IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 6, Dec. 1996.

Visualization Techniques for Mining Large Databases: A Comparison

Daniel A. Keim, Hans-Peter Kriegel

Abstract

Visual data mining techniques have proven to be of high value in exploratory data analysis and they also have a high potential for mining large databases. In this article, we describe and evaluate a new visualization-based approach to mining large databases. The basic idea of our visual data mining techniques is to represent as many data items as possible on the screen at the same time by mapping each data value to a pixel of the screen and arranging the pixels adequately. The major goal of this article is to evaluate our visual data mining techniques and to compare them to other well-known visualization techniques for multidimensional data: the parallel coordinate and stick figure visualization techniques. For the evaluation of visual data mining techniques, in the first place the perception of properties of the data counts, and only in the second place the CPU time and the number of secondary storage accesses are important. In addition to testing the visualization techniques using real data, we developed a testing environment for database visualizations similar to the benchmark approach used for comparing the performance of database systems. The testing environment allows the generation of test data sets with predefined data characteristics which are important for comparing the perceptual abilities of visual data mining techniques.

Keywords: Data Mining, Explorative Data Analysis, Visualizing Large Databases, Visualizing Multidimensional and Multivariate Data

1. Introduction

Having the right information at the right time is crucial for making the right decisions. Because of the fast technological progress, the amount of information which may be of interest for making decisions increases very fast. One reason for the ever increasing stream of data is the automation of activities in all areas, including business, engineering, science, and government. Today, even simple transactions, such as paying by credit card or using the telephone, are typically recorded by using computers. Test series in physics, chemistry, and medicine generate large amounts of data which are collected automatically via sensors and monitoring systems. Even larger amounts of data are collected by satellite observation systems which are expected to generate one terabyte of data every day in the near future. But finding the valuable information hidden in them, is like searching a pin in a haystack. Very large amounts of data are an important resource, but most of the time it is very hard to find the relevant information.

'Data Mining' may be defined as the (non-trivial) process of searching and analyzing data in order to find implicit but potentially useful information [1]. Let $D = \{d_1, ..., d_n\}$ be the data set to be analyzed. Then, the data mining process may be described as the process of finding

- a subset D' of D and
- hypotheses H_U (D', C) about D'

that a user U considers useful in an application context C.

Note that D' may not only have fewer data elements than D, but it may also have a lower dimensionality (m'). Since in databases the data is often partitioned into relations or object classes, D may be considered as a union of relations $R_1, ..., R_k$ ($D = \bigcup_{i=1}^{k} R_i$), each having its own dimensionality ($m_1, ..., m_k$). The hypotheses expressing interesting aspects of the data may deal with the whole database or with a single relation (D' = D or D' = R_i); they may deal with real subsets of the database ($D' \subset D$ with $|D'| \ll |D|$ and $|D'|$ sufficiently large) or with single exceptional data items, so-called hot spots ($D' \subset D$ and $|D'| = 1$ or sufficiently small when compared to $|D|$). Among others, hypotheses may be

- properties that hold for all or most $e_i \in$ D', ($D' \subseteq D$),
- classifications of D' into classes C_i with different properties P_i
  [$P_i(e_1) \neq P_j(e_2) \Rightarrow e_1 \in C_i \wedge e_2 \in C_j \wedge i \neq j$],
- functional dependencies F or relationships R between two or more dimensions
  [$d_{i1} = F(d_{i2}, ..., d_{il})$ or $R(d_{i1}, ..., d_{il})$, $1 \leq m$].

The definition of data mining can be further formalized, e.g. by defining a hypothesis description language, a context description formalism and so on. The users and their notion of 'usefulness', however, can hardly be formalized since 'usefulness' not only depends on the changing knowledge of the user and the application domain, but it also includes some notion of creativity and users may not be able to define their usefulness criteria. On the other hand, if a data mining tool helps the user to find useful D' and to find and verify hypotheses, then it may not be important to have the hypothesis, the context and so on formally specified. All these aspects are present in the users' minds who will also be able to express and communicate their ideas towards other humans.

Our definition of data mining is a quite broad definition which does not only include the work done in the area of data mining and knowledge discovery, but also relates to a wide range of other research areas including multivariate statistics (principal component analysis, cluster analysis, and multidimensional scaling [2]), database interfaces (cooperative database interfaces [3], interfaces for imprecise querying [4], intelligent data browsing [5]), and information retrieval (approximate matching algorithms [6] [7]). The work done in data mining focuses on the semi-automatic extraction of knowledge. In all mentioned areas, important advances have been made over the last years. Many novel data mining techniques have been developed and several advanced data mining systems have been implemented [1] [8]. Nowadays, however, only a limited number of approaches work for very large amounts of data (millions of data items) and little interest has been given to noisy data [8]. Examples for techniques that work for very large data sets are DHP [9], Apriori [10], and DBLearn [11], and examples for techniques that also work for noisy data are DBLearn [11] and CLARANS [12].

