Eurogrpahics/ IEEE-VGTC Symposium on Visualization (2007), pp.1-8
Torsten Möller, Anders Ynnerman and Ken Museth (Editors)

Multi-Resolution Techniques for Visual Exploration of Large Time-Series Data

Ming Hao[1] and Umeshwar Dayal[1] and Daniel Keim[2] and Tobias Schreck[2]

[1]Hewlett-Packard Laboratories, Palo Alto, USA
[2]University of Konstanz, Germany

Abstract

Time series are a data type of utmost importance in many domains such as business management and service monitoring. We address the problem of visualizing large time-related data sets which are difficult to visualize effectively with standard techniques given the limitations of current display devices. We propose a framework for intelligent time- and data-dependent visual aggregation of data along multiple resolution levels. This idea leads to effective visualization support for long time-series data providing both focus and context. The basic idea of the technique is that either data-dependent or application-dependent, display space is allocated in proportion to the degree of interest of data subintervals, thereby (a) guiding the user in perceiving important information, and (b) freeing required display space to visualize all the data. The automatic part of the framework can accommodate any time series analysis algorithm yielding a numeric degree of interest scale. We apply our techniques on real-world data sets, compare it with the standard visualization approach, and conclude the usefulness and scalability of the approach.

CR Categories and Subject Descriptors: I.3.3 [Computer Graphics]: Picture/Image Generation – Display Algorithms; H.5.0 [Information Systems]: Information Interfaces and Presentation – General.

1. Introduction

Large amounts of time-series data and data streams occur in many important application domains such as Finance (e.g., feeds of asset prices) or Network Monitoring (feeds of automatically probed network performance metrics). Regarding the discrepancy between acquisition and analytical processing of large time-related data sets, Eamonn Keogh et al. [WKL*05] noted:

"*Recent advancements in sensor technology have made it possible to collect enormous amounts of data in real-time. However, because of the sheer volume of data most of it will never be inspected by an algorithm, much less a human being.*"

Appropriate visualization of time-series data is a valuable tool for exploring previously unknown data and searching for interesting patterns. The standard approach using bar- and line-charts without support for the large data set problem is ineffective for visual analysis of time-series data: Given the limits of current display devices, we either have to accept overplotting (occlusion) effects in the display, or we have to integrate scrolling interaction. Both effects are not optimal regarding usability effectiveness. Basically, there are two options for addressing the large time-series visualization problem: (1) Numerically reduce the data size by sampling or aggregation, and (2) improve the drawing method used to be more space efficient than standard techniques. Potential drawbacks include that (1) introduces a loss in information, while (2) may lead to visualization metaphors not immediately familiar to the user.

In this paper, we propose to apply data- or time-dependent nonlinear distortion techniques, generating visually aggregated layouts suited to effectively represent long time series in an easy to understand metaphor: Multiresolution grid layouts. Our work is novel in that we propose a non-linear rescaling of the time axis, where the rescaling is based on either a predefined or an automatically obtained distortion profile. The approach is useful in many interesting applications in Network Monitoring, in Business and Finance, and in Science and Engineering. It can be used for online monitoring of data streams with high update rates, and for off-line analysis of large volumes of historic data.

2. Related Work In Time Series Visualization

Visualization for time series data involves a number of interesting research problems. An overview of the topic can be found in [MS03]. Considering the large time series visualization problem, we differentiate methods supporting many time series, or long time series, or both. In our own previous work, in [HDKS05] we addressed the problem of space-filling visualization of many time series of moderate length. Long time series have been previously supported by methods relying on data aggregation or efficient rendering methods. We next recall work from that area.

2.1. Data Aggregation for Time Series Visualization

In [Mun04] the author introduces a system where the user sets the number of bins to aggregate time-series values into prior to rendering. The user is allowed to zoom into any region of the time-series on the fly. The system provides a time scale legend linked to the chart, indicating the level of aggregation. In [Chu98], the author proposes to lay out time-related data in a circular histogram, mapping the time axis to the circumference of a circle. Mapping of multiple values to a same pixel line may occur, which is then resolved by aggregation. In [SSJ05], the authors discuss visualizing sets of non-equal spaced time series arising e.g., in auction bid series. The time scale is transformed to an ordinal index, and each consecutive value is plotted in an equally spaced time-line chart.

2.2. Space-Efficient Time Series Drawing

A number of techniques focus on space-efficient rendering solutions. In [Chu02], time-series values are represented by width and color in bar chart plots of uniform height. The slim chart profile allows many charts to be displayed in parallel. In [WAM01], time-series values are drawn by color and shape attributes along a spiral. The technique is especially recommended for periodic data. Spirals for time-series have also been explored in [CK98]. The scalability of time series displays may be maximized by resorting to the pixel level. In [KKA95] a recursive scheme mapping sets of time series to color-coded raster pixel displays is introduced. In [SK00], a qualitative and quantitative discussion on the limiting factors of certain time-related visualization techniques regarding scalability with data size can be found.

