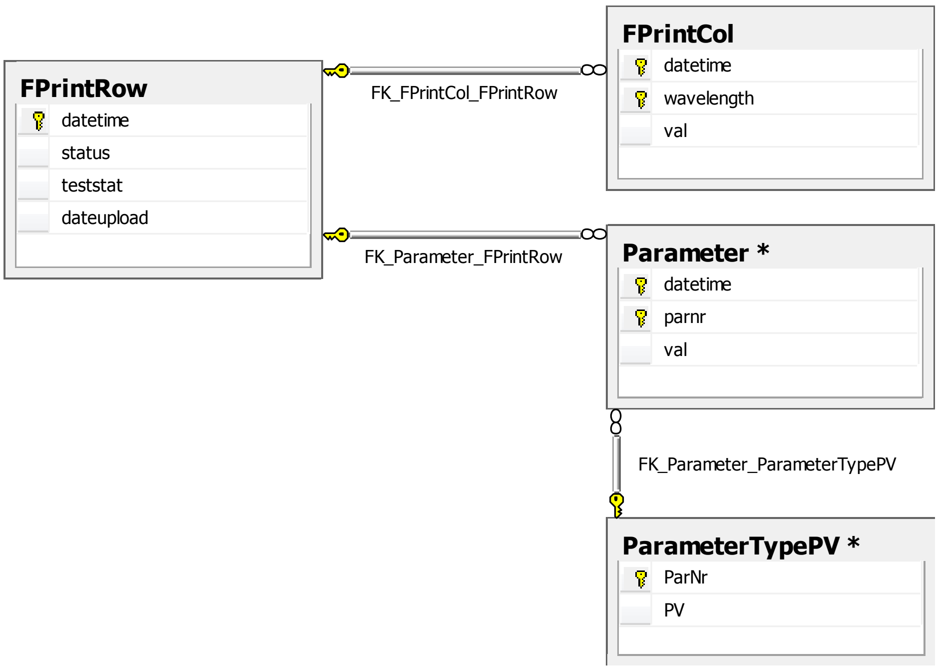

Figure 2: Database for each probe

The server performs the following tasks:

- Reads all new files from given directory
- Stores new entries (fingerprint and parameter) in the database
- If the date and time of the entry in the file correspond to the actual time -> send the value of the parameters to the PV (corresponding to the configuration in the table “ParametertypePV”). The server does this by calling the interface server by the .net remoting interface.

Visualization of the measurements

Visualization takes place on two sites: in the RESY-PMC system and in the Watervis client.

The SCADA system used always shows the latest real-time data in the diagrams and overview screens. The main purpose of SCADA (and any other SCADA system) is thus to provide information on real-time values and historical values of measurements of any kind. In the project ,the SCADA system visualizes water pressure, water levels from miscellaneous sensors and water quality measurements from the s::can probes. The water quality measurements are transmitted through the s::can server and the interface server, originating from the values of the s::can parameter files.

The primary function of the Watervis client is to visualize the fingerprints of the s::can system. A fingerprint shows the original measurements of the spectrometer probe. The spectrometer probe flashes at equidistant time intervals and measures the extinction of the light on wavelengths between 200 nm and 750 nm. The parameter values are derived from the fingerprint. Examples of parameter values are, for example, the mass of NO2-N in the water or the pH value of the water. In other words, the parameter values are the interpretation of the fingerprints.

In normal cases, the user only checks parameter values in the SCADA system. If there is any conspicuousness in the values, the user can check the corresponding fingerprint directly. Technically, the Watervis application is called from the SCADA client, with the serial number of the probe as command line argument.

In the user interface, Watervis provides three different views of the fingerprints:

- Visualization of one fingerprint at a specific moment. Normally, the latest measurement is shown, but the user can also go back in time.
- A surface view where the measurements are visualized as the surface of many measurements within a specified interval.
- The analysis view provides the means to filter fingerprints by parameters.

In the application, three different visualization libraries are used:

- Ilnumerics.net
- Microsoft Charting
- GAV

Prefacing of the graphic packages is described and then implementation of Watervis with these packages is annotated.