An interesting observation is that all mentioned techniques work fully automatically but need to have a-priori defined tasks. The tasks are a specific type of hypothesis and the goal of the algorithms is to find quantitative rules that make the hypotheses more specific and allow the user to confirm or reject them. Task-oriented data mining is important but it is also important to develop techniques for data-driven hypotheses generation. For this purpose, it is necessary to include the human in the data mining process and combine the flexibility, creativity, and general knowledge of the human with the enormous storage capacity and the computational power of today's computers. In particular, the human's unmatched abilities of perception enable the users to analyze complex events within a short time, to recognize important information, and to make decisions. The human perceptual system processes different types of data in a very flexible way, automatically recognizing unusual properties while at the same time ignoring well-known properties. The human handles vague descriptions and imprecise knowledge easier and better than today's computer systems and, using general knowledge, easily draws complex conclusions.

Our approach to data mining therefore aims at integrating the human in the data mining process and applying its abilities to the large data sets available in today's computer systems. For this purpose, techniques which provide a good overview of the data and use the possibilities of visual representation for displaying large amounts of multidimensional data are especially important. The basic idea of our new visual data mining techniques for multidimensional data is to represent as many data items as possible on the display at the same time by mapping each data value to one pixel of the screen and arranging the pixels adequately. The color of the pixel corresponds to the data value or the distance between the data value and a given query value. Different visual data mining techniques are available for the different stages of the data mining process. In using our visual data mining techniques, the possibility to directly interact with the visualizations is important. In the process of hypotheses generation, the user is guided by the visual feedback of the visualizations and quickly learns more about the properties of the data in the database.

Since the reader is not assumed to be familiar with visual data mining techniques, in section 2 we give a brief general survey of visualization techniques for multidimensional multivariate data. We classify the existing techniques into five groups: pixel-oriented, geometric, icon-based, hierarchical, and graph-based techniques. In section 3, we provide a detailed evaluation and comparison of several visual data mining techniques including pixel-oriented, geometric and icon-based techniques. In addition to testing the techniques using real data (cf. subsection 3.1), we developed a testing environment for database visualizations similar to the benchmark approach used for comparing the performance of databases (cf. subsections 3.2 and 3.3). For the evaluation of visual data mining techniques, the perception of properties and correlations of the data is more important than the CPU times or the number of secondary storage accesses. Still, the interactivity of the system is essential and therefore, in section 4, we analyze the time performance of our algorithms. Section 5 summarizes our work and points out some of the open problems for future work.

For our considerations, we assume a simple structure of the database as we may find it in the relational model. This is adequate for most of the considered applications, because very large amounts of data are typically managed with the aid of relational systems. Our visual data mining techniques, however, can also be used for visually mining large amounts of data stored in object-oriented or other types of databases.

2. Techniques for Visualizing Large Amounts of Multidimensional 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 [13] [14], 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 multidimensional data sets with the attributes of the database corresponding to the dimensions. There are several well-known techniques for visualizing multidimensional data sets: scatterplot matrices and coplots [15] [16], prosection matrices [17], parallel coordinates [18] [19], projection pursuit [20], and other geometric projection techniques (e.g., hyperbox [21] and hyperslice [22]), icon-based techniques (e.g., [23] [24]), hierarchical techniques (e.g., [25] [26] [27]), graph-based techniques (e.g., [28] [29] [30]), dynamic techniques (e.g., [31] [32] [33]), pixel-oriented techniques (e.g., [34] [35] [36]), and combinations hereof (e.g., [37] [38]). 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 [39], XGobi [40], and Data Desk [41], visualization oriented systems such as ExVis [42], XmdvTool [42], and IBM's Parallel Visual Explorer, as well as database oriented systems such as TreeViz [27], the Information Visualization and Exploration Environment (IVEE)[44], and the *VisDB* system [45]. In the following, we briefly classify and describe some important techniques which are suitable for visually mining large databases.

2.1 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 attribute 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 attribute(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 attribute (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 attributes 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.