2.3. Our Contribution

Common to the previously noted approaches is that they apply a uniform resolution level for aggregating and drawing, not allowing for locally varying degrees of aggregation. In this work, we introduce the notion of degree of interest (DOI) defined over time series, and use it to generate multi-resolution layouts of long time series. The layouts display data portions considered interesting at high resolution levels, enabling the analyst to quickly perceive important data interval characteristics in the time series. At the same time, the layout displays less interesting data portions at lower resolution levels, providing the context of the full time-series. Please note that the basic idea of this work has been sketched previously in a poster paper [HDKS06].

3. Background

Central to our approach to multi-resolution time-series visualization is the degree of interest (DOI) concept [Fur86]. It defines the notion of interest over a dataset by distinguishing between areas of focus (high interest) and context (low interest). DOI was introduced to model certain naturally occurring distortions like fisheye views [Fur86]. DOI aims at visualization of large data spaces by representing certain data subsets which are in the focus of a user at high level of detail, while keeping the remaining data in context at lower levels of detail (a smaller fraction of the display real estate). DOI defines the interest of any data element x as its a-priori (data-dependent, constant) interest level *API(x)*, minus its distance to the current point of focus, *y*. Given a data set with DOI function, a transfer function is then used to map the data set to the display space. The transfer function relies on the DOI function and controls the level of detail allocated.

In [Fur86], the DOI concept was instantiated for tree-structured data. A distance function on trees was defined, and by thresholding the DOI function the most interesting tree portions were displayed. In the Table Lens system [RC95], a two-dimensional DOI function was defined on the rows and columns of a spreadsheet. The DOI for each dimension was assumed to be a pulse function, with the pulse at a number of consecutive rows and columns in user focus. A linear transfer function was used to scale rows and columns in the tables, and data was shown at the highest aggregation levels permitted after the scaling of the spreadsheet cells. Figure 1 illustrates the Table Lens DOI concept.

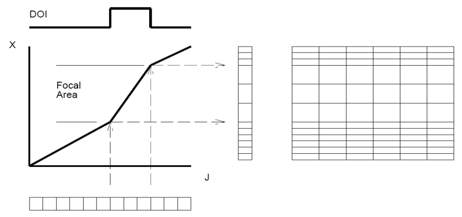

Figure 1: A single-peaked DOI profile applied on a spreadsheet (taken from RC95).

In [Fur86] and [RC95], it was assumed that the focus was interactively set by the user. Here we are concerned with layouts where the focus is automatically determined by the system. We use DOI profiles that are either preconfigured to fit the given application domain, or which are automatically derived from the data by a suitable analysis algorithm. The DOI profile is then applied to display large time-series data by visual aggregation. Guided by preliminary experimentation preformed, we believe that the number of distinct resolution levels in a multi-resolution time-series display should not be too high; a limited number of distinct resolution levels support the user in visually discriminating the different levels. A clear perception is necessary because the resolution level has to communicate (1) the amount of data per display unit (data density), and (2) the relative interest of the data partitions in context to the whole series. (1) is especially critical because it influences the perception of time in terms of duration of periods and localization of values along the time axis. Perception of time is important for visually finding recurring patterns in the time series data, and for comparative analysis in general.

We next describe a static (Section 4) and a dynamic (Section 5) approach for deriving DOI functions from time-series data, map these to appropriate multi-resolution time series displays.

4. Time-Dependent Multi-Resolution Layout Generation

In many application domains there exists a stable time-to-interest relationship. We call this case time-dependent, as the DOI profile is depending only on the time axis and not on local characteristics of the data. In other words, only the age of the data determines its interest. An important application where time-dependent layouts are useful is network monitoring. The data consists of series of regularly sampled performance measures regarding utilization of network communication, process workload, etc. The network monitoring expert's task is to survey the performance of the network, to ensure that it meets predefined requirements, and to resolve problems as they occur. The main factor determining the interest of the network metrics can be assumed to be its age. The less current an observation is, the less important it is for the monitoring task. Conversely, the newest data can be assumed to be of highest interest. We note that, apart from age, of course characteristics of the observations will determine the interest of data portions, at least to some degree. In general, the interest can be modeled as a combination of age and data characteristics.

In the time-dependent case, we need to specify the degree of interest as a function of data age. Then, we can generate corresponding multi-resolution layouts where the level of resolution for each data partition to be visualized is depending on the corresponding data age. The level of resolution of a given data interval consists of a visualization method and a subset of display space allocated. Considering the visualization method, any technique for time-related data, e.g., line-charts, pixel-based displays, and glyph representations are possible.

We go on to formalize the concept of a discrete time-series degree of interest profile as a set of triples called multi resolution indices, *MRI*. An MRI triple indicates the amount of data to show, the amount of display space to allocate, and a rendering method:

$$\mathrm{MRI}_i = (\mathrm{D}_i, \mathrm{R}_i, \mathrm{V}_i)$$

Data range D_i specifies the data interval shown; display space R_i specifies the fraction of display space that is allocated to D_i. Rendering method V_i specifies the method for drawing D_i within R_i. By specifying appropriate profiles and drawing methods, we accommodate many use cases.

