Pixel-oriented Visualization Techniques for Exploring Very Large Databases

Daniel A. Keim

Institute for Computer Science
University of Munich
Leopoldstr. 11 B, D-80802 Munich, Germany
keim@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 describe a set of pixel-oriented visualization techniques which use each pixel of the display to visualize one data value and therefore allow the visualization of the largest amount of data possible. Most of the techniques have been specifically designed for visualizing and querying large databases. The techniques may be divided into query-independent techniques which directly visualize the data (or a certain portion of it) and query-dependent techniques which visualize the data in the context of a specific query. Examples for the class of query-independent techniques are the screen-filling curve and recursive pattern techniques. The screen-filling curve techniques are based on the well-known Morton and Peano-Hilbert curve algorithms, and the recursive pattern technique is based on a generic recursive scheme which generalizes a wide range of pixel-oriented arrangements for visualizing large data sets. Examples for the class of query-dependent techniques are the snake-spiral and snake-axes techniques, which visualize the distances with respect to a database query and arrange the most relevant data items in the center of the display. Beside describing the basic ideas of our techniques, we provide example visualizations generated by the various techniques, which demonstrate the usefulness of our techniques and show some of their advantages and disadvantages.

Keywords: Visualizing Large Data Sets, Visualizing Multidimensional and Multivariate Data, Visualizing Large Databases

1. Introduction

One of today’s problems in explorative data analysis is the rapidly increasing amount of data that needs to be analyzed. The automation of activities in all areas, including business, engineering, science, and government, produces an ever-increasing stream of data. The data is collected in very large databases because people believe that it contains valuable information. Extracting the valuable information, however, is a difficult task. Even with the most advanced data analysis systems, finding the right piece of information in a very large database with millions of data items remains a difficult and time-consuming process. The process cannot be fully automated since it involves human intelligence and creativity which are 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.

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 and existing techniques have been extended to work for larger data sets and make the displays interactive. For most of the data stored in databases, however, there is no standard mapping into the Cartesian coordinate system, since the data has no inherent two- or three-dimensional semantics. In general, relational databases can be seen as multivariate data sets with the attributes of the database corresponding to the variables of the multivariate data set. There are several well known techniques for visualizing multivariate data sets: scatterplot matrices and coplots [And 72, Cle 93], prosection matrices [FB 94], parallel coordinates [Ins 81, ID 90], projection pursuit [Hub 85], and other geometric projection techniques (e.g., hyperbox [AC 91] and hyperslice [WL 93]), iconic display techniques (e.g., [Che 73, PG 88, Bed 90, SBM 93]), hierarchical techniques (e.g., [BF 90, LWW 90, RCM 91, Shn 92]), dynamic techniques (e.g., [BMMS 91, MZ 92, AWS 92, Eic 94, Shn 94, ADLP 95]), and combinations hereof (e.g. [Asi 85, AS 94]). The research also resulted in data exploration and analysis systems which implement some of the mentioned techniques. Examples include statistical data analysis packages such as S Plus / Trellis [BCW 88], XGobi [SCB 92], and Data Desk [Vel 92, WUT 95], visualization oriented systems such as ExVis [GPW 89], XmdvTool [Ward 94, MW 95], and IBM’s Parallel

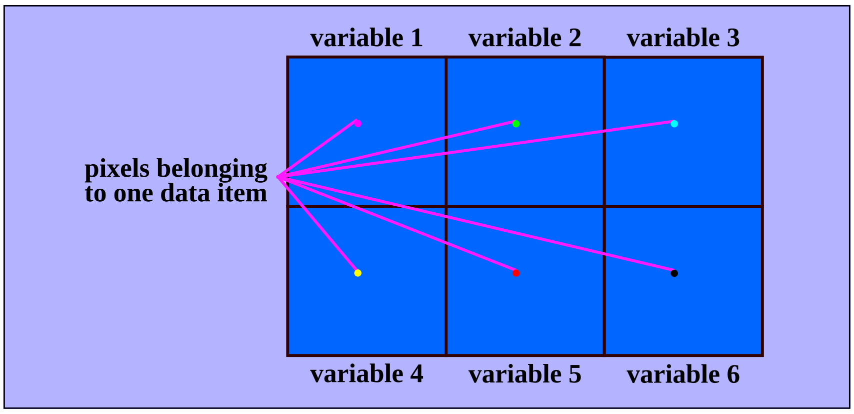


Figure 1: Arrangement of Windows for Data with Six Variables

Visual Explorer, as well as database oriented systems such as TreeViz and the Information Visualization and Exploration Environment (IVEE) [AW 95].

In this article, we present a novel class of techniques for visualizing multivariate data called *pixel-oriented techniques*. The basic idea of pixel-oriented techniques is to map each data value to a colored pixel and present the data values belonging to one variable in separate windows (cf. Figure 1). Since in general our techniques use only one pixel per data value, the techniques allow us to visualize the largest amount of data, which is 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 to arrange the pixels on the screen. Our pixel-oriented techniques use different arrangements for different purposes. If a user wants to visualize a large data set, the user may use a query-independent visualization technique which sorts the data according to some variable(s) and uses a screen-filling pattern to arrange the data values on the display. The query-independent visualization techniques are especially useful for data with a natural ordering according to one variable (e.g., time series data). However, if there is no natural ordering of the data and the main goal is an interactive exploration of the database, the user will be more interested in feedback to some query. In this case, the user may turn to the query-dependent visualization techniques which visualize the relevance of the data items with respect to a query. Instead of directly mapping the data values to color, the query-dependent visualization techniques calculate the distances between data and query values, combine the distances for each data item into an overall distance, and visualize the distances for the variables and the overall distance sorted according to the overall distance. The arrangement of the data items centers the most relevant data items in the middle of the window, and less relevant data items are arranged in a spiral-shape to the outside of the window.

The techniques described so far partition the screen into multiple windows. For data sets with m variables (attributes), the screen is partitioned into m windows — one for each of the variables. In case of the query-dependent techniques, an additional (m+1)th window is provided for the overall distance. Inside the windows, the data values are arranged according to the given overall sorting which may be data-driven for the query-independent techniques or query-driven for the query-dependent techniques. Correlations, functional dependencies, and other interesting relationships between variables may be detected by relating corresponding regions in the multiple windows.

Another class of techniques, the so-called grouping techniques, is based on the idea of combining the separate windows into one window. The principle idea is to group all values belonging to one data item into one area. The possibilities for arranging the pixel areas — each corresponding to one data item — are similar to the arrangements of the techniques which use multiple windows and can again be query-dependent or query-independent. Since additional pixels are needed to distinguish the pixel areas belonging to different data items, the number of data items that can be visualized with the grouping technique is lower compared to the techniques which use multiple windows.

An important aspect of our visualizations is the mapping of data values to color. 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, KK 95] is a variation of the HSV model [FDFH 90] and 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 provide colormaps with a monotonically decreasing brightness whereas linear interpolations within the HSV model usually do not have this property. A well-known problem of color mappings is that the perceived difference between colors does not necessarily correspond to the difference between the corresponding data values. The problem can be solved by using different color mappings. In our system, we therefore allow users to use their own color mappings.

In the rest of this paper, we present our pixel-oriented visualization techniques for exploring very large databases. Section 2 describes the query-independent techniques, section 3 the query-dependent techniques, and section 4 the grouping techniques. In section 5, we briefly discuss the implementation of our techniques as part of the *VisDB* system. Section 6 summarizes our approach and points out some of the open problems for future work.

2. Query-Independent Visualization Techniques

As already mentioned, the basic idea of our visualization techniques is to present as many data values as possible at the same time with the number of data values being only limited by the number of pixels of the display. In dealing with arbitrary multivariate 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 variable (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 (cf. Figure 2a). Another possibility is to arrange the data items top-down in a column-by-column fashion (cf. Figure 2b). If these arrangements are done pixelwise, in general, the resulting visualizations do not provide useful results. More useful are techniques which provide a better clustering of closely related data items and allow the user to influence the arrangement of the data. Techniques with nice clustering properties are screen-filling curves (cf. section 2.1). A technique which provides nice clustering properties as well as user influence on the arrangement is the recursive pattern technique (cf. section 2.2).