All pixel-oriented techniques partition the screen into multiple windows. For data sets with m attributes (dimensions), the screen is partitioned into m windows — one for each of the attributes. 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 attributes may be detected by relating corresponding regions in the multiple windows.

Query-Independent Pixel-oriented Techniques

Simple query-independent arrangements are to arrange the data from left to right in a line-by-line fashion or top-down in a column-by-column fashion. 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 such as space-filling curves (e.g., the well-known curves by Peano & Hilbert [46] [47] and Morton [48]). For data mining even more important are techniques that provide nice clustering properties as well as an arrangement which is semantically meaningful. An example for a technique which has these properties is the recursive pattern technique. The recursive pattern is based on a generic recursive scheme which allows the user to influence the arrangement of data items.

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 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. On recursion level i ($i \geq 1$), the algorithm draws w_i level(i-1)-patterns h_i times alternately to the right and to the left. The pattern on recursion level i consists of $w_i \times h_i$ level(i-1)-patterns, and the maximum number of pixels that can be presented on recursion level k is given by $\prod_{i=1}^{k} w_i \times h_i$. An example for a recursive pattern visualization of a database containing the 100 stocks of the FAZ index (Frankfurt Stock Index) from 20 years of stock price data (altogether 532,900 data values) can be found in [36].

Query-Dependent Pixel-oriented Techniques

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 attribute values to colors, the distances of attribute values to the query are mapped to colors. To describe the idea of the query-dependent techniques, we view the relations of a relational database as sets of tuples $(a_1, a_2, ..., a_k)$ with $a_1, a_2, ..., a_k$ denoting the attribute values of a data item. Simple queries against the database can be described as regions in the k-dimensional space defined by the k attributes of the relation. If exactly one query value is specified for each attribute, the query corresponds to a point in k-dimensional space; if a query range is specified for each attribute, the query corresponds to a region in k-dimensional space. The data items which are within the query region form the result of the query. In most cases, the number of results cannot be determined a priori; the resulting data set may be quite large, or it may even be empty. In both cases, it is difficult for the user to understand the result and modify the query accordingly. To give the user more feedback on the query, our visual data mining techniques do not only present the data items which are within the query region, but also those which are 'close' to the query region and only approximately fulfill the query. For determining the approximate results, distances between the data and query values are calculated. The distance functions are data type and application dependent. For numeric types such as *integer* or *real* and other metric types such as *date,* the distance of two values is easily determined by their numerical difference. For other types such as *strings*, multiple distance functions such as the lexicographical difference, character-wise difference, substring difference, or even some kind of phonetic difference may be useful. The distance calculation yields distance tuples $(d_1, d_2, ..., d_k)$ which denote the distances of the data to the query. We extend the distance tuples by a distance value d_{k+1}, denoting the overall distance of a data item to the query. The value of d_{k+1} is zero if the data item is within the query region; otherwise d_{k+1} provides the distance of the data item to the query region. In combining the distance values $(d_1, d_2, ..., d_k)$ into the overall distance value d_{k+1}, user-provided weighting factors $(w_1, w_2, ..., w_k)$ are used to weight the distance values according to their importance. The distance tuples $(d_1, d_2, ..., d_k, d_{k+1})$ are sorted according to the overall distance d_{k+1}. Then the distance tuples are mapped to color. In this step, the value ranges for each of the attributes and for the overall distance are mapped to a colorscale which has been specifically designed for our visual data mining techniques. Note that the human visual system has a non linear response to luminance and spectral content. Incorrect use of color can hide existing relations between variables, and introduce artifacts. It is therefore important to use a colorscale which is perceptually equally spaced [49]. Our colorscale uses yellow to depict the distance 'zero' and a decreasing lightness to depict increasing distance values. The colors for approximate results range from green over blue and red to almost black. For details about our color mapping, the reader is referred to [50].