4.1. Matrix-Based Drawing Method

The color-coded matrix technique [HDKS07] is a generalization of the pixel-oriented visualization given in [KKA95]. It displays each data value by a rectangular amount of screen space, where the layout of rectangles can be configured in a number of different ways. The rectangles are color-coded to represent the magnitude of the value to be shown. The color is obtained from an appropriate function mapping the values *x* of the given data range to a color value:

$$colormap :: x \to color$$

Figure 2 illustrates the color map used in this paper. The cells are found by partitioning the input display rectangle into rows and columns forming a grid of cells. The number of rows and columns is found such that at most one column is only partially occupied with values, and that the cell shapes approach squares as closely as possible. The cells are filled in the order of the time series in a column-by-column manner from left to right. The display is efficient in visualizing large amounts of data. Compared to bar or line charts, color-coded matrix charts require much less minimum display space for lossless time-series display. Based on the amount of data and available screen space the matrix display scales down rectangle sizes, eventually utilizing the pixel level. Therefore, the matrix display is a compact and efficient approach for visualizing large time series. We note that the pixel-level is the ultimate limit for efficient lossless time series drawing. Accommodating even more data in a space-filling way, we need to apply numeric data aggregation methods. In our implementation, we resort to data sampling for drawing data partitions which sizes exceed the number of pixels in the allocated hosting display space.

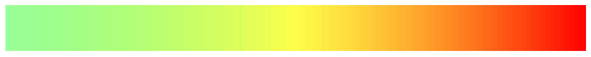

Figure 2: Example color scale.

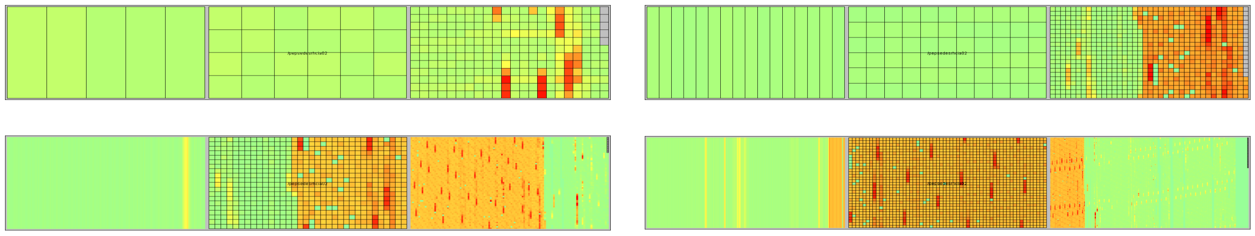

Figure 3: CPU utilization values for *288*, *864*, *7.776*, and *25.920* values (top-left to bottom-right) taken from one target host on a network. The color map assigns shades of green (low) to red (high) CPU utilization measurements (cf. the color map illustrated in Figure 2). The display allows visually analyzing large time intervals, keeping the most recent data (leftmost partition in each image) at highest resolution. The cell sizes for the older data are decreased by increasing data size. Considering the full time frame in the last image, we learn that the server is currently idle, but before that it experienced medium CPU utilization for quite some time (orange-colored cells). The medium utilization phase was characterized by periods of high peak utilization (the red "hot spots"). The utilization peaks occurred in regular intervals since the hot spots are evenly distributed.

4.2. Application of the Time-Dependent MR Layout

We now apply the time-dependent multi-resolution technique on a large real-world data set from the network monitoring domain. We define a three-fold multiresolution profile which has been found useful in this domain. It is defined using weights for data and display as:

$$\begin{aligned} MRI &= \{ MRI_1 , MRI_2 , MRI_3 \} \\ &= \{ (1, 1, c\text{-}matrix), \\ &\quad (4, 1, c\text{-}matrix), \\ &\quad (16, 1, c\text{-}matrix) \} \end{aligned}$$

This means that the whole time series is divided into three intervals. Each interval is given the same amount of display space, as $R_i = 1$ for all MRI. The partitions contain increasingly more data: $D_1 = 1$, $D_2 = 4$, $D_3 = 16$ (a factor of *4* is used for expanding the data intervals). The display emphasizes perception of the most recent data interval D_1 at high resolution (data-to-display relation is *1:1*), while maintaining the intermediate (D_2) and old data (D_3) in context (data-to-display relations are *4:1* and *16:1*, respectively). By putting increasingly more data into the same amount of display space, data density increases. Perception of data density is visually supported by drawing borders around each matrix cell in the display. For the first MRI matrix, we arrange the color-coded matrix cells in a fixed one-row layout. For the other data partitions, we find the cell grid dimensionality to approximate squares while minimizing empty cells. We have found using one-row matrices for the first data partition to be the best policy to underscore the interestingness of the first (and most recent) partition, while supporting discrimination of the time-interval durations in the consecutive data partitions.