2.1 Screen-filling Curve Techniques: Peano-Hilbert Curve and Morton Curve

The screen-filling curves techniques are based on the well-known space-filling curve algorithms by Peano & Hilbert [Pea 90, Hil 91] and Morton [Mor 66]. The basic idea of space-filling curves is to provide a continuous curve which passes through every point of a regular spatial region (e.g., a square). For a long time, the space-filling curve algorithms were mainly used for studying recursion and for producing pretty pictures. Two decades ago, researchers started to use the spatial clustering properties of space-filling curves for indexing spatial databases [AG 80]. The basic idea is to optimize storage and processing of two-dimensional data by mapping them into one dimension. Space-filling curves provide a mapping which preserves the spatial locality of the original two-dimensional image. In visualizing multivariate data which is sorted according to one dimension, we have the opposite problem namely mapping a one-dimensional distribution of data

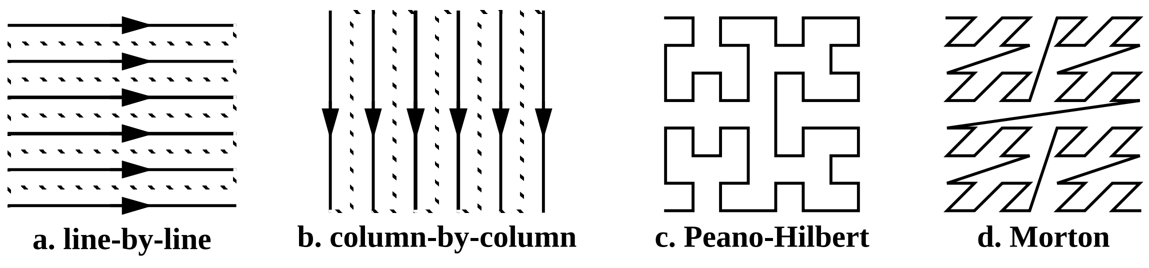


Figure 2: Data Arrangements

```
void PeanoHilbert(dir R, dir D, dir L, dir U, int level)
{if (level>0)
   {PeanoHilbert(D,R,U,L,level-1);
      Draw(R);
    PeanoHilbert(R,D,L,U,level-1);
      Draw(D);
    PeanoHilbert(R,D,L,U,level-1);
      Draw(L);
    PeanoHilbert(U,L,D,R,level-1);
   }
}
void Draw(dir d)
{switch (d)
   { case down:  SetPixel(x  ,y++,color); break;
     case up:    SetPixel(x  ,y--,color); break;
     case right: SetPixel(x++,y  ,color); break;
     case left:  SetPixel(x--,y  ,color); break;
   }
}
called by  SetPixel(startX,startY,color);
           PeanoHilbert(right,down,left,up,max_level);
```

Figure 3: Peano-Hilbert Algorithm (cf. [Gol 81])

items onto the two dimensions of the screen. For this purpose, we may also use space-filling curves since space-filling curves have the nice property that data items which are close together in the one-dimensional distribution are likely to be close together in the two-dimensional visual representation. This property of space-filling curves may help to discover patterns in the data which are difficult to discover otherwise. If each of the variables is visualized using the same arrangement, interesting properties of the data may be revealed including the distribution of variable values, correlations between variables, and clusters.

Both, the Peano-Hilbert and the Morton curve can best be described by recursive algorithms. Both algorithms successively fill squares of sizes (2^i x 2^i), i = 0, ..., max_level. A pattern of size (2^i x 2^i) always consists of four subpatterns of size (2^{i-1} x 2^{i-1}). In the Morton curve, the orientation of subpatterns is fixed, whereas in the Peano-Hilbert curve the orientation of subpatterns changes. In contrast to the Peano-Hilbert curve, however, the Morton curve does not provide a continuous path through the squares. Because the basic pattern of the Morton curve resembles the shape of a ‘z’, the Morton curve is sometimes also referred to as ‘z’-curve. Schematic representations of the Peano-Hilbert and Morton arrangements are provided in Figures 2c and 2d. A short and elegant recursive algorithm for generating Peano-Hilbert visualizations is given in Figure 3 (cf. [Gol 81]); the algorithm for generating Morton visualizations is provided in Figure 4.

An example for a visualization generated by using the Peano-Hilbert algorithm for arranging the colored pixels (corresponding to the variable values) on the screen is provided in Figure 5a. The

```
void Morton(int x, int y, int level)
{if (level>0)
   {Morton(x         ,y          ,level-1);
    Morton(x+pow(2,i) ,y          ,level-1);
    Morton(x         ,y+pow(2,i) ,level-1);
    Morton(x+pow(2,i) ,y+pow(2,i) ,level-1);
   }
 else {   // level==0
    SetPixel(x  ,y  ,color);
    SetPixel(x++,y  ,color);
    SetPixel(x  ,y++,color);
    SetPixel(x++,y++,color);
 }
}
called by  Morton(startX,startY,max_level);
```

Figure 4: Morton Algorithm

visualization shows a stock exchange database containing 16,350 data items[1] with the price of the IBM stock, Dow Jones index, and Gold as well as the exchange rate of the US-Dollar, from January ‘87 to March ‘93 with nine data items referring to one day. Since 16,350 is about 2^{14}, we have 14 recursion levels in the Peano-Hilbert and Morton visualizations. According to our experience, visualizations generated by the Peano-Hilbert technique are quite difficult to read and interpret. While the Peano-Hilbert curve clearly shows clusters of variable values, it is difficult to follow the curve, even if one knows the Peano-Hilbert algorithm. Since only one pixel is used to represent one variable value, it is quite difficult to relate the multiple windows for the different variables.

In case of the Morton curve, finding correlations between the windows corresponding to the different variables is much easier. This is due to the fact that the Morton arrangement is more regular and therefore it is easier to distinguish subpatterns and pursue their ordering — in other words, it is easier to follow the curve. In Figure 5b, we present a visualization generated by the Morton technique showing the same data set as presented in Figure 5a. Clearly distinguishable are the patterns generated by the three final recursion levels (cf. first, second, and third level subsquares in Figure 5b). An advantage over the Peano-Hilbert visualization is that the sequence of subsquares is more clear and therefore it is also easier to relate the visualization windows for the different variables. Since the structure of the visualizations is fixed to squares of sizes (2^i x 2^i), the structure does not have any meaning and is not related to the semantics of the data.

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.

