

SIGNALPLAYER - A TIME SERIES VISUALIZATION SYSTEM

Pekka Siirtola, Perttu Laurinen, Juha Rönning

Intelligent Systems Group, Computer Engineering Laboratory
University of Oulu
Finland
email: pesiirto, perttu, jjr@ee.oulu.fi

ABSTRACT

Graphical presentation of long time series signals is challenging because the resolution of a computer screen in pixels is usually smaller than the length of the signal in points. However, visualizing signals at a glance is highly important because analysing data in graphical form can be much more productive than it is in numerical or written form. Visualization of complete signals is possible only when using presentative approximations of the signals. For this purpose SignalPlayer, a time series visualization system, is developed and presented in this paper. SignalPlayer is a flexible tool for the basic visualization of time series. It enables the display of the complete signal at a glance and gives the user the option to choose or implement the visualization technique for the presentation. The tool can also be used to examine the signals in detail.

1. INTRODUCTION

Technology of the 21st century enables the collection and handling of high frequency and long time series signals using sensors and computers, for example. The analysis and visualization of these signals are both challenging research topics. This paper concentrates on presenting solutions for presenting long time series on a computer screen at a glance. The graphical presentation of long time series is challenging because the resolution of a computer screen is not good enough to represent every point of the signal on the screen at once when the number of points of the signal is greater than the width of the screen in pixels. See for example Figure 1, where a signal of 60,000 points is drawn on a display of 364 points in x direction. Several methods been proposed for solving this problem.

Unvin has introduced different statistical graphic displays for large data sets [1]. Histograms are found to be an inefficient tool when data sets are massive because the number of bins can become so large that it is not possible to show all the bins on the screen. The number of bins needed can be calculated using the formulas presented in [2]. Histograms also suffer from the fact that they lose the order of

the points. Dot plots are also represented in [1]. They are considered practical when data sets include less than 100 cases. When the number of cases is larger it is impossible to identify individuals because there are rarely any gaps.

Many applications have been produced to visualize time series data. One of these is TimeSearcher [3]. TimeSearcher is a visualization tool for exploring time series data sets, see Figure 2. TimeSearcher suffers from the same problem as many other applications and is not useful when the data sets are large. It tries to represent every point of the signal on the screen at once, so it is impossible to read time series when it contains more points than the screen can separate. On the other hand, TimeSearcher has other useful features. For example it can be used to search similar patterns. The finding of patterns is not completely automated because the user needs to specify query regions in order to find similar patterns. TimeSearcher 2 is presented in [4]. It has the same problems as TimeSearcher but allows the exploration of multidimensional data and other more advanced features than TimeSearcher.

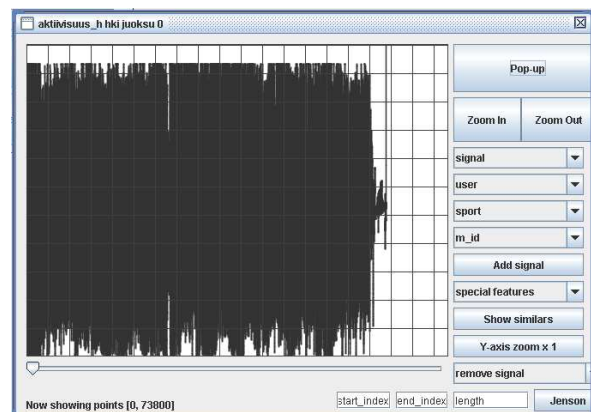


Fig. 1. Visualization of large data sets is problematic. Every point of the data set is drawn on the screen using the default approximation of the tool. The number of observations is 60,000 and the screen resolution is 364 pixels in x direction.

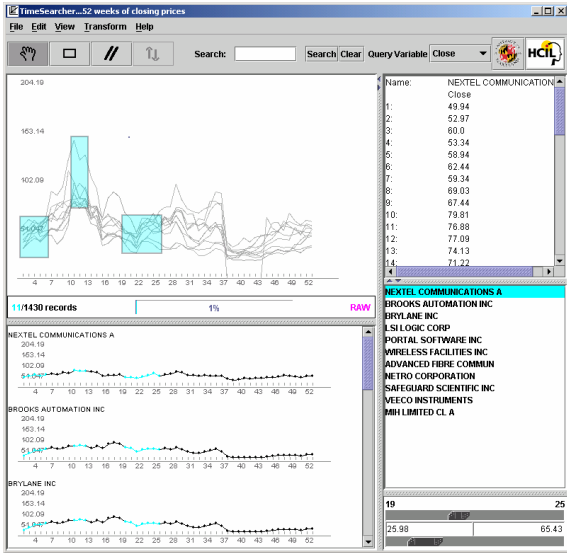


Fig. 2. A screenshot of TimeSearcher. Similar patterns can be found using query boxes.