Figure 3 illustrates the color-coded matrix display for an increasingly longer time series of a server's CPU utilization metric, probed in 5 minute intervals. The full time series compounds observation data from 3 months, resulting in about 25,000 probes. The full data set is shown in the last row in Figure 3. We note that this number of observations could not be plotted properly and space-filling in a standard line or bar chart, and on a current workstation display device.

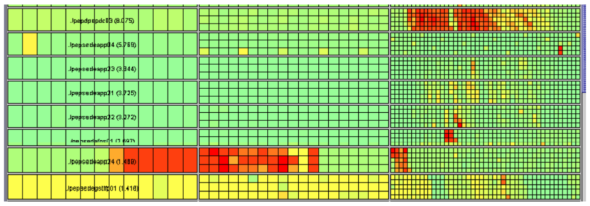

Figure 4: Comparison of multiple time series along three resolution levels. The chart shows the continuous CPU utilization history (*3* days = *864* values) of *8* different computing hosts. The color map assigns shades of green (low) to red (high) CPU utilization. Overall, there is little correlation between the servers' utilization profiles, leading us to conclude that these operate independently of each other (they are not pooled in a load-balancing configuration). The overall utilization level is rather low (green), except for one high utilization period of medium duration for server pepsedeapp24 (2nd from below) and one high utilization period of long duration for server pepdpepdc03 (1st from above).

Another application is given in Figure 4, illustrating the color-coded matrix display for CPU utilization metrics of multiple hosts simultaneously. By plotting them in a tabular-like layout below each other, one is able to visually correlate the performance of the hosts along the present, intermediate, and long-term scales.

4.3. Animating the Multi-Resolution Displays

The matrix technique can also support the monitoring of data streams in real time. We propose two modes of operation for the technique. One is to fix the grid sizes and allow a constant number of most recent values in the display at any time (incoming data flushes oldest data). The other mode is to allow any number of data values to accumulate in the display by distributing the set of values according to the multi-resolution layout definition, as the data streams in. One nice aspect of the matrix technique is that it elegantly allows encoding the interestingness of any amount of data by simply setting the data density, which in turn determines the matrix cell size (resolution). If we reduce the given display space, the same number of observations will be drawn with smaller cell sizes, reflecting less interestingness of the data.

We note that an open problem of animating the display in the sketched ways is that of positional changes of values occurring (positional discontinuities). Studying these effects more carefully, and developing improved visualizations which diminish these effects, are considered interesting future work.

5. Data-Dependent Multi-Resolution Layout Generation

In Section 4 we discussed the usage of fixed multi-resolution layouts appropriate when a stable DOI function over time exists. On the other hand, there exist scenarios where the user has no a-priori knowledge regarding the data, but expects the visualization to structure (guide) the data exploration and analysis process. In this paper, we propose to have an algorithm automatically analyze the data, and derive an appropriate data-dependent DOI function based on the outcome of the analysis. The design space for building such functions is large. We next give a basic binning-based algorithm that calculates a DOI function for time-series. The algorithm can accommodate any time series analysis function yielding a scalar interest measure defined over the time series, as required by the application.

Like pointed out earlier, we assume that the layouts should consist of only a limited number of different resolution levels in order to be effective. The aim of our data-dependent multiresolution display is to enable the user to quickly identify (focus on) the most interesting portions of the data, while keeping the complete time series in perspective. In particular, the display has to effectively communicate the following:

1. The interest levels of each portion with respect to the whole data set;
2. The interval covered by each partition, in terms of number of observations (duration on the time scale); and
3. The timestamps of start and end endpoints of the identified data interval.

We support these goals by abstracting the data set into a limited number of data partitions which are shown at discretized resolution levels corresponding to their degree of interest, as determined by the algorithm generating the DOI function.

5.1. Calculating the DOI Function

The basic idea of our DOI profile generator is to partition the time series into a number *N* of equal-width data bins, and determine the magnitude of a suitably defined real valued measure for the interest level for every bin. Depending on the application at hand, this measure might be anything from a simple averaging function to an advanced unsupervised time-series analysis algorithm like given in [WKL*05]. Optionally, we then reduce the number of bins by merging each pair of adjacent bins if the respective interestingness scores have similar magnitudes (as determined by a threshold parameter *t*; cf. Section 5.3.2). The result is a piecewise constant DOI function with cardinality less than or equal to *N*, representing the degree of interest over the time series, as implemented by the time series analysis function. Figure 5 illustrates the DOI calculation process.