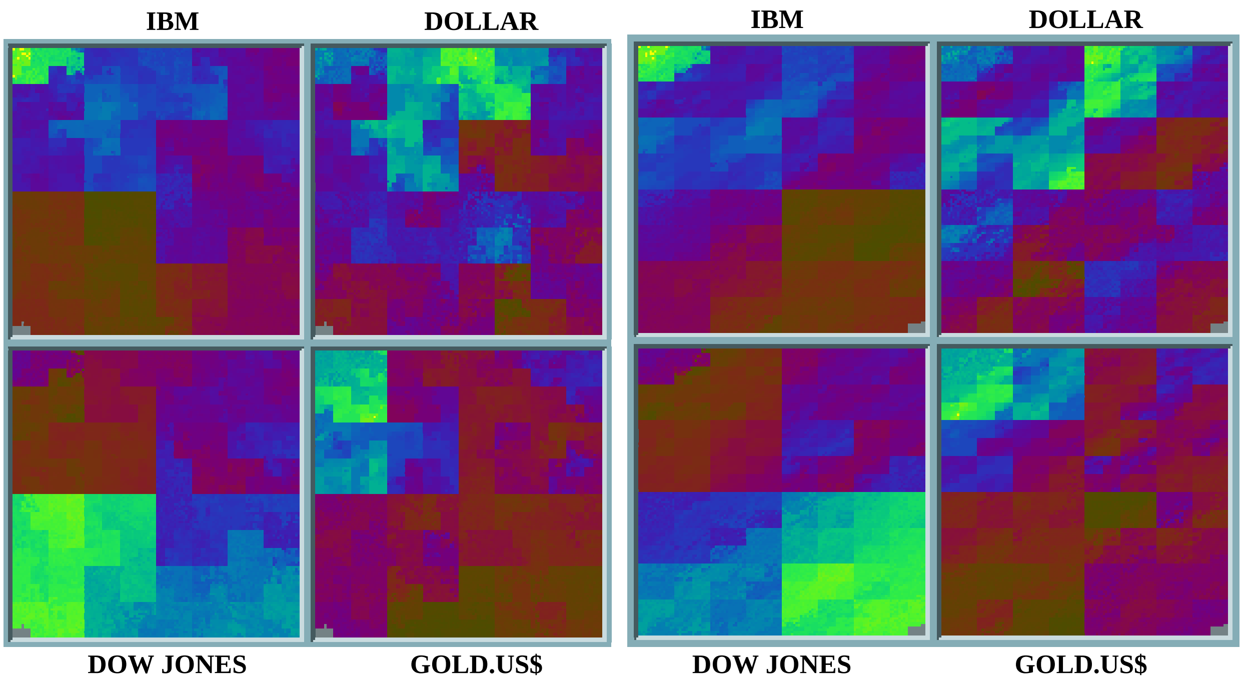


Figure 5: Visualizations generated by the Screen-filling Curve Techniques

2.2 Recursive Pattern Technique

A technique which helps to overcome this drawback is the *recursive pattern technique*. The basic idea is to retain the clustering properties of the Peano-Hilbert and Morton techniques but allow the user to influence the arrangement of pixels that it becomes semantically meaningful. We believe that this is important since in many cases the data has some inherent structure which should be reflected by 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 a first level pattern, those belonging to one week in a second level pattern, those belonging to one month in a 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 visualizations. This requirement is reflected by the generic definition of the recursive pattern technique. As the Peano-Hilbert and Morton arrangements, the recursive pattern technique is based on a recursive scheme. In contrast to the Peano-Hilbert and Morton techniques, however, it allows user-defined parameter settings for the various recursion levels.

