Published in: Proc. Visualization '95, Atlanta, GA, 1995.

Recursive Pattern: A Technique for Visualizing Very Large Amounts of Data

Daniel A. Keim, Hans-Peter Kriegel, Mihael Ankerst
Institute for Computer Science, University of Munich
Leopoldstr. 11B, D-80802 Munich
{keim, kriegel, ankerst}@informatik.uni-muenchen.de

Abstract

An important goal of visualization technology is to support the exploration and analysis of very large amounts of data. In this paper, we propose a new visualization technique called 'recursive pattern' which has been developed for visualizing large amounts of multidimensional data. The technique is based on a generic recursive scheme which generalizes a wide range of pixel-oriented arrangements for displaying large data sets. By instantiating the technique with adequate data- and application-dependent parameters, the user may largely influence the structure of the resulting visualizations. Since the technique uses one pixel for presenting each data value, the amount of data which can be displayed is only limited by the resolution of current display technology and by the limitations of human perceptibility. Beside describing the basic idea of the 'recursive pattern' technique, we provide several examples of useful parameter settings for the various recursion levels. We further show that our 'recursive pattern' technique is particularly advantageous for the large class of data sets which have a natural order according to one dimension (e.g. time series data). We demonstrate the usefulness of our technique by using a stock market application.

Keywords: *Visualizing Large Data Sets, Visualizing Multidimensional and Multivariate Data, Visualizing Large Sequential Data Sets, Recursive Visualization Techniques, Interfaces to Databases*

1. Introduction

Having the right information at the right moment is crucial for making the right decisions. Decisions based on accurate and reliable information may lead to great success, while decisions based on incomplete or incorrect information may have serious consequences. In stock trading, for example, having the right information at the right moment improves the chances for a high profit, whereas having some wrong information may mean losing everything.

One of the problems decision makers face today is the rapidly increasing amount of information that needs to be considered in decision making. As computers come to affect more and more aspects of modern society, one by-product is the growing amount of information that is captured in a computer-readable form. The automation of activities in all areas, including business, engineering, science, and government produces an ever-increasing stream of data. In making decisions, we are often simply overwhelmed by the quantity of information. As a consequence, every day millions of people make decisions without being able to analyze all the relevant information, which leads to decisions that are often wrong or at least suboptimal.

To increase the amount of information that is considered in making decisions, it is important to effectively use the power of the available computer hardware and software. However, even with the most advanced systems, finding the right piece of information in a very large database remains a difficult and time-consuming process. The process cannot be fully automated since it involves human intelligence and creativity which are (still) unmatchable by computers. Humans will therefore continue to play an important role in searching and analyzing the data. In dealing with very large amounts of data, however, humans need to be adequately supported by the computer. One important way of supporting the human in analyzing and exploring large amounts of data is to visualize the data. Visual representations of the data are especially useful for supporting a quick analysis of large amounts of multi-modal information, providing the possibility of focusing on minor effects while ignoring known regular features of the data.

Visualization of data which have some inherent two- or three-dimensional semantics has been done even before computers were used to create visualizations. In the well-known books [Tuf 83, Tuf 90], Edward R. Tufte provides many examples of visualization techniques that have been used for many years. Since computers are used to create visualizations, many novel visualization techniques have been developed by researchers working in the graphics field. Visualization of large amounts of arbitrary multidimensional data, however, is a fairly new research area. Early approaches include scatterplot matrices [And 72, Cle 93], Chernoff faces [Che 73], parallel coordinates [Ins 81], and others [And 57, Bri 79]. Researchers in the graphics/visualization area are currently extending these techniques to be useful for large data sets, as well as developing new techniques and testing them in different application domains. The techniques can be classified into geometric projection techniques (e.g., [Hub 85, ID 90]), iconic display techniques (e.g., [Che 73, Bed 90, SGB 91, SBM 93]), hierarchical techniques (e.g., [BF 90, LWW 90, MGTS 90, Shn 92]), dynamic techniques (e.g., [MZ 92, MTS 91, AWS 92, Eic 94]), and combinations hereof (e.g. [Asi 85, AS 94]). The research also resulted in various prototype systems for data exploration which implement some of the mentioned techniques. Examples include statistical data analysis packages such as Data Desk [The 95], XGobi [SCB 92], and Trellis [BC 95], and database oriented systems such as the Information Visualization and Exploration Environment (IVEE) [AW 95a, AW 95b].

In most of the approaches proposed so far, the number of data items that can be visualized on the screen at the same time is still quite limited (in the range of 100 to 5,000 data values). In our work, we focus on visualization techniques that allow a visualization of much larger amounts of data. The basic idea of our techniques is to map each data value to a colored pixel and present the data values belonging to each of the dimensions in separate windows. Since in general our techniques only use one pixel per data value, the techniques allow us to visualize the largest amounts of data which are possible on current displays (up to about 1,000,000 data values). If each data value is represented by one pixel, the main question is how the pixels are arranged on the screen. Our previous work focuses on supporting the data exploration and data analysis process by providing query-dependent visualizations of the data, presenting the most relevant data items in the center of the display [KK 94, Kei 94]. For a wide range of applications, however, different arrangements of the data seem to be more appropriate. Consider, e.g., the large class of data sets which have an a-priori ordering such as time series data. If the data is visualized in a query-independent way, there is no reason to present certain data items centered in the middle of the display. Instead, the natural ordering of the data may be used to arrange the data on the screen. In this paper, we propose a new visualization technique called 'recursive pattern' which supports a highly structured visualization of large data sets. The technique is based on a generic recursive scheme which allows user-defined parameter settings for the various recursion levels.