5.2. Generating the Multi-Resolution Layout

The second step is to transfer the calculated DOI function to the multi-resolution display. We first note that the DOI profile we obtained already defines the display layout up to the resolution level: The piecewise constant DOI function breaks up (segments) the time series into a number of partitions each associated with a constant degree of interest. Naturally, these segments should be displayed in the same visual unit and at the same resolution level each. We can determine the resolution level for each partition by considering the area under the DOI function over the considered data partition in relation to the total area under the DOI curve. The resolution level RES_i for a data segment S_i in the interval $[i^0, i^n]$ of a time series can then be expressed as:

$$RES_i = \left(\frac{\int_{i^0}^{i^n} DOI(t)\,dt}{\int_{T_0}^{T_N} DOI(t)\,dt} \right)^{1/s},$$

where $DOI(t)$ is the degree of interest function defined for each time index (time stamp) *t* between $[T_0, T_N]$ of the time-series domain, and *s* is a non-zero scaling factor. For $s>1$ ($s<1$), smaller resolution levels are emphasized (deemphasized).

As a result of calculating a discrete DOI function and transferring it to a corresponding multi-resolution layout, all we have to do is find the appropriate rendering methods parameterized by resolution level. One method is to predefine a number of rendering methods, order them by resolution level, and map the numeric resolution scales calculated for the data portions to the set of predefined rendering methods. Here we choose another way that relies directly on the matrix rendering method described in Section 4.1 by interpreting the resolution level RES_i as a weight for allocating display space to data intervals. Recall that the matrix drawing method operates on a data range and a partition of display space. By increasing (decreasing) the amount of display space allocated to a given data partition, we increase (decrease) the amount of space available per value, thereby increasing (decreasing) the resolution of the display. As we recall from the application of the matrix technique, the grid resolution is reflected in the size of the cells. Thereby, the level of interest is readily perceivable when using the matrix technique in conjunction with automatic DOI generation. Figure 6 illustrates the transfer of a DOI profile to the matrix drawing approach.

To summarize, the data-dependent multi-resolution layout automatically assigns display space to data partitions guided by the DOI clustered and merged by similarity of interest scores.

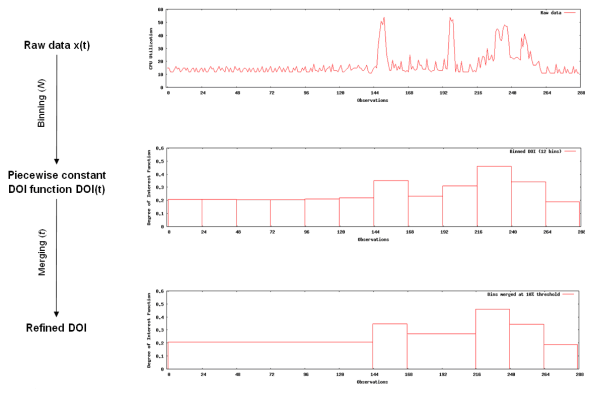

Figure 5: DOI extraction from raw time series by determining the interestingness score over the time series for a number of bins *N*, followed by (optional) merging of bins with similar interestingness scores.

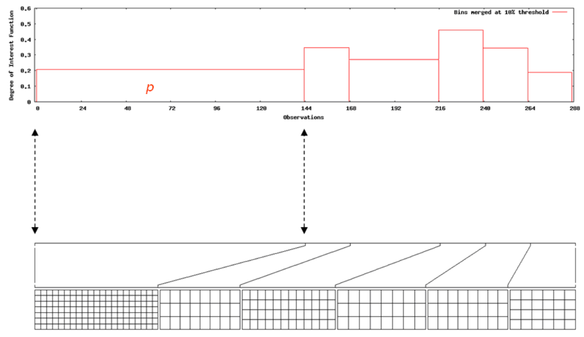

Figure 6: Mapping the generated DOI profile to a corresponding multiresolution layout using the color-coded matrix display visualization. The DOI profile determines the data partitioning, while the transfer function maps the data intervals to screen partitions. Implicitly, this also reflects the level of resolution of the partitions via the matrix dimensionality (cell sizes).

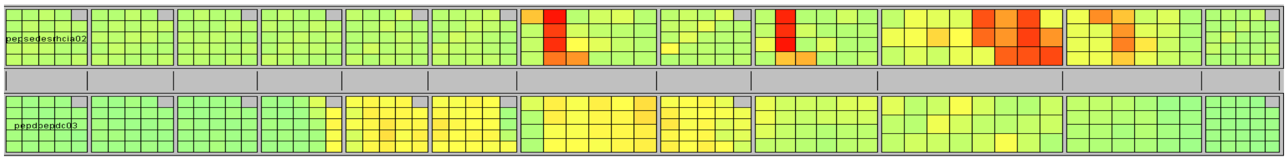

Figure 7: Two series of CPU utilization measures in a fixed time-slice multi-resolution layout (FTS). The time slice was set to 2 hours. The DOI profile was generated using the *AVERAGE* aggregation function analyzing both time series.