ILNumerics.net

ILNumerics.Net is a class library for mathematical problems of all kinds. It is open source (download from http://ilnumerics.net) and in C#. It provides n-dimensional arrays, complex numbers, linear algebra, and fast Fourier transformation (1 to n dimensional). The most important features for Watervis are the visualization controls. 2D and 3D graph types are provided. All graphic output is conducted with OpenGL, so the output is hardware-accelerated and very fast.

Arrays in ILNumerics.net

ILnumerics.net uses a special data type for storing data in arrays: ILArray<T>. It is a generic data type. <T> stands for the definition of the underlying data type (float for example). ILArray provides n-dimensional array definition and many operations using them. For example, to define a three-dimensional array measuring 500x300x100 and set all elements to pi, write

```
ILArray<double>.zeros(500,300,100) + ILMath.pi;
```

Any matrix calculation can be performed easily with ILNumerics.net. Thus, linear equation solving with matrices in the form A x = b is also supported.

Visualizations

Simple 2D line plots and 3D surface- and image plots are available in ILNumerics.Net. The controls are very well integrated into Windows.Forms (standard class library for Windows from Microsoft). The use of OpenGL makes them platform-independent (available for .Net and mono) and high-performance.

Firstly, you need a control for the graphic output. The following controls are available:

- ILPanel is the main basic control for all available graphs. Automatic mouse interaction, such as rotating, is supported.
- ILSubfigure wraps ILPanel together with controls like colorbar and titlebar.
- ILFigure is for displaying graphics in a separate form.

To create a graph on an ILPanel, you simply have to create an object of the corresponding graph class, for example ILSurfaceGraph. Several properties of the graph can be set by accessing the attributes of the graph object. The structure is very clear and easy to understand, thus it is quite simple to make modifications to the default output options.

Microsoft Chart Controls

Since the autumn of 2008, Microsoft has offered Microsoft Chart controls for .net 3.5 in a free download. It supports 25 different chart types. Windows.Forms applications and ASP.NET sites are supported. The output is handled with DirectX Library, thus it is hardware-accelerated, but not as fast as ILNumerics.net or GAV.

The Microsoft Chart controls are a set of classes. An overview of all classes is provided in Figure 3 (from (Microsoft, 2008)). Generally, the structure is clear and tidy, but, when using the control, it was sometimes difficult to find the correct position to adapt the output (for example to round the axis label values).