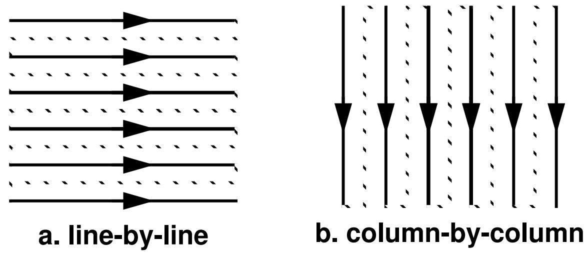

Figure 1: Simple Data Arrangements

The 'recursive pattern' visualization technique generalizes a wide range of pixel-oriented visualization techniques.

In the rest of this paper, we describe and evaluate our 'recursive pattern' visualization technique. Section 2 provides a detailed description of the technique including examples of useful parameter settings, which show the wide range of pixel-oriented visualizations that can be generated using our technique. We also describe how the system supports the user in determining adequate parameter settings for the various recursion levels. In section 3, we demonstrate the usefulness of the 'recursive pattern' visualization technique by using an application from the financial domain and compare our technique to standard techniques for visualizing financial data. Section 4 summarizes our approach and points out some of the open problems for future work.

2. The 'Recursive Pattern' Visualization Technique

As already mentioned, the basic idea of our visualization technique is to present as many data items as possible at the same time with the number of data items being only limited by the number of pixels of the display. In dealing with arbitrary multidimensional data without any 2D- or 3D-semantics, one major problem is to find meaningful arrangements of the pixels on the screen. Even if the data has a natural ordering according to one dimension (e.g., time series data), there are many possibilities for arranging the data. One straightforward possibility is to arrange the data items from left to right in a line-by-line fashion (c.f. Figure 1a). Another possibility is to arrange the pixels top-down in a column-by-column fashion (c.f. Figure 1b). If these arrangements are done pixelwise, in general, the resulting visualizations do not provide useful results. One possibility to improve the visualizations is to organize the pixels in small groups and arrange the groups to form some global pattern. This strategy corresponds to a two step approach with a first-order pattern formed by grouping the pixels and a second-order pattern formed by the global arrangement. By taking the result of the second order structure as the basic building element for a third level structure, we may intro-

```
void draw(x,y,level)
{if (level==0)
    Setpixel (x,y,color);
else      // level >= 1
    {for (int h=1; h<=height[level]; h++)
        {if (h%2)   // odd height ?
            {for (int w=1; w<=width[level]; w++)
                {draw(x,y,level-1);        // recursive call of the algorithm
                x+=next_x[level-1];
            }}
        else      // even height ?
            {for (int w=1; w<=width[level]; w++)
                {x-=next_x[level-1];
                draw(x,y,level-1);         // recursive call of the algorithm
            }}
        y+=next_y(level-1);
        }
    }
}
```

with: `next_x[level]` $= \prod_{i=1}^{level} w_i$ **and** `next_y[level]` $= \prod_{i=1}^{level} h_i$

Figure 2: 'Recursive Pattern' Algorithm

duce a third order pattern. This process may be iterated up to an arbitrary level, forming a general recursive scheme. In the simplest case, the patterns for all recursion levels are identical. In many cases, however, the data has some inherent structure which should be reflected by the pattern of the visualization. Consider for example time series data, measuring some parameters several times a day over a period of several years. It would be natural to group all data items belonging to one day in the first level pattern, those belonging to one week in the second level pattern, those belonging to one month in the third level pattern, and so on. This, however, means that the technique must be defined in a generic fashion, allowing user-provided parameters for defining the structure of the patterns for the recursion levels. This requirement is reflected by the generic definition of the 'recursive pattern' technique.

Note that our technique does not necessarily require the data to have some natural ordering. In searching for dependencies between dimensions, one might sort the data according to one dimension and use the 'recursive pattern' visualization technique for examining the dependencies to the other dimensions. Consider, for example, a large database of personnel data. If one wants to find dependencies between the parameter sales (of a person) and other dimensions such as salary, age, and travel expenses, one might sort the data according to the sales parameter and visually examine the dependencies of the other dimensions.