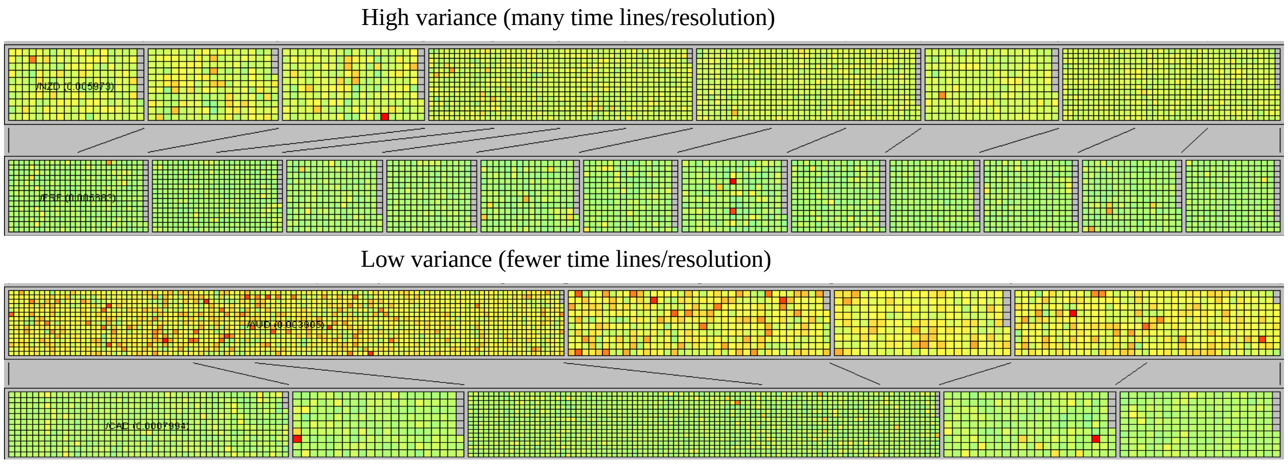

Figure 8: Showing four variable time-series results for exchange rate returns out of 12 International currencies measured against US-$ using the MAXIMUM aggregation function and independent DOI analysis for each of the time series.

5.3. Instantiations

We discuss two instantiations for data-dependent multi-resolution layout generation. The first fixes the basis time interval for the automatic resolution determination. The other instantiation also determines the interval size automatically.

5.3.1. Fixed Time-Slice Displays (*FTS*)

In this mode, we fix am atomic time slice which is used throughout the multi-resolution layout generation. This approach is motivated by the fact that in many applications, there is an atomic base time interval with which the users are familiar and which the analysts use to base their rankings on. E.g., in Finance, often developments are considered on a daily basis. The fixed time slice display is generated by fixing the number of bins for which the DOI is calculated, such that each bin represents the number of values corresponding to the fixed time unit. Let time series TS contain $|TS|$ values, and the atomic unit contain u values. Then the number of bins is set to $b = |TS| / u$. The fixed time slice instantiation results in multi-resolution layouts where each interval (data chunk) corresponds to the constant, atomic time interval predefined, and interestingness of the chunks is reflected by respective data density as allocated by the transfer function. Note that FTS is effectively produced by omitting the bin-merging step illustrated in Figure 5 (bottom).

5.3.2. Variable Time Slice Displays (*VTS*)

In this mode we make the determination of data partition sizes dependent on the interestingness profile by merging partitions of only small interest differences, and showing them in the same visual unit. VTS results in MR layouts where each interval (data chunk) length is dependent on the outcome of the merging step (cf. Figure 5, bottom) in the analysis algorithm based on the scale of the interesting function and the merging threshold parameter t. The length of a time slice is an integer multiple of the length of the initial bin partitioning according to parameter N. VTS is able to more compactly represent the interestingness profile by being allowed to merge individual bins, possibly forming less isolated data intervals in the display. On the other hand, VTS is somewhat more difficult to visually process, as the distribution of individual time-series partition lengths is potentially heterogeneous. To compensate for this potential drawback, we next introduce special charting elements aimed to support the perception of the different individual time-slice lengths.

5.4. Multi Resolution Charting Elements

We propose including special charting elements for multi-resolution displays. The multi-resolution legend is placed on top of a time-series chart layout. It textually gives the time range as well as the number of values covered by each partition (see bottom of Fig. 6). It contains a visual representation of the partition layout (grid or bar chart structure). Time lines then map from the reference linear time scale (top edge in the diagram) to the multi-resolution time scale (middle edge in the image) as defined by the partitions, providing explicit visual clues on the levels of resolution applied. Inter-series time-lines map from partition boundaries in one series to the respective position in another series, providing time alignment information for displays consisting of multiple multi-resolution time series. These connecting time-lines are specifically useful for array of time series where for each time series an individual layout is generated.