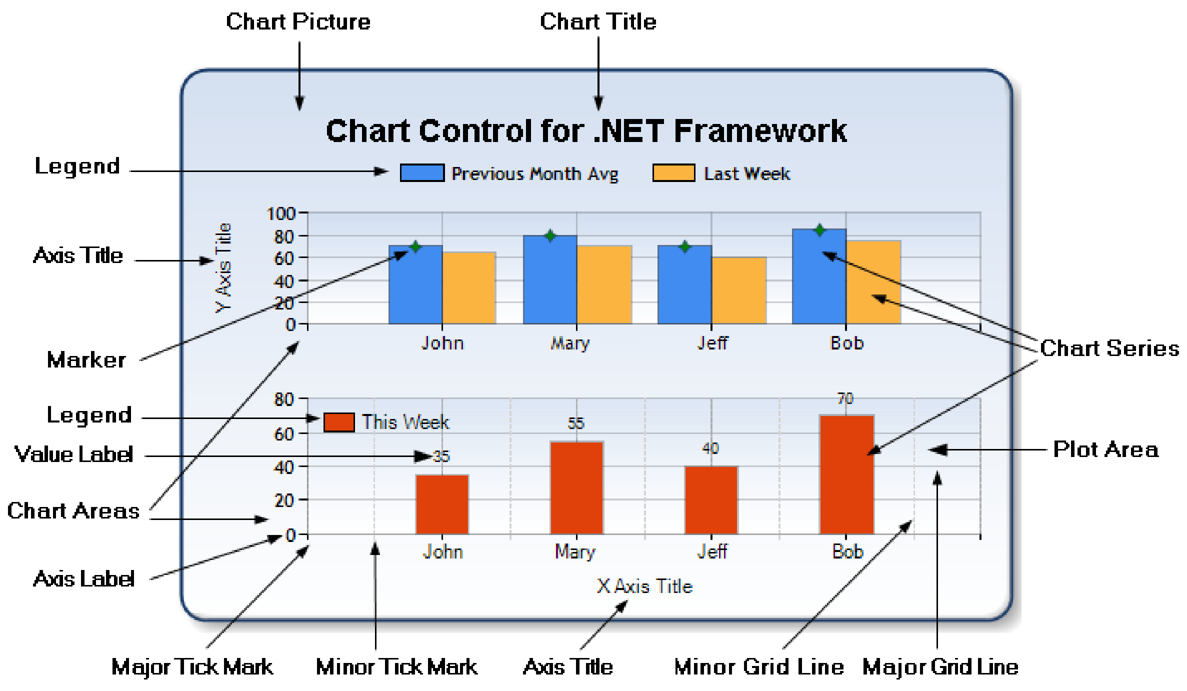

Figure 3: Components of the Microsoft Chart control

In order to insert values into the chart, you have to use the Series class (or the SeriesCollection class if you want to show more than one). In the series, you also set the chart type for output of the series, for example SeriesChartType.Line. From the moment you add the series to the chart, it is also shown automatically. Every property update is also shown automatically, thus you can easily determine the effects of changing a parameter. The library is well documented in help files and a large example file with demos for all chart types. At social.msdn.com there is a large community that will provide answers to any questions relating to Microsoft Chart.

GAV GeoAnalytics Visualization

The GAV framework from the University of Linköping in Sweden was developed primarily for representation of geographical, time-varying and multivariate data. GAV supports interactive influence on the output. All output is conducted by DirectX. GAV is very suitable for mass data because of its speed.

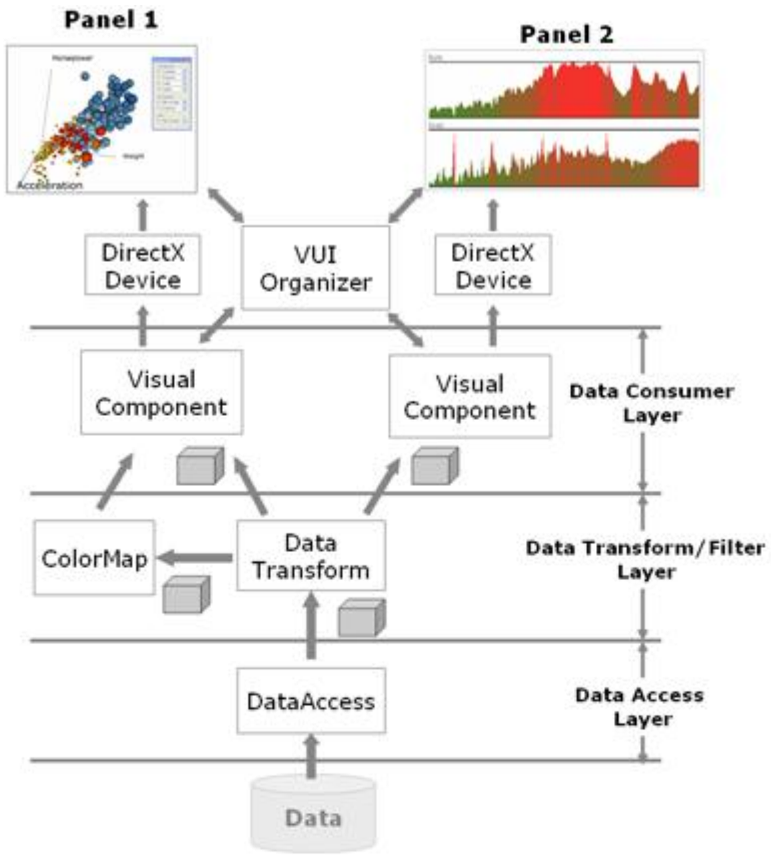

Figure 4: Architecture of GAV

The architecture is shown in Figure 4 (from (Åström, 2008). The data access layer reads data from many data sources (Microsoft Excel, for example). In our project, this layer is replaced by our own access to data (to our database). The data transformation layer provides data structure for the visual components. The class for this data transformation is called cube and is comparable to ILArray from ILNumerics.net.