2.1 Description

The 'recursive pattern' visualization technique is based on a simple back and forth arrangement: First, a certain number of elements is arranged from left to right, then below backwards from right to left, then again forward from left to right, and so on. The same basic arrangement is done on all recursion levels with the only difference that the basic elements which are arranged on level i are the patterns resulting from level(i-1)-arrangements. Let w_i be the number of elements arranged in the left-right direction on recursion level i and h_i be the number of rows on recursion level i. Then, the pattern on recursion level i consists of $w_i \times h_i$ level(i-1)-patterns, and the maximum number of pixel that can be presented on recursion level k is given by $\prod_{i=1}^{k} w_i \times h_i$.

The recursive algorithm for generating 'recursive pattern' visualizations is given in Figure 2. The algorithm is initially called by `'draw(0,0,max_level)'` with the width and height of all recursion levels being stored in a previously defined array. The recursion is terminated by recursion level 0, in which case the algorithm actually draws one pixel. For recursion levels i (i ≥ 1), the algorithm draws w_i level(i-1)-patterns h_i times alternately to the right and to the left.

The maximum number of recursion levels, which is necessary to fill a screen of 1,024 x 1,024 pixels is 20, allowing

2^{20} (~1,000,000) data values to be displayed. Note that in this case the complete screen is filled with the visualization of one data dimension. If all dimensions are to be displayed on the screen, the number of data items which can be displayed is correspondingly lower. For 9 dimensions, e.g., the number of data items that can be displayed is about 116,000 (= $\left\lceil \frac{1024}{3} \right\rceil^2$), which corresponds to a maximum of 17 recursion levels. (The maximum number of recursion levels is reached by alternating between $(w, h) = (2, 1)$ and $(w, h) = (1, 2)$ on successive recursion levels.)

2.2 Examples of Possible Arrangements

As described above, one major difference between the ‘recursive pattern’ technique and other visualization techniques is, that it is not based on a fixed algorithm for arranging the pixels, but on a generic algorithm allowing the user (and/or application) to control the arrangement of pixels. By specifying the height and width for each of the recursion levels, users may adapt the generated visualizations to their specific needs. This allows our ‘recursive pattern’ technique to be used for a wide range of tasks and applications.

The most simple data arrangement is the already mentioned line-by-line or column-by-column arrangement. An easy way to obtain an arrangement similar to the line-by-line arrangement is to use $\lceil \sqrt{\#\text{data items}} \rceil$ for the height and width of recursion level one. This parameter setting results in a line-by-line back-and-forth arrangement as denoted in Figure 3a. To obtain the line-by-line orientation denoted in Figure 1a, the user may use $(w_1,h_1) = (\lceil \sqrt{\#\text{data items}} \rceil, 1)$ and $(w_2,h_2) = (1, \lceil \sqrt{\#\text{data items}} \rceil)$, and to obtain the column-by-column arrangement denoted by Figure 1b, the user may use $(w_1,h_1) = (1, \lceil \sqrt{\#\text{data items}} \rceil)$ and $(w_2,h_2) = (\lceil \sqrt{\#\text{data items}} \rceil, 1)$. Although, in general, these arrangements do not provide the best visualizations, the user may still want to start with these kind of arrangements, especially if the user does not know the inherent structure of the data and just wants to get a first impression of the data. The resulting visualization may help the user to find some inherent structure of the data, leading to more appropriate parameter settings using more recursion levels. Even if some inherent structure of the data is known a-priori, it will often be useful to start with a simple and straightforward arrangement since it may reveal additional structure of the data.

In general, the visualizations get more expressive by using more than one recursion level. Even if the recursion levels do not correspond to the inherent structure of the data, the clustering of a certain amount of closely related data items into a lower-level pattern helps to make data characteristics visible. Even bigger is the advantage if there is a grouping of a constant number of data items as, e.g., in time series data. Suppose, a data set consists of data values measured nine times a day for three consecutive weeks. To visualize this data set using the ‘recursive pattern’ algorithm, the user may enter the parameters $(w_1, h_1) = (3, 3)$ and $(w_2, h_2) = (3, 7)$, with the level(1)-pattern describing a day and the level(2)-pattern representing the three weeks (c.f. Figure 3c). For larger data sets, the user may repeat this procedure, either by enlarging the size of the second level pattern (e.g. $(w_2, h_2) = (28, 54)$, which corresponds to four weeks per row) or by adding additional recursion levels denoting months, years, decades, and so on. The user may also use a completely recursive arrangement which may be achieved by using $(w_i, h_i) = (c_1, c_2)$ for all recursion levels. The arrangement resulting from a fully recursive arrangement with $(w_i, h_i) = (3, 3)$ *for* $i=1..3$ is shown in Figure 4. The level(1)-patterns are shown by the small ‘S’-like patterns, the ordering of level(2)-patterns is denoted by color (light to dark colors), and the ordering of level(3)-patterns is denoted by numbering the level(2)-patterns.