Other visualization tools are also available; one of these is VizTree [5]. VizTree is specially designed to aid Aerospace analysts, but can be used for other purposes as well. It can handle massive data sets by using a zooming option, but if the used time series is long enough and the user wants to see the whole signal at once, it suffers from the same problem as TimeSearcher and TimeSearcher 2. The article does not reveal what kind of approximation method VizTree uses when long data sets are viewed, but the user is not able to choose this method. It has the option of viewing several data sets at the same time. VizTree has very advanced tools for finding similar patterns and for that purpose it is one of the most effective applications available. The method that is used to find similar patterns is presented in [6]. A screenshot of VizTree can be found on Figure 3.

A cluster and calendar-based time series system is presented in [7]. The system cuts the time series into sequences of daily data patterns and then the system clusters similar day patterns and visualizes the average patterns as graphs and corresponding days on the calendar. This approach works well if the data is calendar-based but in other cases the system is not very useful.

2. DESCRIPTION OF THE FRAMEWORK

The main problem with visualizing massive data sets is that only limited amount of data fits on the screen at once as was discussed in the introduction and demonstrated in Figure 1. If the number of points in the screen is marked with n_1 and the number of points in the signal with n_2 , it is usually the case that $n_1 \ll n_2$. If the points n_2 are to be presented in a

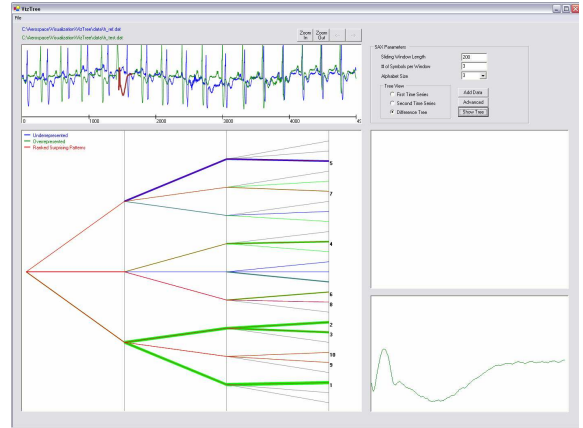


Fig. 3. A screenshot of VizTree. VizTree offers versatile methods for exploring time series data. Similar patterns can be found using tree panel.

resolution n_1 , a suitable mapping has to be defined. For this purpose, SignalPlayer, a time series visualization system, is developed and represented in this paper.

SignalPlayer fetches time series signals selected by the user from a database and presents them in a suitable graphical form, see Figure 5. The main function of the system is to represent data sets in graphical form in such a way that the visualized signal can be infinite long. The visualization is built so that the user can select the visualization method, which is then applied to the original time series, after which the signal is represented on the screen. The visualization method divides the data into equal-sized segments of n points. Each segment is then replaced by one single feature x that is calculated from the points of the segment. These features x are then displayed on the screen. Features are calculated using the function

$$f : D_{n,i} \mapsto x_i, \quad (1)$$

where $D_{n,i}$ is the data set that contains all the n points from the i th segment and x_i is the feature presenting the segment. The user is free to choose the type of feature used. This method offers an opportunity to explore the whole time series at once even in cases where the data set is massive.

SignalPlayer has also a zooming option which allows the user to view some parts of the signal in more detail (see Figure 4). The user can decide what subsignals he wants to explore. When zooming is used, SignalPlayer changes the width of the segment n using the formula