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

```
void RecPattern(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++)
            {RecPattern(x,y,level-1);     // recursive call of algorithm
            x+=next_x[level-1];
         }}
      else      // even height ?
         {for (int w=1; w<=width[level]; w++)
            {x-=next_x[level-1];
            RecPattern(x,y,level-1);      // recursive call of 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 6: Recursive Pattern Algorithm (cf. [KKA 95])

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 6. The algorithm is initially called by `‘RecPattern(startX,startY,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.

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 user interaction in specifying the parameters and to maximize the screen utilization. One way of supporting the

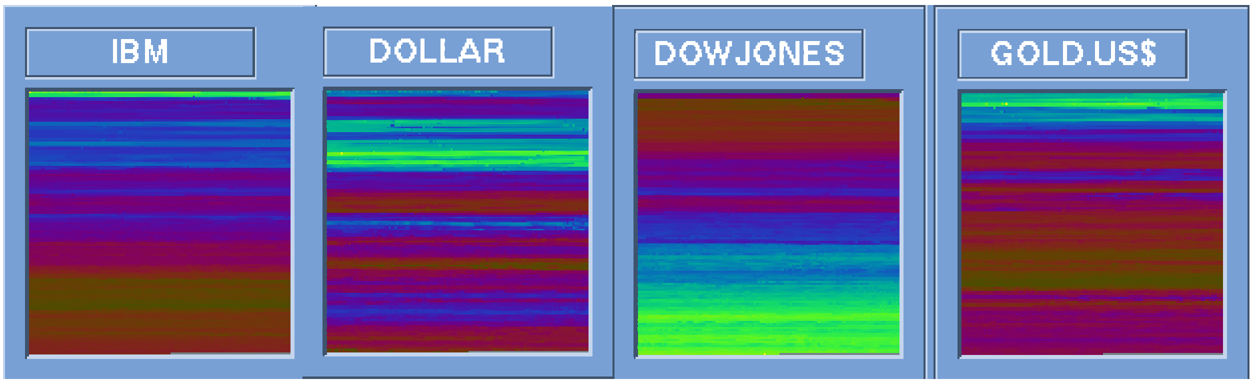


Figure 7: Line-by-Line Back-and-Forth Arrangement $[(w_1,h_1) = (\lceil\sqrt{16350}\rceil,\lceil\sqrt{16350}\rceil)]$

user is to provide options for standard parameter settings such as line-by-line (cf. Figure 2a), column-by-column (cf. Figure 2b) and fully recursive (with given (w_i,h_i) for all recursion levels). If the user however does not choose one of these options, the system checks the appropriateness and validity of the given parameters (e.g., $w_k \times (h_k - 1) \times \prod_{i=1}^{k-1} w_i \times h_i \le \#data\ items$ and $(w_k - 1) \times h_k \times \prod_{i=1}^{k-1} w_i \times h_i \le \#data\ items$) and proposes suitable modifications.

Using the recursive pattern visualization technique the user is able to generate a wide range of different pixel-oriented visualizations. The user may, for example, start with a simple line-by-line back-and-forth square arrangement (cf. Figure 2a) by using only one recursion level and specifying $\lceil\sqrt{16350}\rceil = 128$ as height and width. The resulting visualization is presented in Figure 7[1]. In the visualization, the user may easily follow the development of all four stock prices. To recall, the data is sorted according to time which means that the pixels at the top represent the older stock prices and the pixels at the bottom the more recent ones. The coloring maps high data values to light colors and low data values to dark colors. In the visualization (cf. Figure 7), the user may easily realize that about four and a half years ago the exchange rate of the US-Dollar was at its highest point (very light), and two as well as four years ago the exchange rate was very low (very dark). Also interesting is that a pattern of higher exchange rates (green/blue horizontal lines) seems to occur pretty regularly.

A second example (cf. Figure 8) shows the effect of using the parameters $(w_1, h_1) = (1, 27)$ and $(w_2, h_2) = (634, 1)$. One vertical line (= level(1)-pattern) contains 27 data items, which corresponds

1. The quality of the printed version of our visualizations is rather bad compared to the quality of the visualizations on the screen. Structures in the visualizations which are easy to perceive on the screen may therefore be difficult to perceive in the printed version.

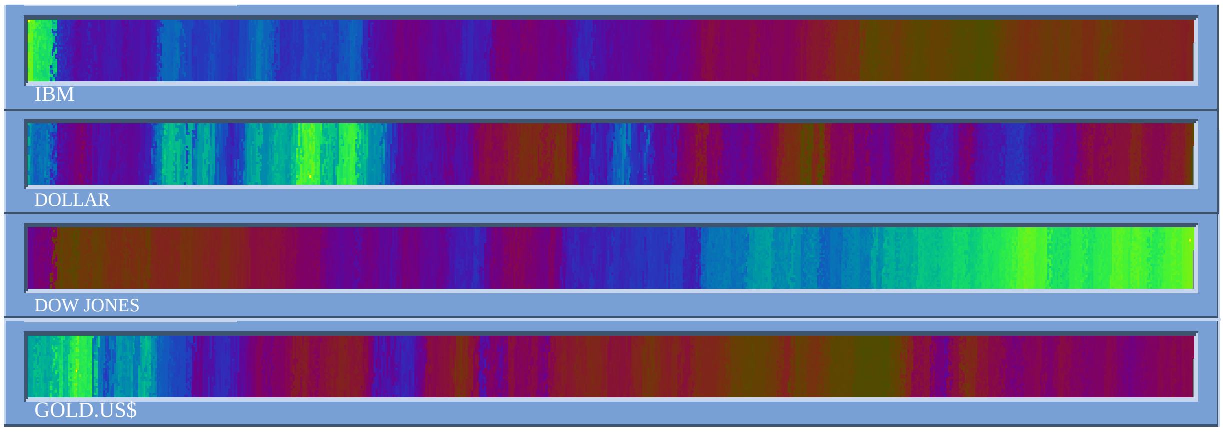


Figure 8: Horizontal Arrangement [$(w_1, h_1) = (1, 27), (w_2, h_2) = (634, 1)$]

to three days. In the visualization, the development of the four stock prices becomes a little bit clearer, but still the visualization is not satisfactory because there is only a limited clustering in the visualization. The third example (c.f. Figure 9) tries to improve the clustering by grouping days in the level(1)-pattern and months in the level(2)-pattern. The corresponding parameters are $(w_1, h_1) = (3, 3), (w_2, h_2) = (24, 1)$ and $(w_3, h_3) = (1, 80)$. In this case, a horizontal line of three pixels height contains the data within a month, which allows the user to easily relate interesting properties of the visualization to time. It is, for example, easy to see how ‘long’ a subset of the data has the same color, which corresponds to the time in which the price remained roughly constant.

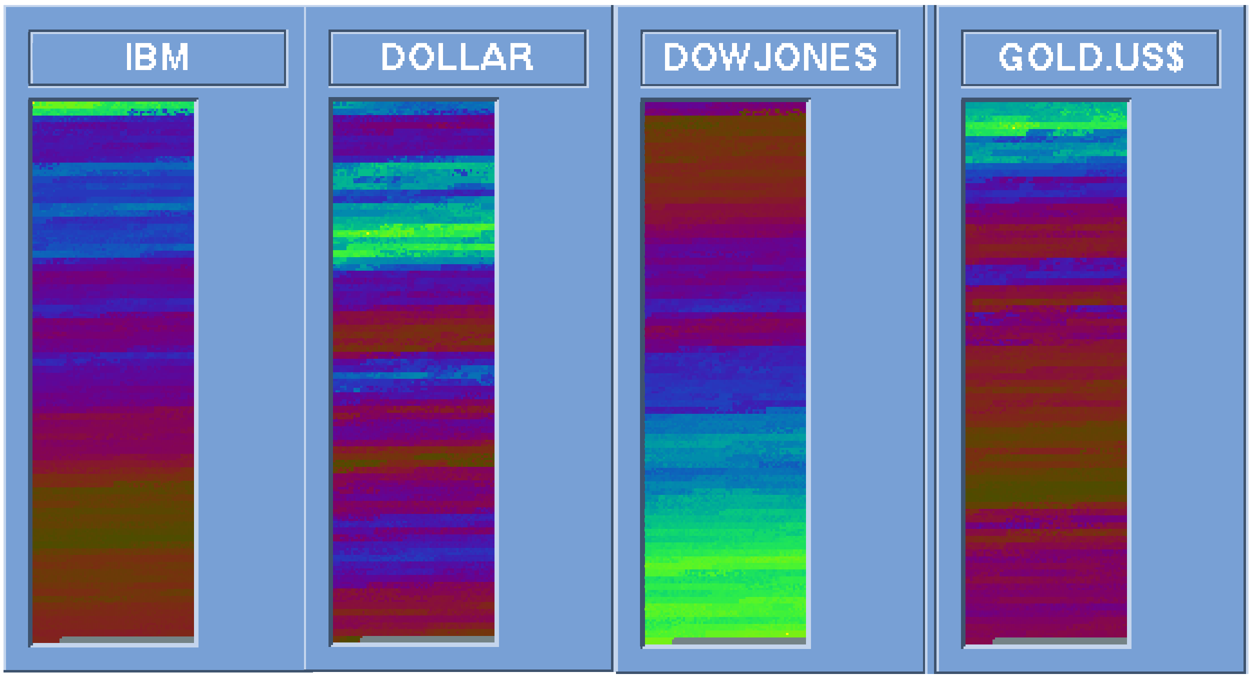


Figure 9: Monthly Arrangement [$(w_1, h_1) = (3, 3), (w_2, h_2) = (24, 1), (w_3, h_3) = (1, 80)$]

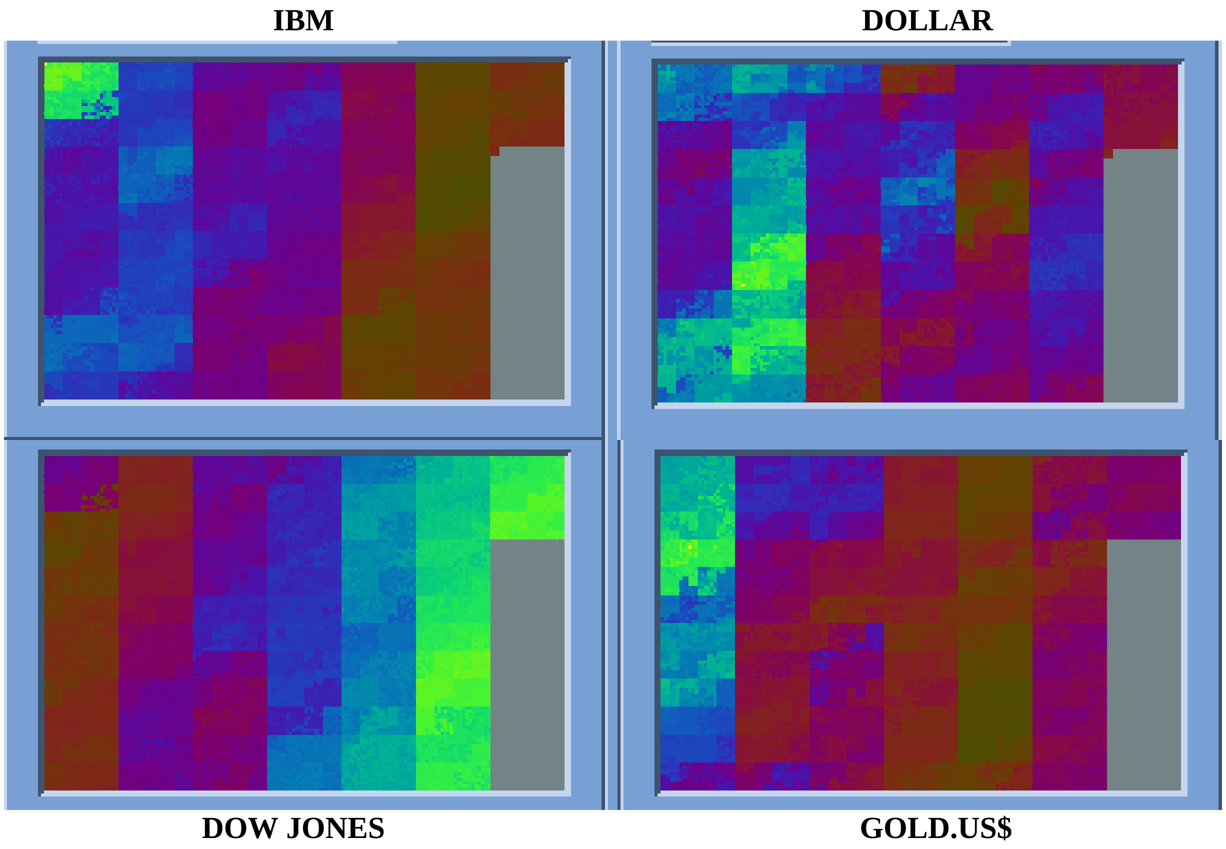


Figure 10: Highly Structured Arrangement

[$(w_1, h_1) = (3, 3), (w_2, h_2) = (2, 3), (w_3, h_3) = (4, 1), (w_4, h_4) = (1, 12), (w_5, h_5) = (7, 1)$]

The last example shows a highly structured recursive pattern visualization (cf. Figure 10). The visualization is generated by grouping days in the level(1)-pattern, weeks in the level(2)-pattern, months in the level(3)-pattern, and years in the level(4)-pattern. The corresponding parameters are $(w_1, h_1) = (3, 3)$, $(w_2, h_2) = (2, 3)$, $(w_3, h_3) = (4, 1)$, $(w_4, h_4) = (1, 12)$ and $(w_5, h_5) = (7, 1)$. In the visualization, areas of 3 x 3 pixels corresponds to the data of one day, areas of 6 x 9 pixels correspond to one week, slices of 24 x 9 pixels correspond to one month, and vertical bars of 24 x 108 correspond to one year. A schematic representation of the arrangement is provided in Figure 11. By structuring the visualization in such a way, it is easy to get detailed information from the dense pixel display containing a maximum of information. The user may, for example, easily see that the gold price was very low in 1991, that the IBM price quickly fell after the first two month, that the US-Dollar exchange rate was highest in autumn of 1988, etc. These are only a few examples for useful information which can be directly seen in the visualization. Note that many other arrangements may be generated by using different parameter settings; for example, $(w_i,h_i) = (3,3)$ *for all i* to achieve a fully recursive arrangement; $(w_1, h_1) = (1, 18)$, $(w_2, h_2) = (144, 1)$, $(w_3, h_3) =$

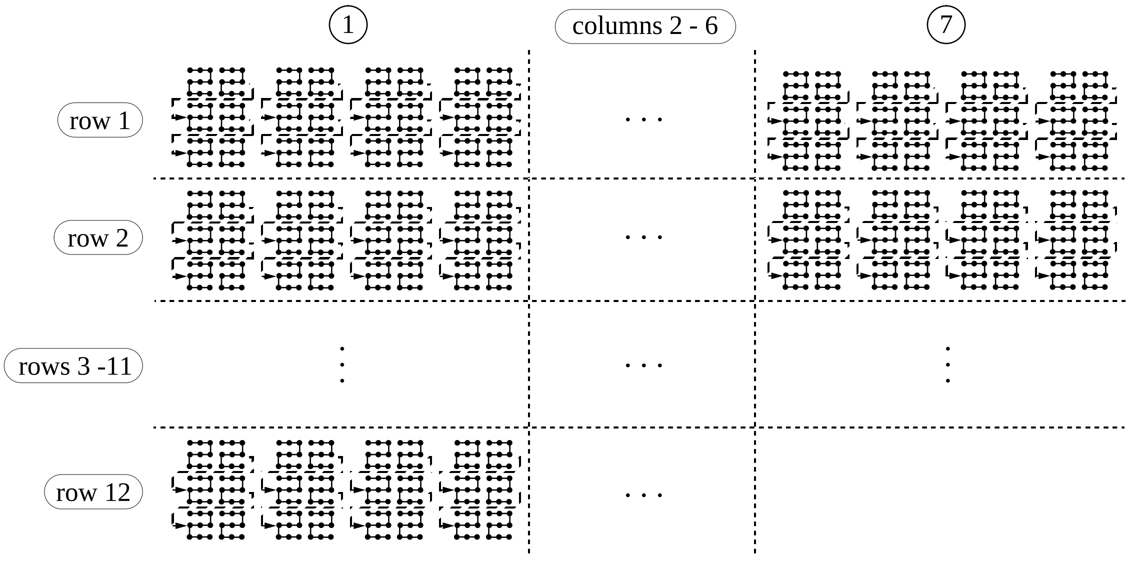


Figure 11: Schematic Representation of the Highly-Structured Arrangement (cf. Figure 10)