In GAV, there are two main visualization components - VizComponent and VizSubComponent. All visualization components in GAV inherit from one of these. The purpose of this is to ensure that two or more components can be rendered on the same render surface. The first line of code below shows how a colour legend is added as a child to the scatter plot. The scatter plot is a VizComponent and the colour legend is a VizSubComponent. The next line shows how the scatter plot is added to the ViewManager. The ViewManager class in GAV is in charge of when, where and how all components are rendered. Now, both components will render on the same surface.

```
scatterPlot.AddSubComponent(colorLegend);

viewManager.Add(scatterPlot, panel);
```

Scatter plots (as 2D and 3D) are one of the GAV graph components provided. Ternary diagrams and parallel coordinate plots are the other diagram types available. In the Watervis client, the parallel coordinate plot is used, so this diagram type will be described in more detail.

User Interface

The three views organize by using Tabs. Standard controls from Windows Forms are used. A ComboBox with a serial number of the measurement data is provided above. When a serial number is selected, the respective view is displayed. The serial number determines from which database the data are read.

If the serial number is given as command line argument at the program start, the serial number is the standard selection for the ComboBox.

Time View

In this view, a fingerprint of one specific moment is shown. The user can move back in time interactively. This can be achieved by moving a slider in the area below the window. The curve is displayed immediately so the user can easily find conspicuous curves by moving around in time. Furthermore, the user is able to select a specific date.

A checkbox control specifies whether the latest measurements should be shown. At the start of the application, the checkbox is checked by default. The update is performed automatically. The user must not push a refresh-button or similar. The update is implemented by a timer. The timer checks periodically whether new data is available.

For the time view, Microsoft Chart is our first choice because of the huge number of options for 2D-curves. We also experimented with the free package Zedgraph, but the illustration facilities are poor. For example, we adapted the standard display of the axis. There are major tick marks and minor tick marks. We adapted the increment to a meaningful dimension, and we tried to make the axis restrained, with the curve as the basic element and in the foreground.

Regarding (Tufte, 1983)the fundamental principle of good statistical graphics is: Above all else show the data. So the lines in a graphic, that don´t show data have to be minimized. With the grid in the background, we deliberately departed from this principle because this is a basic orientation for the user of the system. The user can easily find out the wavelength of an abnormality to a standard curve. In order to ensure that it is not shown too dominantly, the grid is marked in light grey.

We also activated the functionality to zoom a section by the user. This function is a user-friendly means of examining the measurements in more detail.

Surface View

The surface view shows the same data as the Time View. In addition, the time is added to the diagram as a new axis. The result is a 3D surface of fingerprints over the time period. An example is shown in Figure 5.

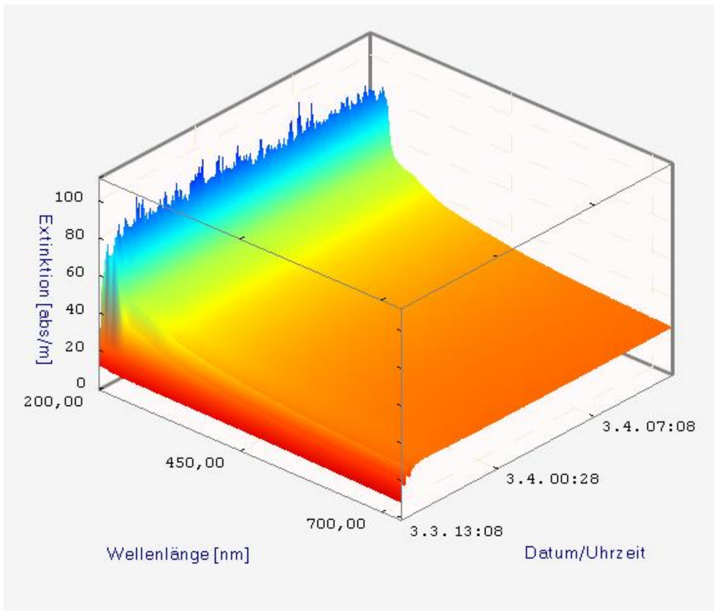

Figure 5: Fingerprints in surface view

The ILnumerics.net visualization control is used for controlling purposes. We found no other suitable control means for the requirements. The control is fast and supports change of the camera-viewing point out of the box.

We did some adapting so that the axes had the correct labels at the tick marks. We also implemented a dialog allowing the user to change the colours in the visualization of the control means. Furthermore, we implemented a function whereby the user can jump directly into an up-view of the surface (see Figure 6). The user can do this by double-clicking on the surface. We implemented a smooth camera pan so that the user understands what is going on when doing a double click.