$$n_{new} = \frac{n_{old}}{z}, \quad (2)$$

where z is the change in zoom rate in percents and can be calculated by

$$z = \frac{z_{old}}{z_{new}}, \quad (3)$$

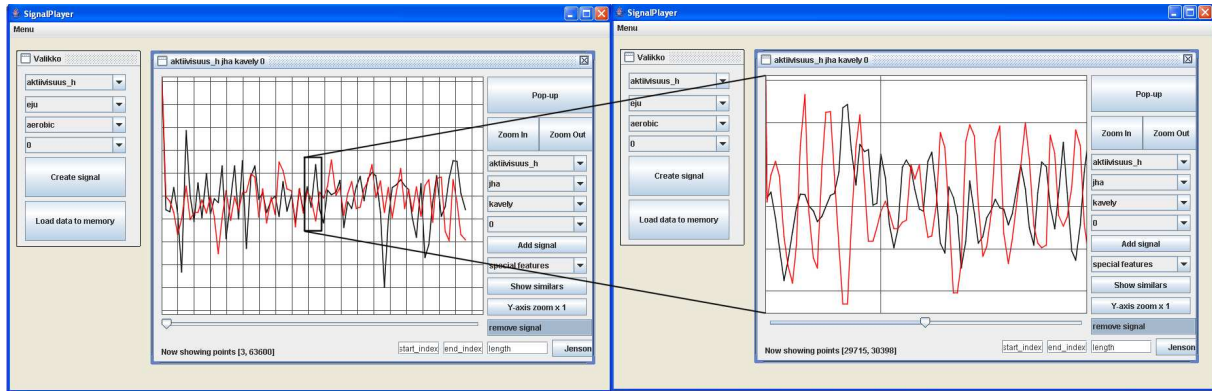


Fig. 4. First look at SignalPlayer. The picture on the left shows two signals with lengths of over 60,000 points and the right one demonstrates zooming. The right picture represents subsignals of from the interval [29713, 30398]. Notice that there can be several signals displayed in the same window.

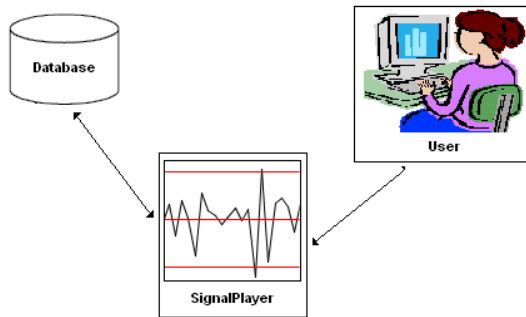


Fig. 5. How SignalPlayer works.

where z_{old} is the old zoom rate and z_{new} is the new zoom rate. This means that SignalPlayer gives compressed presentations of the original signal when zooming as well. When the level of zooming reaches the number of points that fit the screen, the original data is displayed.

SignalPlayer can display several signals at once in the same window and several windows can be open at the same time. When there is more than one signal drawn on the same window, signals are separated from each other using different colors (see Figure 4).

3. VISUALIZING TIME-SERIES USING THE FRAMEWORK

The user can decide the way SignalPlayer compresses the presented information, which makes it a flexible tool. At the moment the tool supports four ways for the approximation. The approximative feature can be one of the following:

1. One point. Only one point presents the segment and

this point can be for example the first or the last point of the segment, see Figure 6.

2. Mode of the points in the segment. The descriptive feature of the segment is the value that has the largest number of observations, see Figure 8.

3. The level with the most observations. The y-axis of the segment can be divided into equal-sized levels and then the value of the level that has the largest number of observations is considered to be the feature, see Figure 7. If the levels are only one pixel in height, then this feature and the one presented earlier give the same result.

4. The average value of the points. See Figure 9.

All of these methods result in a different visualization. Each of them is usable, but also have their shortcomings. The problem with the first method is that it does not actually tell much about the nature of the signal. It only draws every n th point but does not give any information about those $n-1$ points that lie between the two points that are drawn on the screen.

The mode-based method suffers from some problems too. Because there may be several y -values that have the same amount of observations, it is possible that the result is not unique. Also, if the y -axis can get a wide range of values, this method can be slow. Notice that this method works only with integers.

The third method has the same problems as the second, except that the values do not have to be integers.

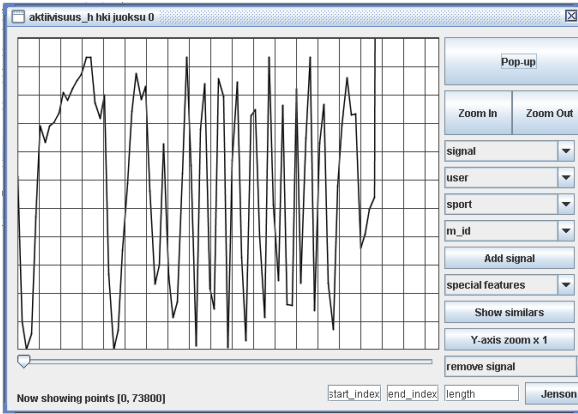


Fig. 6. Every n th point drawn to the screen.

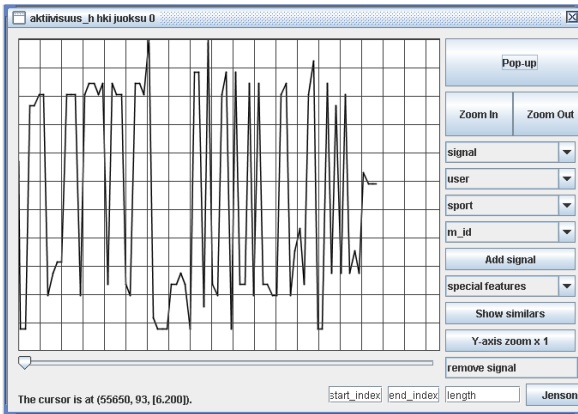


Fig. 7. Mode of the segment's points when the number of segments is 13.