(1, 7) to get a grouping of three days in a vertical line and horizontal bars which correspond to the years; $(w_1, h_1) = (3, 3)$, $(w_2, h_2) = (24,1)$, $(w_3, h_3) = (1, 12)$, $(w_4, h_4) = (7, 1)$ to get a grouping of one month into a horizontal line and vertical bars which correspond to the years; etc.

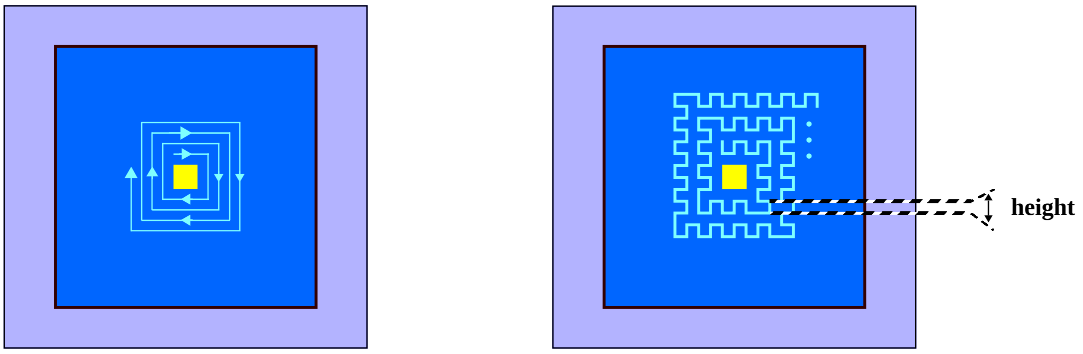
Note that for the query-independent techniques, it is not necessarily required that the data has some natural ordering. In searching for dependencies among variables, one might sort the data according to one variable and use our visualization technique for examining the dependencies of the other variables. Consider, for example, a large database of personal data. If one wants to find dependencies between the parameter sales (of a person) and other variables 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 variables.

3. Query-Dependent Visualization Techniques

The query-independent visualization techniques visualize the variable values by directly mapping them to color. The idea of the query-dependent visualization techniques is to visualize the data in the context of a specific user query to give the users feedback on their queries and direct their search. Instead of directly mapping variable values to colors, the distances of variable values to the query are mapped to colors. Since the focus of the query-dependent techniques is on the relevance of the data items with respect to the query, different arrangements of the pixels seem to be appropriate. In developing the system, we experimented with several arrangements such as the left-right or top-down arrangements. We found, that for visualizing the result for a database query it seems to be most natural to present the data items with highest relevance to the query in the center of the display. Our first approach described in [Kei 94, KK 94] was to arrange the data items with lower relevances in a rectangular spiral shape around the center (cf. Figure 12a and 14a). The snake techniques presented in this paper are a generalization of those techniques. Instead of arranging the data in a rectangular spiral shape, the spiral is extended to a generic wave- or snake-like form, of which the user may choose the height (cf. Figure 12b and 14b). The original spiral and axes techniques are now the special case of a snake with a height of one pixel. The advantage of the snake-like spiral form is that the degree of clustering is higher (cf. Figure 13).

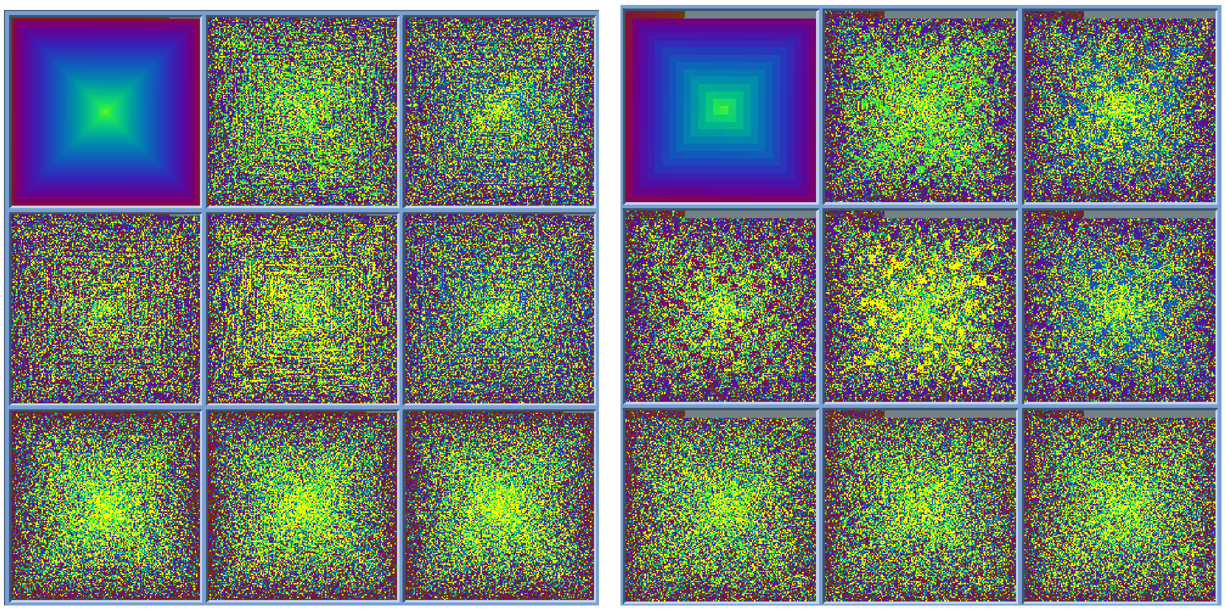
3.1 Snake-Spiral Technique

The basic idea for visually displaying the data on the screen is to present the one hundred percent correct answers in the middle of the window and the approximate answers sorted according to their overall distance (or relevance) in a wave- or snake-like spiral shape around this region (cf. Figure 12b). As in case of the query-independent visualization techniques, a separate visualization for each of selection predicates (variables) is generated (cf. Figure 1). An additional window shows the overall distances. In all of the windows, we place the pixels for each data item at the same position as the overall distance for the data item in the overall distance window. By relating corresponding regions in the different windows, the user is able to perceive data characteristics such as multidimensional clusters or correlations. Additionally, the separate windows for each of the selection predicates provide important feedback to the user, e.g. on the restrictiveness of each of the selection predicates and on single exceptional data items. An example visualization of about 24,000



a. Spiral Technique **b. Snake-Spiral Technique**

Figure 12: Snake-Spiral Technique



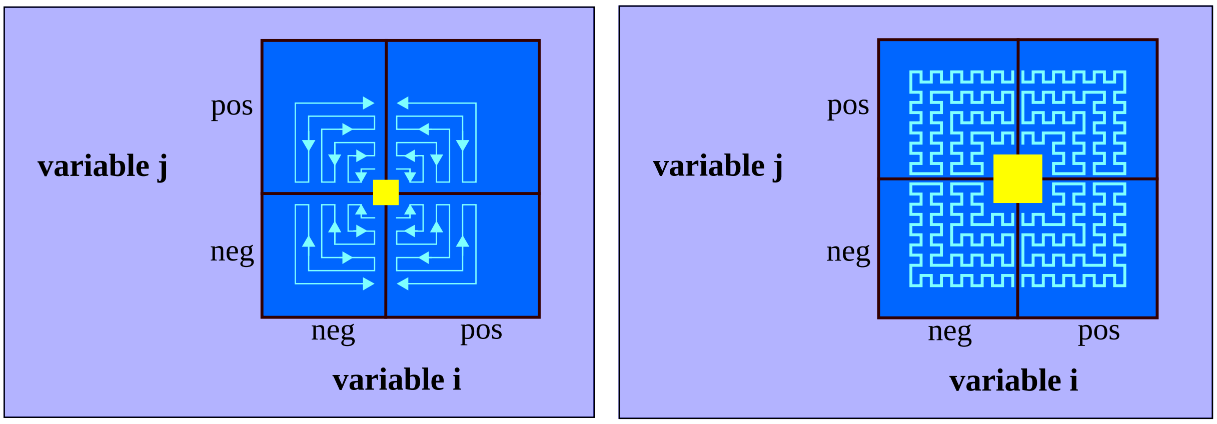
a. Spiral Technique **b. Snake-Spiral Technique**

Figure 13: Visualizations of Eight-Variate Data

test data items with eight variables is provided in Figure 13. Most of the data set (20,000 data items) is randomly generated in the range [-100, 100]. The remaining 4000 data items split up into two clusters which are only defined on the first five variables and are inserted at specific locations of the eight-dimensional space. The query used is [-20, 20] for each of the variables. Figure 13a shows the visualization generated by the original spiral technique and Figure 13b shows the visualization generated by the snake-spiral technique with a height of six pixels. The example clearly shows the advantage of the snake-spiral over the original spiral technique.

3.2 Snake-Axes Technique

The axes arrangement improves the spiral technique by including some feedback on the direction of the distance into the visualization. The basic idea is to assign two variables to the axes and to arrange the data items according to the direction of the distance; for one variable negative distances are arranged to the left, positive ones to the right and for the other variable negative distances are arranged to the bottom, positive ones to the top (cf. Figure 14). The partitioning of the data into four subsets provides additional information on the position of data items with respect to the variables assigned to the axes. Since the quadrants which correspond to the four subsets are not equally filled, the number of data items which may be visualized is slightly lower. In the example visualization provided in Figure 15a, the same data set is used as in Figure 13. Interesting

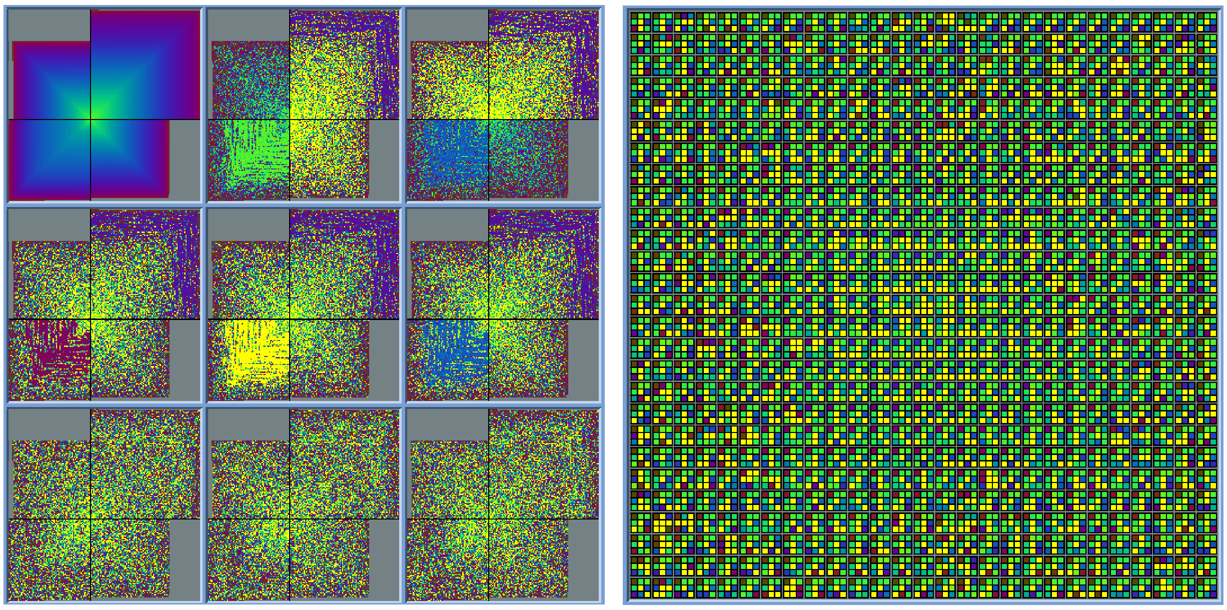


a. Axes Technique **b. Snake-Axes Technique**

Figure 14: Snake-Axes Technique

is that the partitioning of the data into four subsets already makes the clusters visible even in the case of using a snake-height of only one pixel (cf. Figure 15a). If the user has an idea which variable might be suitable to be assigned to the axes, the snake-axes technique has an advantage over the snake-spiral technique.