5.5. Application of the Data-Dependent MR Layout

In Section 4.2, we applied the CPU utilization data in time-dependent multi-resolution layouts (cf. Figures 3 and 4). In this section, we apply the same CPU utilization data to demonstrate the FTS in Figure 7. We assume we are interested in identifying bottlenecks in the performance data. Opposed to the application in the last section (online monitoring), here we are concerned with analysis of historic (non-online) data. When generating layouts for multiple time series, basically there are two modes of operation. The first mode refers to obtaining the layout automatically from one specific time series in the set of time series, and then repeatedly applying this layout on all time series. This choice is most appropriate when there is one root (master) time series in the data set, which in some way exercises influence on the other time series, and for which we want to search for correlations. E.g., in network monitoring, we can cumulate several performance measures like response time, disk utilization, availability and so on into one master metric like service level compliance. Then, this master metric can be used to derive an aligned overall multi-resolution. The second mode of operation is to generate individual layouts for each time series in the set, and use the inter-series time-lines charting element.

Figure 7 shows the application of the FTS multi-resolution layout of a data set of two time series representing CPU utilization probes during one day. We used the *maximum* function on a number of bins to derive the DOI function. The top chart shows the FTS result, when using the first time series as the master for deriving the interestingness profile. The time slice was fixed to contain 2 hours. The first series gives the multi-resolution layout which is adopted for the second time-series. We quickly recognize two time slices with high maximum CPU utilization: These slices are allocated significantly more display space by the layout generator, while the intervals with low utilization get less display space. Again we can perform detailed visual correlation analysis to see if the utilization profile of the first server corresponds with utilization patterns of the remaining servers.

Figure 8 shows the independent variable time slice analysis (VTS) approach. The image shows the result of the variable time slice layout generator applied on another data set: Time series of return rates of International currencies measured against the US-$ for roughly *10* years, corresponding to *2566* return rates per currency [Duk]. We set the profile generator to consider the *maximum* aggregation function for generating variable time slice layouts independently for each of the series, using time lines as visual clues to aligning the time periods. The DOI analyzer isolates time subintervals with sufficiently different maximum spot rate returns for the different currencies. We recognize that compared to the fixed (FTS) layout in Figure 7, it is somewhat harder for the user to align the different time period subintervals, because we now have different time slice sizes. Nevertheless, the time lines allow getting a rough alignment, and it is easy to support time-based alignment perception using standard interaction techniques. The main use-case and advantage of variable-time slice and independent analysis is that it allows assumption-free and comparative visual analysis of the characteristics present in many and long time-series data. The tessellation into the different MR levels indicates the variability of the interestingness metric within and among the different time series. E.g., the Canadian $, as given in the bottom row of the low variance time-series in Figure 8, has the most homogenous return rates with little variance, and it is segmented into only 5 different subintervals by the DOI analyzer. The other currencies in the example show more dynamics in the return rates, as is obvious from the many different time slices in the top high variance time-series.

6. Comparison to Line Charts

As discussed previously, over plotting and information loss may easily result when drawing large time-series using

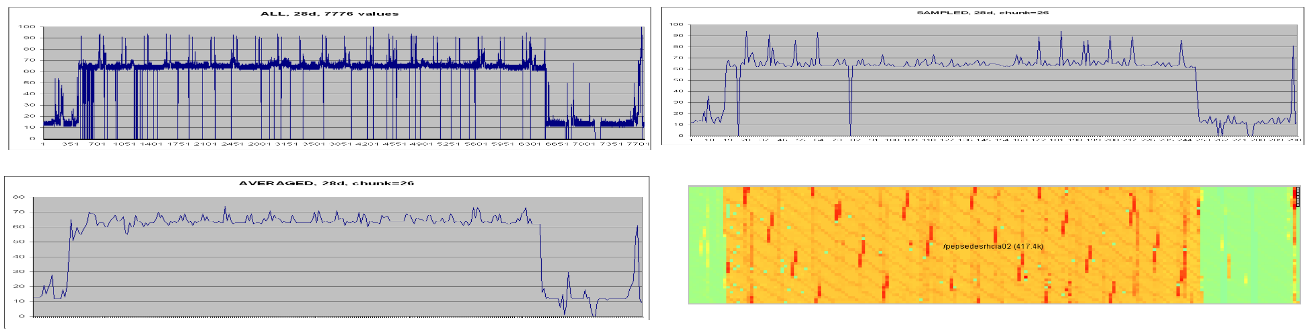

Figure 9: Visualization of *28* days worth of CPU utilization probes from one server resulting in *7776* observation values. From top-left to bottom-right is given (1) the raw data showing overplotting artefacts, (2) sampled data, (3) averaged data (both loosing detail), and (4) the color-coded matrix display representing the whole information. Users can focus on an interesting area and zoom to a detailed line chart.

standard line-chart displays. In Figure 9, we show a time series with *7776* values. Using no data reduction, we observe severe overplotting effects (top-left). We cannot see detailed developments anymore, but rather just the bandwidth in terms of minimum and maximum values for many time intervals. When using *data reduction by equi-spaced sampling* (top-right), we may lose information, and based on the sampling intervals, aliasing artifacts may show characteristics which are not present in the original data. *Averaging* (left-bottom) introduces a smoothing effect which shows a somewhat more deterministic representation than sampling does, but extreme values are also smoothed, resulting in loose information which is important in many applications, like network monitoring where the peaks (extreme) are especially important. Given is also the color-coded matrix display in Figure 9 (bottom-right; a single data partition profile was chosen). The display is able to visualize the full data set without numeric data reduction, thereby not incurring any information loss. The overall time series pattern as well as local characteristics is well perceivable. The multi-resolution technique complements line charts without losing information. Users can zoom in on any interesting areas to detailed line charts for further analysis.