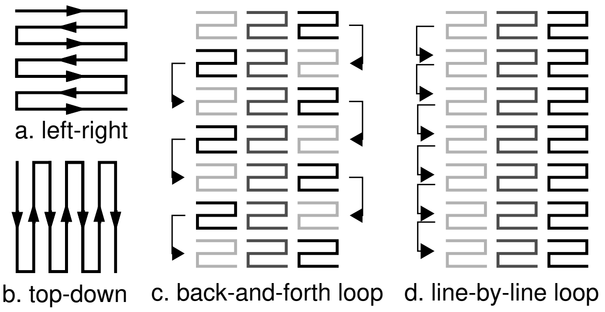

Figure 3: Examples of Possible Arrangements

In experimenting with various parameter settings, we found that the back-and-forth movement of higher level patterns is sometimes a bit confusing. The reason is that in our western culture we are used to read left-to-right in a line-by-line fashion. In scanning the visualization, our focus therefore moves left-to-right in a line-by-line fashion starting in the upper left corner. As reading is a frequent human activity, the perceptual system tries to ‘read’ the visualizations in the same way. The generality of the ‘recursive pattern’ technique allows the user to specify parameter settings which also support this natural fashion of ‘reading’ the visualization. Instead of the arrangement presented in Figure 3c, the user may use the parameter setting $(w_1, h_1) = (3, 3)$, $(w_2, h_2) = (3, 1)$, $(w_3, h_3) = (1, 7)$ to obtain the line-by-line arrangement on the second recursion level as denoted in Figure 3d. Note that the line-by-line left-to-right arrangement should only be used for higher recursion levels whereas on lower recursion levels the back-and-forth arrange-

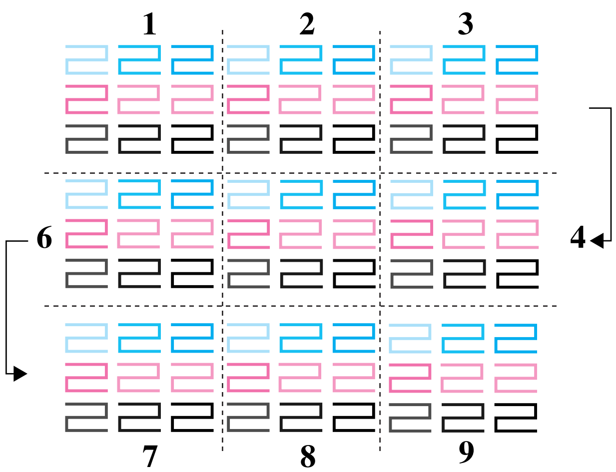

Figure 4: Fully Recursive Arrangement

ment is more appropriate to achieve a better clustering of closely related data values. If different arrangements such as top-down or bottom-up are more suitable for certain users (e.g., from a different culture) and/or applications, in most cases the arrangements can easily be generated by using appropriate parameters.

2.3 System Support for Finding Adequate Parameters

Since finding adequate parameters for the recursion levels is not always straightforward, the system supports the user in determining the parameters. The goal is to minimize the user's input in specifying the parameters and to maximize the screen utilization. In the first step, the system presents the number of data items as well as several standard parameter settings (e.g., line-by-line, column-by-column, fully recursive with given (w_i, h_i) or fixed number of recursion levels, etc.) to the user. If the user does not choose one of the standard parameter settings, for each of the subsequent recursion levels i, the system determines the number of level(i-1)-patterns $(\#pat_{i-1})$ which are necessary to present all data items. This number is important since, in general, the final recursion level (w_i, h_i) should be chosen such that

$$w_i \times h_i \geq \#pat_{i-1} \geq (w_i - 1) \times h_i \quad \text{and}$$

$$w_i \times h_i \geq \#pat_{i-1} \geq w_i \times (h_i - 1) \ .$$

Even if $\#pat_{i-1}$ is provided by the system, the user still needs to find adequate factors (w_i, h_i) whose multiplication is just larger than $\#pat_{i-1}$. The system supports this process by proposing parameters which allow a good screen utilization. After the user has specified the parameters for an arbitrary recursion level, the system checks the condition

$$\prod_{i=1}^{k} w_i \times h_i \leq \#data\ items$$

and as long as the condition is fulfilled, the user is asked for the parameters for an additional recursion level.

3. Using the 'Recursive Pattern' Technique for Exploring Financial Data

In this chapter, we provide example visualizations which are generated using the 'recursive pattern' technique. The considered databases are two stock exchange databases. The first database contains the prices of the IBM stock, Dow Jones index, and Gold as well as the exchange rate of the US-Dollar, from Sep. '87 to Feb. '95 with nine data items referring to one day. The database consists of 64,800 data values (16,200 data entries per stock). The second database contains the stock prices of the FAZ index (Frankfurter Aktien Index) from Jan. '74 to Apr. '95 on a daily basis. The FAZ index contains 100 important stock prices which are traded at the Frankfurter stock exchange. The database consists of 532,900 data values[1].

The traditional technique for visualizing financial data are X-Y diagrams which effectively show the ups and downs of stock prices. X-Y diagrams, however, only allow the visualization of about 500 data items (c.f. Figure 5), which for our first database corresponds to a period of less than four months. The only possibility for displaying larger data sets is to use sampling or aggregate (average, minimum, maximum) techniques, which implies not to represent every data item.