Instead of the snake-arrangement, other space-filling curves such as the Peano-Hilbert and Morton curves may be used locally on the rectangular spiral to achieve a better local clustering. A detailed description of possible variants (Hilbert-Spiral, Morton-Spiral, Axes-Hilbert, Axes-Morton, etc.) as well as an experimental comparison are provided in [Kei 95].



a. Snake-Axes Technique **b. Grouping Technique**

Figure 15: Visualizations of Eight-Variate Data (cont'd)

4. Grouping Arrangement

In both the spiral and axes techniques, the pixels corresponding to the k variables of one data item are distributed in k windows. In contrast, in the grouping technique all variables for one data item are grouped together in one area. The areas, each corresponding to one data item, may be arranged using any of the arrangements described in the previous two sections (Peano-Hilbert, Morton, Recursive-Pattern, Snake-Spiral, Snake-Axes). The visualizations generated using the grouping technique are completely different from those generated using the other techniques. The grouping visualizations consist of only one window with many small areas visualizing all variables of the considered data items instead of many windows, each providing a visual representation of only one variable. In Figure 16, we illustrate the grouping technique with underlying spiral arrangement. Our first practical experience shows that in contrast to the other techniques, the grouping technique requires multiple pixels per data value. For a single data value to be perceptible, at least 2x2 pixels are necessary. Additional pixels are needed for surrounding the areas corresponding to one data items. In contrast to the other techniques, a border is necessary since it would otherwise be impossible to know which pixels belong to which data item. As a consequence, the number of data items which may be visualized by the grouping technique is considerably smaller. In Figure 15b, an example for a grouping visualization is provided. The data set is the same as in Figure 13 and Figure 15a, but only 5% of the data items are visualized.

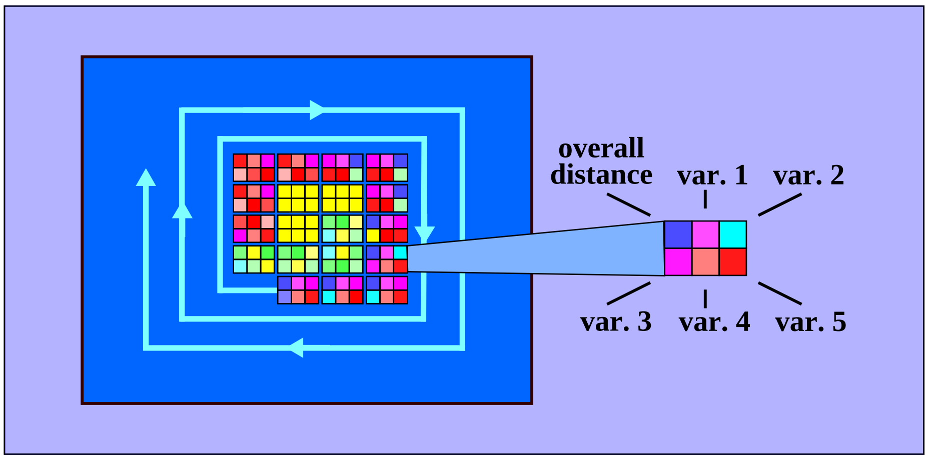


Figure 16: Grouping Technique of Five-Variate Data

5. Implementation

All techniques described in the previous subsections (except some of the grouping variants) are implemented as part of the *VisDB* system. In addition to our pixel-oriented techniques, the *VisDB* system also supports the parallel coordinates technique developed by Inselberg & Dimsdale [Ins 81, ID 90] at IBM and the stick figure technique developed by Picket & Grinstein at the University of Massachusetts, Lowell. All of our techniques adhere to the goal of presenting as many data items as possible on the display by allowing each data value to be presented by one pixel. The parallel coordinate and stick figure techniques require more than one pixel per data value, and in general, they also produce visualizations with overlapping data items. We therefore extended the parallel coordinate and stick figure techniques to be useful for visualizing and exploring large databases by coloring the stick figure icon and the line segments of the parallel coordinate technique by using the overall distance. Additionally, in case of overlapping stick figures or line segments, we draw the most relevant data items on top of the less relevant data. Drawing and coloring the data items according to their overall distances allow the most relevant data items to be easily located and compared, which is especially important in dealing with larger data volumes. Still, the parallel coordinate and stick figure techniques are only suitable for data sets with a limited number of data items.

We believe that, in general, multiple techniques have to be used in different stages of the data exploration process. First, the user might want to get an overview of the data and therefore start with a query-independent technique. If the data is unstructured or the structure of the data is unknown, the user will probably start with the Peano-Hilbert or Morton visualizations. If the structure is known, it might be better to directly use the recursive pattern technique with the appropriate parameter settings. If the search is more directed or if the user has already gained some hypotheses about the data, the user might switch to the query-dependent techniques. Again, the user will probably start with the most general technique, namely the snake-spiral technique. If the user has some idea about appropriate variables to be assigned to the axes, s/he will probably switch to the snake-axes technique which provides additional information about the data. Once the user has identified some interesting clusters or functional dependencies, s/he might use the grouping, parallel coordinate, or stick figure techniques for a focussed search on the identified portions of the data set. The *VisDB* system allows the users to choose the adequate technique depending on the data, the exploration task, and the stage of exploration. The users may arbitrarily switch between all techniques, allowing them to compare visualizations generated by different techniques and to find the most appropriate technique.

The *VisDB* system is implemented in C++/MOTIF and runs under X-Windows on HP 7xx machines. The system consists of an interactive interface which is divided into the visualization portion and the query specification portion (for the query-dependent techniques). The query specification portion provides a slider-based direct-interaction interface which allows an intuitive specification of queries [Kei 94, KK 94]. Different types of sliders are available for different data types. Other options which support the data exploration process are the possibility to focus on certain colors and the possibility to get the data values corresponding to a pixel of the display.