7. Conclusions

In this paper, we have introduced a family of multi-resolution techniques for visualizing long time-series data using non-linear rescaling in conjunction with a space-efficient rendering method. The rescaling was performed by generating either time-dependent or data-dependent DOI profiles. For the analysis, any appropriate time-series analysis algorithm which produces a numeric interest scale can be employed. We presented applications of the technique in the network monitoring and finance domains, and concluded its effectiveness for visual analysis of long time-series data. Resorting to multi resolution visualization is a useful approach for (a) coping with increasingly larger data sets occurring in many application domains, and (b) guiding the user in distinguishing more and less important data types.

Future work involves visualizing stream data and researching additional drawing methods that continuously scale with the obtained DOI functions. We will integrate more complex time-series analysis algorithms and extend them to further application domains. The usage of iconographic representations of patterns in time-series databases is another promising approach we will explore for scaling with very large data. Finding appropriate iconographic drawing methods that intuitively represent predefined patterns in time series data along with their interestingness at a given resolution level is expected to lead to semantically rich visualizations.

References

[Chu02]. M. Chuah. *Demystifying Venture Capital Investing*, Proc. IEEE Symposium on Information Visualization, p. 161-164, 2002.

[Chu98] M. Chuah. *Dynamic Aggregation With Circular Visual Designs*, Proc. IEEE Symposium on Information Visualization, p. 35-43, 1998.

[CK98] J. Carlisa and J. Konstan. *Interactive Visualization of Serial Periodic Data*, Proc. ACM Symposium on User Interface Software and Technology, p. 29-38, 1998.

[Duk] *Spot Exchange Rate Dataset,* Institute of Statistics and Decision Sciences, Duke University.

[Fur86] G. Furnas. *Generalized Fisheye Views,* Proc. SIGCHI Human Factors in Computing Systems, p. 16-23, 1986.

[HDKS05] M. Hao, D. Keim, U. Dayal, T. Schreck: *Importance-Driven Visualization Layouts for Large Time Series Data,* Proc. IEEE Symposium on Information Visualization, p. 203-210, 2005.

[HDKS06] M. Hao, U. Dayal, D. Keim, T. Schreck: *Multi-Resolution Techniques for Visual Exploration of Large Time-Series Data*, Poster at IEEE Symposium on Information Visualization, 2006.

[HDKS07] M. Hao, U. Dayal, D. Keim, T. Schreck: *A Visual Analysis of Multi-Attribute Data Using Pixel Matrix Displays*, Proc. IS&T/SPIE Conference on Visualization and Data Analysis, 2007.

[KKA95] D. A. Keim, H.P. Kriegel, and M. Ankerst. *Recursive pattern: A Technique for Visualizing Very Large Amounts of Data,* Proc. IEEE Visualization, p. 279-286, 1995.

[MS03] W. Mueller, H. Schumann: *Visualization Methods for Time- dependent Data - an Overview*, Proc. of Winter Simulation Conference, 2003.

[Mun04]. T. Munzner. *BinX: Dynamic Exploration of Time Series Datasets Across Aggregation Levels*, Proc. IEEE Symposium on Information Visualization, p. 215.2, 2004.

[RC95] R. Rao and S. K. Card. *Exploring Large Tables With the Table Lens*, Proceedings of ACM Conference on Human Factors in Computing Systems (CHI'95), p. 403-404, 1995.

[SK00] S. Eick, A. Karr. *Visual Scalability*, IEEE Transactions on Visualization and Computer Graphics, Vol. 6, Nr. 1, p. 44-58, 2000.

[SSJ05] A. Aris, B. Shneiderman, C. Plaisant, G. Shmueli, W. Jank. *Representing Unevenly-Spaced Time Series Data for Visualization and Interactive Exploration,* Proc. International Conference on Human-Computer Interaction, p. 835-846, 2005.

[WAM01] M. Weber, M. Alexa, W. Mueller. *Visualizing Time-Series on Spirals*, Proc. IEEE Symposium on Information Visualization, p. 7-14, 2001.

[WKL*05] L. Wei, N. Kumar, V. Lolla, E. Keogh, S. Lonardi, C. Ratanamahatana. *Assumption-Free Anomaly Detection in Time Series*, Proc. International Conference on Scientific and Statistical Database Management, p. 237-240, 2005.