Using the 'recursive pattern' technique, the user may represent a much larger amount of data. Usually, it makes sense to start with quite simple representations to get a first general impression of the data. In case of the first database, the user may, for example, start with the straightforward line-by-line square arrangement using $\lceil \sqrt{16200} \rceil = 128$ as height and width. There are many other possibilities for visualizing the database including the parameter settings $(w_1, h_1) = (1{,}27)$, $(w_2, h_2) = (634{,}1)$ for grouping three days (27 data items) into the first level pattern and $(w_1, h_1) = (3, 3)$, $(w_2, h_2) = (24, 1)$, $(w_3, h_3) = (1, 80)$ for grouping one day into the first level pattern and one month in the second level pattern.

More useful, however, are arrangements with a higher structure. One possibility for a highly structured arrangement of the data is presented in Figure 6. The corresponding parameters are $(w_1, h_1) = (3, 3)$, $(w_2, h_2) = (5, 1)$,

1. It turned out to be quite difficult to obtain stock data with more than 500 to 1,000 data entries (usually on a daily basis). This is due to the fact that most data providers only store past data for presenting it using the common x-y diagrams. For longer periods, they usually store only the maximum, minimum, and average value per month since only this data can be visualized by traditional techniques.

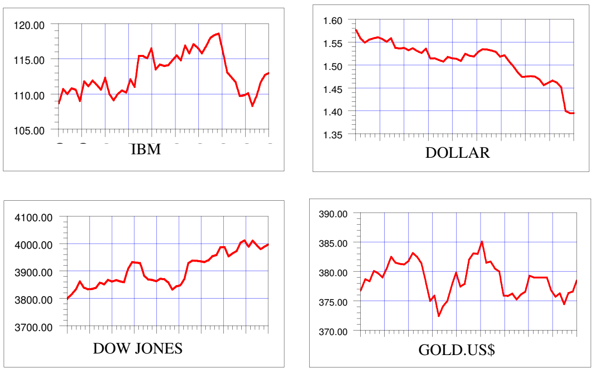

Figure 5: X-Y Diagrams

$(w_3, h_3) = (1, 4), (w_4, h_4) = (12, 1)$ and $(w_5, h_5) = (1, 8)$. In this case, the level(1)-pattern represents one day, the level(2)-pattern one week, the level(3)-pattern one month, and the level(4)-pattern one year. In the resulting visualization, the eight horizontal bars correspond to the eight years and the subdivision of the bars to the 12 months within each year. By having this structure in the visualization, it is easy to get detailed information from the dense pixel display. The user may, for example, easily see that the gold price was very low in the sixth year, that the IBM price quickly fell after the first one and a half months, that the US-Dollar exchange rate was highest in June 1989, etc. These are only a few examples for useful information which can be directly seen in the visualization.

To recall, the data is sorted according to time which means that the pixels to the left represent the older stock prices and the pixels to the right the more recent ones. The coloring maps high data values to light colors and low data values to dark colors. The colormap used is based on a linear interpolation within the HSI color model using a constant saturation, an increasing value (intensity), and a hue (color) ranging from yellow over green, blue, and red to almost black. The HSI color model [Kei 94] is a variation of the HSV model, which uses a circular cone instead of the hexcone of the HSV model. The reason for using the HSI over the HSV model is that linear interpolations within the HSI model can be used to generate colormaps with a monotonically decreasing brightness whereas linear interpolations within the HSI model usually do not have this property. At this point, it should be noted that the users may also use their own colormaps.

Note that many other highly structured arrangements are possible by using different parameter settings, e.g., $(w_i,h_i) = (3,3)$ *for all* i to achieve a fully recursive arrangement; $(w_1, h_1) = (1, 18)$, $(w_2, h_2) = (144, 1)$, $(w_3, h_3) = (1, 7)$ to get a grouping of two days in a vertical line and horizontal bars which correspond to the years; etc. Furthermore, note that our visualization technique can also be used for data without a-priori sequential ordering.

Our second example database, the FAZ index from Jan. ‘74 to Apr. ‘95, is presented in Figure 7. The chosen parameter setting is $(w_1, h_1) = (1, 22)$ $(w_2, h_2) = (243, 1)$, and the coloring maps high values to light colors and low values to dark colors. Since some of the stock prices do not exist for the whole time period, the remaining space is filled with the background color. It is obviously impossible to visualize all the 532,900 data values using conventional techniques.