In implementing the system, special consideration has been given to two aspects — fast recalculation of the visualizations, which is crucial for allowing an interactive data exploration, and easy extensibility which is necessary for adapting the system to the needs of different application areas. Easy extensibility is achieved by implementing the system in a modular fashion, allowing user-defined distance and combinator functions as well as new display methods (e.g. new types of sliders and new visualization techniques) to be easily integrated. Interactivity is achieved by using efficient algorithms and adapting them for our application. The algorithms used in calculating the visualizations are all linear in the number of displayed data values (#data values = #data items * #variables). While this is clear for the query-independent techniques, it is not straightforward for the query-dependent technique. Calculating the query-dependent visualizations is a multi-step process which consists of loading the data, calculating the distances, normalizing and combining them, determining the desired percentage of data items with lowest overall distances, and sorting them according to their overall distance. For most steps in calculating our visualizations, a linear time complexity is straight-forward. The steps which do not naturally have a linear complexity are determining the desired percentage of data items with lowest overall distances (which corresponds to determining an $\frac{\text{\#displayed data items}}{\text{\#data items}}$-quantile) and sorting of the data according to the overall distance. Determining the desired percentage of data items with lowest overall distances is done with linear time-complexity by adapting the bottom-up heap algorithm [Weg 90] to our problem, and the sorting is done with linear complexity by using a bucket sort algorithm which is sufficient since the sorting granularity is limited to the number of colors. The details are beyond the scope of this paper and are presented in [Kei 94]. Note that a linear time complexity in the number of data values is the best we can achieve since the input of our techniques (database) and the output of our techniques (visualization) is in the same order of magnitude.

The current version of the *VisDB* system is main memory based and allows interactive query-dependent visualizations of very large databases. For databases with less than 100,000 data values, the recalculations can be considered to be truly interactive; for larger databases, the time is still in the range of a few seconds (for 1,000,000 data values, for example, the response time is about 20 seconds). When interfacing with current commercial database systems, however, performance problems arise since no access to partial results of a query is available, no support for incrementally changing queries is provided, and no multidimensional data structures are used for fast secondary storage access. We are currently working on improving the performance in directly interfacing to a database system. In the future, we plan to implement the *VisDB* system on a parallel machine which will be able to support interactive query modifications even for larger amounts of data.

The *VisDB* system has been successfully used in several application areas including a financial application where the system has been used to analyze multivariate time-dependent data, a CAD database project where the system has been used to improve the similarity search, as well as a molecular biology project where the system has been used to find possible docking regions by identifying sets of surface points with distinct characteristics [Kei 94]. Currently, we explore several other data sets including a large database of geographical data, a large environmental database, and a NASA earth observation database.

6. Conclusions

Pixel-oriented visualization techniques which use each pixel of the display to visualize one data value provide a valuable help in exploring very large databases. The techniques described in this paper allow users to get a visual overview of large data sets and supports them in finding correlations, functional dependencies, and clusters. Our query-independent techniques directly visualize the variable values by arranging the values according to some screen-filing curve. In addition, the recursive pattern technique allows the user to control the arrangement of the data values, providing the possibility to generate more meaningful visualizations. Our query-dependent techniques visualize the data variables in the context of a specific query and provide visual feedback in querying the database. The techniques are especially helpful for interactively exploring large databases. The grouping techniques combine the separate windows for the variables and group all variable values corresponding to one data item into one area. For perceptual reasons, the number of data values that can be presented by the grouping technique is lower and therefore, the grouping technique is mainly suitable for a focussed search on smaller data sets. We believe that the different techniques (query-dependent — query-independent / partitioned — grouping) are useful for different data exploration tasks and for different stages of the data exploration process.

At this point, we want to stress that our visualization techniques are not designed to replace or substitute current statistical methods for visualizing multivariate data. Also, we do not claim that our techniques are in general better than statistical methods such as correlation or regression analysis. Both data visualization and multivariate statistics have their advantages and we view them as being complementary to each other. Statistical analysis may, for example, be used to validate the hypotheses generated by the visualizations and vice versa. Integrated tools for exploratory data analysis should therefore include not only statistical methods and scatter diagram representations of the data but also other data visualization techniques such as our pixel-oriented visualizations.