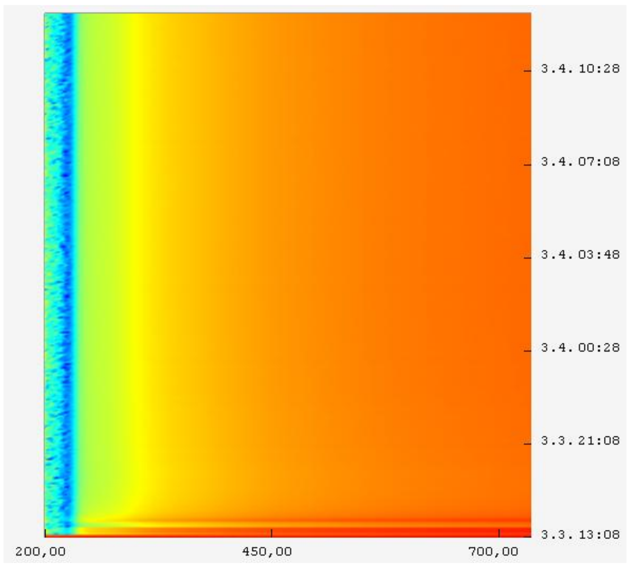

Figure 6: Surface view from the up side

Use of Color the Surface View

One point of discussion was the selection of the color scheme. By default ILNumerics.net use the rainbow colors for the visualization. A potential disadvantage of this color scale is that yellow (a very striking color) is in the middle of the scale. This can be a problem if one is interested in depicting extreme values since attention may be driven to the yellow areas (Silva, Madeira, & Sousa Santro, 2007). Another problem is that 8% of the male an 0,4% of the female peoples are color-blind (Krömker, 2007). Nevertheless the rainbow scale is still the most used color scale in Visualization (Borland & Taylor, 2007).

In our opinion selecting the perfect color scale is impossible. Colors tend to have strong cultural connotations (which unfortunately vary from culture to culture) (Rheingans). Sometime it is also up to the taste of the user which color scale is best. So we implemented a dialog where the user has the opportunity to choose his color scale for the surface view.

Analysis view

In our last view, the user has the opportunity to filter the fingerprints interactively by varying the intervals of parameter values allowed. The view is set up as a combination of a parallel coordinate plot and an array of curves. Firstly, we shall explain the parallel coordinate plot:

Parallel coordinate plot

The dynamic parallel coordinate plot employs a novel methodology to visualize beyond three dimensions by representing each observation not as a *point* in a scatter plot but as a series of unbroken *line segments* connecting parallel axes, each of which represents a different variable (Edsal, 1999). It differs from all other visualization methodologies in a fundamental way. Parallel Coordinates yield graphical representations of Multi-Dimensional relations rather than just finite point sets (Inselberg & Dimsdale, 1981). Furthermore offers the parallel coordinate plot with the integrated filter function the ability for the user to explore the data more in detail.

The parallel coordinate plot in the WaterVis application visualizes the given parameters from s::can and statistical calculations for the given fingerprints. These statistical calculations are the average value of the fingerprint, the variance of the fingerprint, and the application calculates a Nalimov test compared to a given reference curve.