In contrast, with the ‘recursive pattern’ technique we are able to visualize the whole database and reveal interesting properties of the data. Interesting are, for example, the similarities between different stock price developments. Using the visualization, it is easy to find stocks with similar stock price developments; for example, the stock price developments of the 1th, 4th, 8th, 10th, and 15th stock in the fourth column (Südzucker, Thyssen, Veba, Volkswagen, Bayr. Hypobank) are similar, although the companies are working in completely different areas. Another example is the stock price development of the 3rd- and 2nd-last stock in the third column and the 4th- and 5th-last stock in the forth column (Binding-Brauerei, Siemens, Lufthansa, Allianz), all four having multiple peaks in ‘87, ’90 and ‘93. Also interesting is that in more than 50% of all ‘price boxes’, there is a light green stripe at nearly the same position, especially the bottom boxes in the first and second columns and the top boxes of the third column. This means that those stocks had their peak price around the same time which was in spring ‘90. From the visualization, it becomes obvious that many of those stocks did not completely recover from the crash that followed. Further interesting are the stocks which do not follow the overall trend. An example is the 11th and 19th stock in the first column (Daimler-Benz, DYWIDAG) which did not have serious fluctuation over the last years and continuously remained on the same relatively high price level. Another example is the seventh and tenth stock in the second column (Harpen, MAN ST), which have their peak prices right at the beginning of the considered time interval (about 20 years ago).

We also compared our visualizations with other techniques for visualizing multidimensional data such as the ‘parallel coordinates’ technique [ID 90]. We found that the parallel coordinates technique is quite useful to find dependencies between data dimensions and deduce similar information as in the case of the ‘recursive pattern’ visualizations. However, the number of dimensions that can be visualized simultaneously is limited and there are also many data characteristics which cannot be seen due to the overlay of lines in the parallel coordinates technique. In general, the parallel coordinate technique is only suitable for a limited amount of data with limited dimensionality or for a focussed search within a large data set, whereas the 'recursive pattern' technique is able to provide an overview of very large amounts of data and helps the users to direct their search.

4. Conclusions

In this paper, we presented the 'recursive pattern' visualization technique as an approach for visualizing large amounts of data which is more general than previous approaches. Using our technique, the user may generate visualizations of very large amounts of multidimensional data, which provide a good overview of the data. The number of data items which can be visualized is only limited by the number of pixels of the display. This means for our stock price database that a user may, for example, get an overview of more than 20 years of daily stock price data from one hundred companies. A further advantage of our technique is that it allows the user to control the arrangement of the data values, which is important not only for sorted data sets but also for data without inherent structure or natural ordering. Our generic recursive algorithm allows the generation of many different representations of the data, consisting of semantically meaningful substructures. Our first results show that our technique is very powerful for visualizing large amounts of data and generalizes a wide range of pixel-oriented arrangements. In our future work, we will investigate space-filling curves such as the z-ordering or Hilbert curve with respect to their suitability for visualizing large amounts of multidimensional data. We will also apply our technique in different applications to explore its strengths and weaknesses in order to further improve the technique.

References

[And 57] Anderson E.: *'A Semigraphical Method For The Analysis of Complex Problems',* Proc. Nat. Acad. Sci. USA, Vol. 13, 1957, pp. 923-927.

[And 72] Andrews D. F.: *'Plots of High-Dimensional Data',* Biometrics, Vol. 29, 1972, pp. 125-136.

[AS 94] Ahlberg C., Shneiderman B.: *'Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Displays'*, Proc. ACM CHI Int. Conf. on Human Factors in Computing (CHI94), Boston, MA, 1994, pp. 313-317.

[Asi 85] Asimov D.: *'The Grand Tour: A Tool For Viewing Multidimensional Data',* SIAM Journal of Science & Stat. Comp., Vol. 6, 1985, pp. 128-143.

[AW 95a] Ahlberg C., Wistrand E.: *'IVEE: An Environment for Automatic Creation of Dynamic Queries Applications',* Proc. ACM CHI Conf. Demo Program (CHI95), 1995.

[AW 95b] Ahlberg C., Wistrand E.: *'IVEE: An Information Visualization and Exploration Environment',* Proc. Int. Symposium on Information Visualization, Atlanta, GA, 1995.

[AWS 92] Ahlberg C., Williamson C., Shneiderman B.: *'Dynamic Queries for Information Exploration: An Implementation and Evaluation',* Proc. ACM CHI Int. Conf. on Human Factors in Computing (CHI92), Monterey, CA, 1992, pp. 619-626.

[BC 95] Becker R., Cleveland B.: *'Trellis displays',* Proc. Workshop on Design and Implementation of Data Analysis Systems, Heidelberg, Germany, 1995.

[Bed 90] Beddow J.: *'Shape Coding of Multidimensional Data on a Mircocomputer Display'*, Visualization '90, San Francisco, CA, 1990, pp. 238-246.

[BF 90] Beshers C., Feiner S.: *'Visualizing n-Dimensional Virtual Worlds with n-Vision'*, Computer Graphics, Vol. 24, No. 2, 1990, pp. 37-38.

[Bri 79] Brissom D.: *'Hypergraphics: Visualizing Complex Relationships in Art, Science and Technology',* Amer. Association for the Advance of Science, Westview Press, Boulder, 1979.

[Che 73] Chernoff H.: *'The Use of Faces to Represent Points in k-Dimensional Space Graphically',* Journal Amer. Statistical Association, Vol. 68, pp 361-368.

[Cle 93] Cleveland W. S.: *'Visualizing Data',* AT&T Bell Laboratories, Murray Hill, NJ, Hobart Press, Summit NJ, 1993.