Inspired by using our prototype, we already have several ideas to extend our system. One idea is the automatic generation of time series of visualizations which correspond to incrementally changing queries. By changing the query, different portions of multidimensional space can be visualized, allowing even larger amounts of data to be displayed. We also plan to apply our techniques in different application domains, each having its own parameters, distance functions, query requirements and so on. This will help us to evaluate the strength and weaknesses of our techniques and to further improve the techniques. We also intend to evaluate our visualization techniques by using artificially generated data sets which allow controlled studies of their possibilities and limits.

Acknowledgments

Implementing a complex system such as the *VisDB* system can not be done by a single person. My thank goes to all my colleagues and students who contributed to the *VisDB* system, especially Thomas Seidl who implemented the first prototype of the system, Juraj Porada who implemented most of the current version, Mihael Ankerst who implemented the recursive pattern technique, and Professor Dr. Kriegel who provided the inspiring environment for doing the research reported in this paper.

References

[AC 91] Alpern B., Carter L.: *'Hyperbox'*, Visualization '91, San Diego, CA, 1991, pp. 133-139.

[ADLP 95] Anupam V., Dar S., Leibfried T., Petajan E.: *'DataSpace: 3-D Visualization of Large Databases'*, Proc. Int. Symposium on Information Visualization, Atlanta, GA, 1995.

[AG 80] Alexandrov V. V., Grosky N. D.: *'Recursive Approach to Associative Storage and Search of Information in Data Bases'*, Proc. Finnish-Soviet Symposium on Design and Application of Data Base Systems, Turku, Finland, 1980, pp. 271-284.

[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 (CHI’94), 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 95] Ahlberg C., Wistrand E.: *‘IVEE: An Information Visualization and Exploration Environment’*, Proc. Int. Symposium on Information Visualization, Atlanta, GA, 1995, pp. 66-73.

[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 (CHI’92), Monterey, CA, 1992, pp. 619-626.

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

[BCW 88] Becker R., Chambers J. M., Wilks A. R.: *‘The New S Language’*, Wadsworth & Brooks/Cole Advanced Books and Software, Pacific Grove, CA., 1988.

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

[BMMS 91] Buja A., McDonald J. A., Michalak J., Stuetzle W.: *‘Interactive Data Visualization Using Focusing and Linking’*, Visualization ‘91, San Diego, CA, 1991, pp. 156-163.

[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.

[FB 94] Furnas G. W., Buja A.: *‘Prosections Views: Dimensional Inference through Sections and Projections’*, Journal of Computational and Graphical Statistics, Vol. 3, No. 4, 1994, pp. 323-353.

[FDFH 90] Foley J. D., van Dam A., Feiner S. K., Hughes J. F.: *‘Computer Graphics: Principles and Practice’*, 2nd Edition, Addison-Wesley, Reading, 1990.

[Gol 81] Goldschlager L. M.: *‘Short Algorithms for Space-filling Curves’*, Software Practive and Experience, Vol. 11, p. 99.

[GPW 89] Grinstein G, Pickett R., Williams M. G.: *‘EXVIS: An Exploratory Visualization Environment’*, Proc. Graphics Interface ‘89, London, Ontario, Canada, 1989.

[Hil 91] Hilbert D.: *‘Über stetige Abbildung einer Line auf ein Flächenstück’*, Math. Annalen, Vol. 38, 1891, pp. 459-460.

[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.

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

[Kei 95] Keim D. A.: *'Enhancing the Visual Clustering of Query-dependent Databases Visualization Techniques using Screen-Filling Curves'*, Proc. Int. Workshop on Database Issues in Visualization, Atlanta, GA, 1995.

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

[KK 95] Keim D. A., Kriegel H.-P.: *'Issues in Visualizing Large Databases'*, Proc. Conf. on Visual Database Systems (VDB-3), Lausanne, Schweiz, März 1995, in: Visual Database Systems, Chapman & Hall Ltd., 1995, pp. 203-214.

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

[Mor 66] Morton .: *'A Computer Oriented Geodetic Data Base and a New Technique in File Sequencing'*, IBM Ltd. Ottawa, Canada, 1966.

[MW 95] Martin A. R., Ward M. O.: *'High Dimensional Brushing for Interactive Exploration of Multivariate Data'*, Visualization '95, Altanta, GA, 1995, pp. 271-278.

[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.

[Pea 90] Peano G.: *'Sur une courbe qui remplit toute une aire plaine'*, Math. Annalen, Vol. 36, 1890, pp. 157-160.

[PG 88] Pickett R. M., Grinstein G. G.: *'Iconographic Displays for Visualizing Multidimensional Data'*, Proc. IEEE Conf. on Systems, Man and Cybernetics, IEEE Press, Piscataway, NJ, 1988, pp. 514-519.

[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.

[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.

[Shn 94] Shneiderman B.: *'Dynamic Queries for Visual Information Seeking'*, IEEE Software, Vol. 11, 1994, pp. 70-77.

[RCM 91] Robertson G., Card S., Mackinlay J.: *'Cone Trees: Animated 3D Visualizations of Hierarchical Information'*, Proc. ACM CHI Int. Conf. on Human Factors in Computing (CHI'91), pp. 189-194.

[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.

[Vel 92] Velleman P. F: *'Data Desk 4.2: Data Description'*, Ithaca, NY, 1992.

[Ward 94] Ward M. O.: *'XmdvTool: Integrating Multiple Methods for Visualizing Multivariate Data'*, Visualization '94, Washington, DC, 1994, pp. 326-336.

[Weg 90] Wegener I.: *'Bekannte Sortierverfahren und eine HEAPSORT-Variante, die QUICKSORT schlägt'*, Informatik Spektrum, Vol. 13, No. 6, 1990, pp. 321-330.

[WL 93] van Wijk J. J., van Liere R.. D.: *'Hyperslice'*, Visualization '93, San Jose, CA, 1993, pp. 119-125.

[WUT 95] Wilhelm A., Unwin A.R., Theus M.: *'Software for Interactive Statistical Graphics - A Review'*, Proc. Int. Softstat '95 Conf., Heidelberg, Germany, 1995.