Combination of GAV and Microsoft Chart

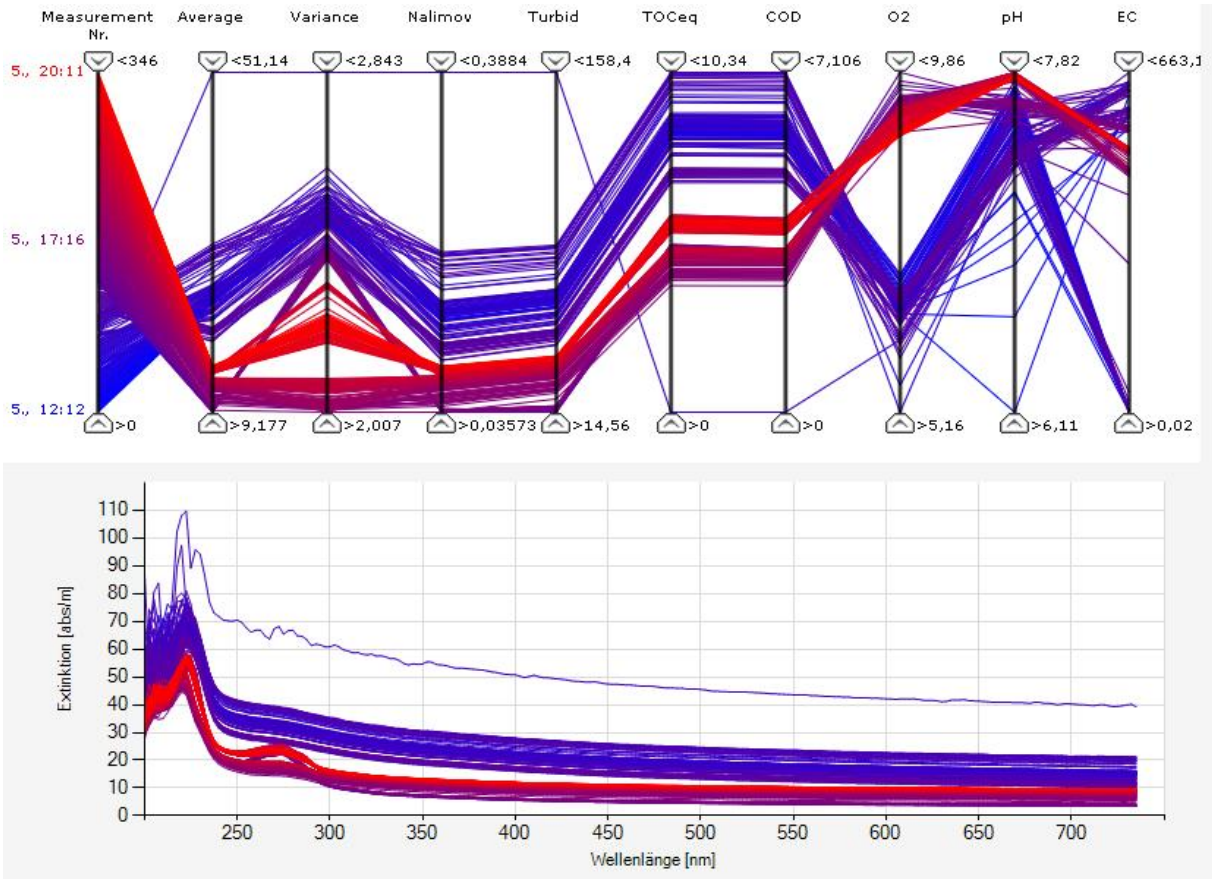

Figure 7: Combination of GAV and Microsoft Chart

The GAV framework from Linköping University in Sweden is used for the coordinate plot. The corresponding array of lines is shown with Microsoft Chart, as in Time View. The reason for this was the good parallel coordinate plot implementation of GAV and the lack of a visualization control means for curves. GAV only supports parallel coordinate plots, scatter plots and ternary diagrams. In view of our positive experience with Microsoft Chart, the decision was easy.

How do we combine these two control means from different vendors? After studying the source code of GAV, the answer was easy. The only thing we had to do was to implement a handler for the “FilterChanged” occurrence by the GAV control. So the handler is called when changing the filter of the parallel coordinate plot. The handler reads the data structures of the parallel coordinate plot control. In the “IndexVisibilityManager“ attribute there is a “Visibility“ array with a Boolean for each entry in the parallel coordinate plot. The handler method reads the values of the array and sets the corresponding visibility attributes from the series of the Microsoft Chart control.

The curves in the parallel coordinate can also be selected. When selecting lines in the parallel coordinate plot, only the corresponding curves in the Microsoft Chart control are shown. This is achieved by implementing the “IHasSelectableIndexes” interface and registering the object to the “SelectionAndPickingManager”.

One large problem with the GAV framework was that only float values are supported for the parallel coordinate plot. This was a problem with the first axis of the parallel coordinate plot. This axis should show the date of the corresponding parameters. We solved this problem by numbering the rows. Furthermore we printed the date values at the start, in the middle and the end of the axis. The colours for this information also correspond to the line colours in the chart.

Future Tasks

The design of the time selection facility in Time View is not perfect. The slider always needs to know the time interval. This control is not intuitive and user-friendly.