[Eic 94] Eick S.: *'Data Visualization Sliders'*, Proc. ACM UIST'94, 1994.

[Hub 85] Huber P. J.: *'Projection Pursuit',* The Annals of Statistics, Vol. 13, No. 2, 1985, pp. 435-474.

[ID 90] Inselberg A., Dimsdale B.: *'Parallel Coordinates: A Tool for Visualizing Multi-Dimensional Geometry'*, Visualization '90, San Francisco, CA, 1990, pp. 361-370.

[Ins 81] Inselberg A.: *'N-Dimensional Graphics Part I: Lines & Hyperplanes',* IBM LA Science Center Report, # G320-2711, 1981.

[KK 94] Keim D. A., Kriegel H.-P.: *'VisDB: Database Exploration using Multidimensional Visualization',* Computer Graphics & Applications, Sept. 1994, pp. 40-49.

[Kei 94] Keim D. A.: *'Visual Support for Query Specification and Data Mining'*, Ph.D. thesis, University of Munich, Juli 1994, Shaker Publishing Company, Aachen, 1995, ISBN 3-8265-0594-8.

[LWW 90] LeBlanc J., Ward M. O., Wittels N.: *'Exploring N-Dimensional Databases'*, Visualization '90, San Francisco, CA, 1990, pp. 230-239.

[MGTS 90]Mihalisin T., Gawlinski E., Timlin J., Schwendler J.: *'Visualizing Scalar Field on an N-dimensional Lattice'*, Visualization '90, San Francisco, CA, 1990, pp. 255-262.

[MTS 91] Mihalisin T., Timlin J., Schwegler J.: *'Visualizing Multivariate Functions, Data and Distributions',* IEEE Computer Graphics and Applications, Vol. 11, No. 3, 1991, pp. 28-35.

[MZ 92] Marchak F., Zulager D.: *'The Effectiveness of Dynamic Graphics in Revealing Structure in Multivariate Data'*, Behavior, Research Methods, Instruments and Computers, Vol. 24, No. 2, 1992, pp. 253-257.

[SBM 93] Sparr T. M., Bergeron R. D., Meeker L. D.: *'A Visualization-Based Model for a Scientific Database System',* in: Focus on Scientific Visualization, Hagen H., Müller H., Nielson G.M. (eds.), Springer, 1993, pp. 103-121.

[SCB 92] Swayne D.F., Cook D., Buja A.: *'User's Manual for XGobi, a Dynamic Graphics Program for Data Analysis',* Bellcore Technical Memorandum, 1992.

[SGB 91] Smith S., Grinstein G., Bergeron R. D.: *'Interactive Data Exploration with a Supercomputer',* Visualization '91, San Diego, CA, 1991, pp. 248-254.

[Shn 92] Shneiderman B.: *'Tree Visualization with Treemaps: A 2-D Space-filling Approach'*, ACM Trans. on Graphics, Vol. 11, No. 1, 1992, pp. 92-99.

[The 95] Theus M.: *'Data Desk',* Proc. Workshop on Design and Implemetation of Data Analysis Systems, Heidelberg, Germany, 1995.

[Tuf 83] Tufte E. R.: *'The Visual Display of Quantitative Information',* Graphics Press, Cheshire, CT, 1983.

[Tuf 90] Tufte E. R.: *'Envisioning Information',* Graphics Press, Cheshire, CT, 1990.