When calculating the average value of the points in the segment, the problem is that the signal loses information about the extremums. Therefore the resulting signal can be flat. Still for the most cases this method is the most useful.

SignalPlayer also allows visualizing of every point of the signal at once, but as it was told this method does not work with large data sets, see Figure 1.

Figures 1, 6, 8, 7 and 9 are all drawn from the data set that contains about 60,000 observations.

4. CONCLUSIONS AND DISCUSSION

SignalPlayer offers a new flexible way for visualizing massive data sets. Data sets can contain more points than there are pixels on a computer screen because SignalPlayer uses summarizing techniques that provide methods for visualizing long time series. The user can decide which summarizing technique to use and new techniques can be added easily if necessary. One technique is not enough because different summarizing methods are appropriate for different

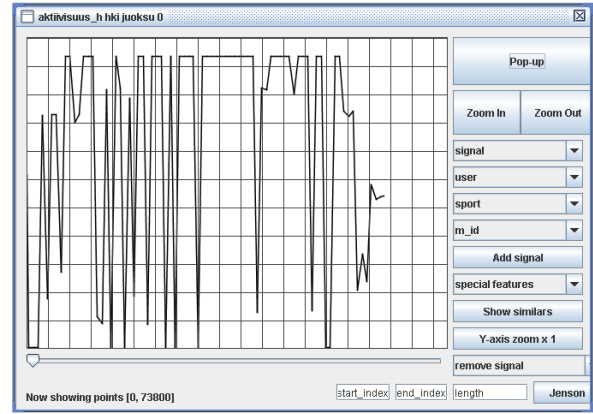


Fig. 8. Mode of the segment's points when every possible y -value has its own segment.

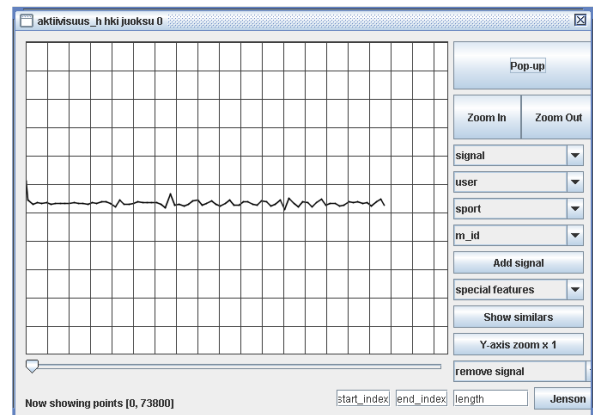


Fig. 9. Average value of the segment's points drawn on the screen.

situations and selecting the correct method can make signal analyzation much easier and faster. Summarizing methods that are available at the moment offer handy tools for the analyzation of signals but it would be important to create methods that would give a better idea about signal dynamics than current methods do.

In the future lot of new features will be included in SignalPlayer. One of these features is the possibility to search similar patterns from the signal.

5. REFERENCES

- [1] A. Unwin, "Visualisation for data mining," *International Conference on Data Mining, Visualization and Statistical System, Seoul 2000*, Dec. 2000.
- [2] D. W. Scott, *Multivariate Density Estimation: Theory, Practice and Visualization*, Wiley, New York, USA, 1992, 376 p.

- [3] H. Hochheiser, *Interactive graphical querying of time series and linear sequence data sets*, Ph.D. thesis, 2003, Director-Ben Shneiderman, 313 p.
- [4] P. Buono, A. Aris, C. Plaisant, A. Khella, and B. Shneiderman, "Interactive pattern search in time series," *Proceedings of Conference on Visualization and Data Analysis, VDA 2005*, vol. 5669, pp. 175–186, Jan. 2005.
- [5] J. Lin, E. Keogh, S. Lonardi, J. P. Lankford, and D. M. Nystrom, "Visually mining and monitoring massive time series," in *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, New York, NY, USA, 2004, pp. 460–469, ACM Press.
- [6] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *DMKD '03: Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, New York, NY, USA, 2003, pp. 2–11, ACM Press.
- [7] J. J. Van Wijk and E. R. Van Selow, "Cluster and calendar based visualization of time series data," in *INFOVIS '99: Proceedings of the 1999 IEEE Symposium on Information Visualization*, Washington, DC, USA, 1999, pp. 4–9, IEEE Computer Society.