In the other views, time selection is also complicated because the user has to define the interval from … to …, showing the date and time in each case. All this data is entered using Windows standard controls, meaning that input is not very convenient.

It is easier to use a control, for example, whose function is similar to the time selection in a video editing program. Two sliders limit the interval. An example of this type of control is shown in Figure 8. A point in time between the limits would be selected by mouse click. This type of control would be preferable as superior choice for all three views, although a challenge for us is the performance because a very large volume of data must be read very quickly to be handled and displayed. The solution to this could be to use aggregate data sets over certain time distances.

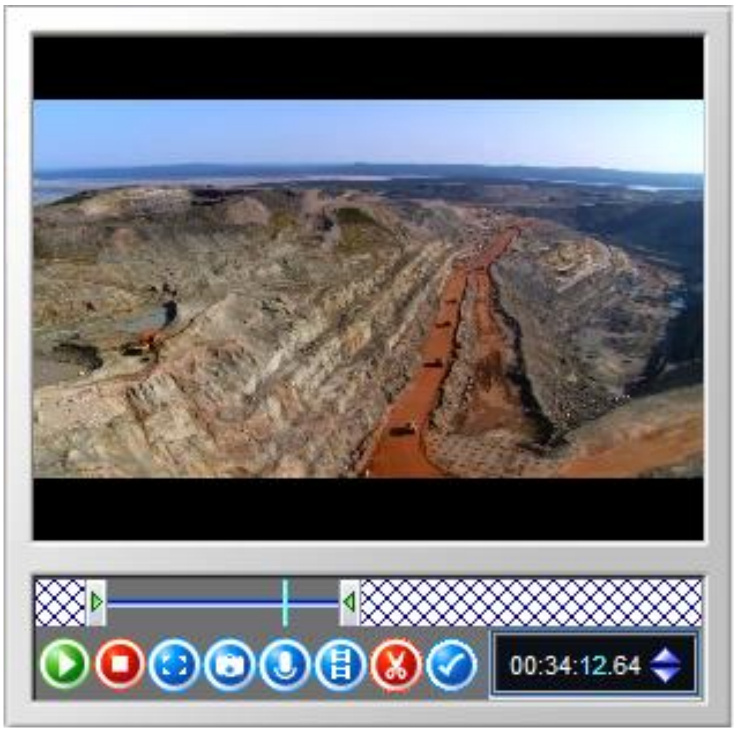

Figure 8: Video cut control in Nero 6

Bibliography

Åström, T. (2008). *GAV*. Retrieved 07 01, 2009, from Linköping University: http://vita.itn.liu.se/pub/jsp/polopoly.jsp?d=13602&l=en

Borland, D., & Taylor, R. (2007). Rainbow color map (still) considered harmfull. *IEEE Computer Graphics & Application 27(2)* , pp. 14-17.

Edsal, R. M. (1999). The Dynamic Parallel Coordinate Plot: Visualizing Multivariate Geographic Data. *19th International Cartographic Conference.* Ottowa, Cannada: http://www.geovista.psu.edu/publications/JSM99/paper.htm.

Inselberg, A., & Dimsdale, B. (1981). *Parallel coordinates: A tool for visualizing multi-dimensional geometry.* Los Angeles: IBM Scientific Center.

Krömker, D. (2007). *Uni Frankfurt, Graphische Datenverarbeitung*. Retrieved 06 3, 2009, from Visualisierung Farbe und Helligkeit: http://www.gdv.informatik.uni-frankfurt.de/lehre/ws2007/VIS/Folien/05-Farbe-und-Helligkeit.pdf

Microsoft. (2009). *.NET Framework Developer Center*. Retrieved July 2, 2009, from .NET Remoting Overview: http://msdn.microsoft.com/en-us/library/kwdt6w2k(VS.71).aspx

Microsoft. (2008). Windows Forms Chart Control. *Chart Elements Overview* .

mutex. (2009). Retrieved June 23, 2009, from Webopedia: http://www.webopedia.com/TERM/M/mutex.html

Silva, S., Madeira, J., & Sousa Santro, B. (2007). *There is More to Color Scales than Meets the Eye: A Review on the Use of Color in Visualization.* University of Aveiro, Portugal: IEETA/DETI.

Tufte, E. R. (1983). In *The Visual Display of Quantitative Information* (p. 92). Connecticut: Graphics Press.