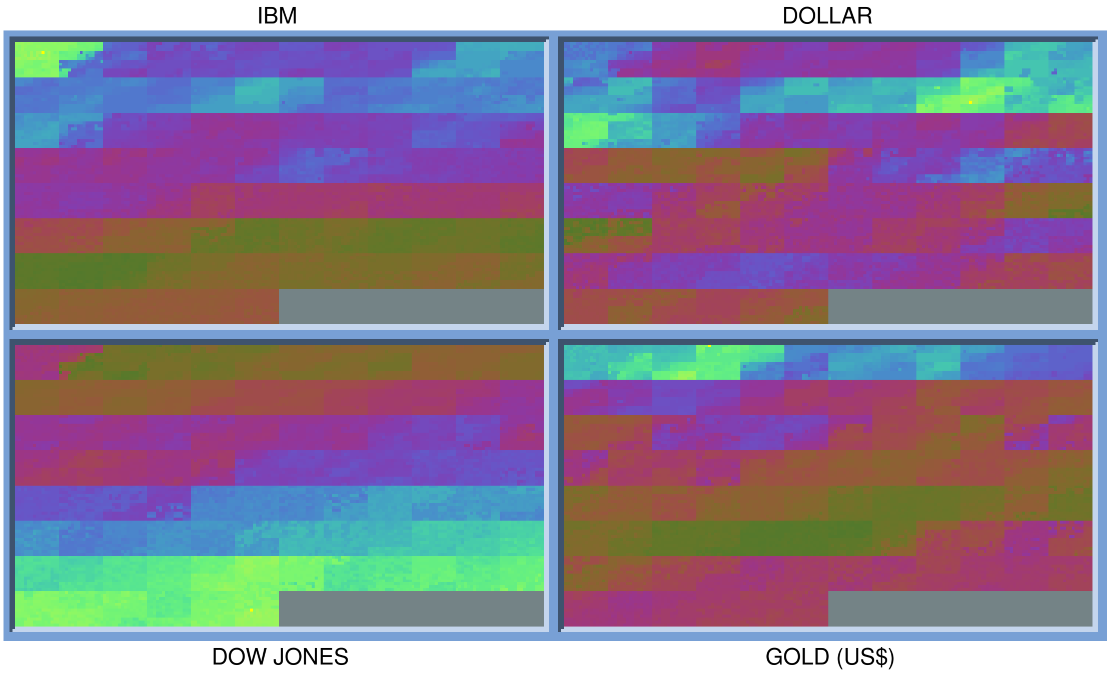

Figure 6: Five Level Recursive Arrangement (Sep. ‘87 - Feb. ‘95)

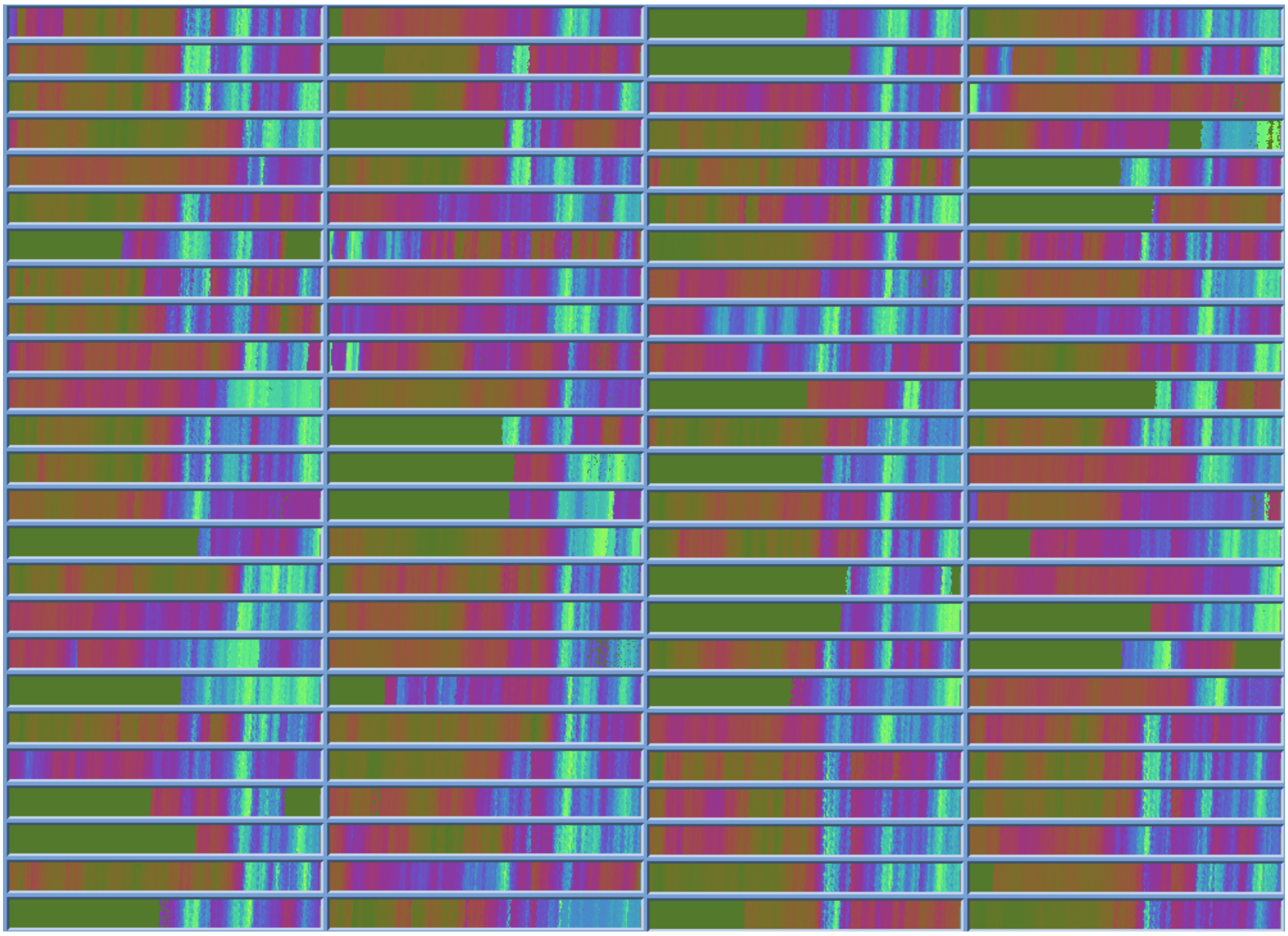

Figure 7: Stock Prices of the FAZ Index (Jan. ‘74 - Apr. ‘95)