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 are 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 results 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 [35] [51] was to arrange the data items with lower relevances in a rectangular spiral shape around the center. The generalized spiral and axes techniques presented in this paper are a generalization of those techniques. Instead of arranging the data in a rectangular spiral shape, the curve is extended to a generic spiral form which may have a Snake-, Peano-Hilbert-, or Morton-like local pattern (cf. Figure 2) of a certain degree (1, 2, 4, 8, 16). The advantage of the generalized spiral and axes techniques is that the degree of clustering is higher. In case of the generalized spiral technique, the one hundred percent correct answers are presented in the middle of the window and the approximate answers sorted according to their overall distance (or relevance) in a generalized spiral form around this region. As for the query-independent visualization techniques, a separate visualization for each of the selection predicates (attributes) 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 is located. 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; for example, on the restrictiveness of each of the selection predicates and on single exceptional data items. Examples of spiral visualizations are provided in section 3. The axes technique improves the spiral technique by including some feedback on the direction of the distance into the visualization. The basic idea is to assign two attributes to the axes and to arrange the data items according to the direction of the distance; for one attribute negative distances are arranged to the left, positive ones to the right and for the other attribute negative distances are arranged to the bottom, positive ones to the top (cf. Figure 3). As in case of the spiral, different local patterns (Snake, Peano-Hilbert, Morton) of different degree (1, 2, 4, 8, 16) may be used. The partitioning of the data into four subsets provides additional information on the position of data items with respect to the attributes 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. This, however, is the price for the higher expressiveness of the resulting visualizations. An example of an axes visualization is provided in subsection 3.1.

Note that all variants (Snake, Peano-Hilbert, Morton) reduce to a simple spiral for a degree of one. A degree of one means that the local pattern consists of only 1x1 = 1 pixel, and in this case the original spiral and axes techniques [35] [51] are identical to the generalized techniques. A detailed comparison of the possible variants (Snake-Spiral, Peano-Hilbert-Spiral, Morton-Spiral, Snake-Axes, Peano-Hilbert-Axes, Morton-Axes) with different degrees (1, 2, 4, 8, 16) is provided in [52]. The formulas for calculating the distances and their combination into the overall distance as well as all aspects related to the handling of complex queries (conditions with nested ANDs and ORs, multiple table and nested queries) are presented in [35]. The focus of this paper is on the data mining capabilities of the various visualization techniques.

2.2 Geometric Projection Techniques

Geometric projection techniques aim at finding ‘interesting’ projections of multidimensional data sets. The class of geometric projection techniques includes techniques of exploratory statistics such as principal component analysis, factor analysis and multidimensional scaling, many of which are subsumed under the term ‘projection pursuit’ [20] [53]. Since there is an infinite number of possibilities to project high-dimensional data onto the two display dimensions, ‘projection pursuit’ systems such as the grand tour system [37] aim at automatically finding the interesting projections or at least helping the user to find them.

Another geometric projection technique is the parallel coordinate visualization technique [18] [19]. The parallel coordinate technique maps the k-dimensional space onto the two display dimensions by using k equidistant axes which are parallel to one of the display axes. The axes correspond to the dimensions and are linearly scaled from the minimum to the maximum value of the corresponding dimension. Each data item is presented as a polygonal line, intersecting each of the axes at that point which corresponds to the value of the considered dimension (cf. Figure 1a). Although the principle idea of the parallel coordinate visualization technique is quite simple, it is powerful in revealing a wide range of data characteristics such as different data distributions and functional dependencies. However, since the polygonal lines may overlap, the number of the data items that can be visualized on the screen at the same time is limited to about 1,000 data items. In Figure 4b, an example visualization of three-dimensional data is presented. Clearly visible in the visualization is that the data consists of several clusters which are restricted to quite limited ranges for the second dimension but may have much larger ranges for the other dimensions. The parallel coordinate technique is used in the comparison of subsection 3.3. The reader is therefore referred to subsection 3.3 for more details and further examples.

2.3 Icon-based Techniques

Another class of techniques for visual data mining are the icon-based techniques (or iconic display techniques). The idea is to map each multidimensional data item to an icon. First approaches of iconic displays are the well-known Chernoff faces [13] [54]. In the Chernoff face visualization, two dimensions are mapped to the two display dimensions. The remaining dimensions are mapped to the properties of a face icon — the shape of nose, mouth, eyes, and the shape of the face itself. The Chernoff face visualization capitalizes on the human sensitivity to faces and facial features. The number of data items that can be visualized using the Chernoff face technique, however, is quite limited.

An iconic display technique, which allows a visualization of larger amounts of data and is therefore more adequate for data mining, is the stick figure technique [23] [55] [56]. As indicated by the name, the icon is some type of stick figure. Again, two dimensions are mapped to the display dimensions and the remaining dimensions are mapped to the angles and/or limb lengths of the stick figure icon (cf. Figure 5a). If the data items are relatively dense with respect to the display dimensions, the resulting visualization presents texture patterns that vary according to the characteristics of the data and are therefore detectable by preattentive perception. Different stick figure icons with variable dimensionality may be used (cf. Figure 5b). Figure 6 shows a stick figure visualization of five-dimensional census data of 1980 US census. In addition to income and age, the attributes occupation, education level, marital status, and sex are visualized by the stick figures. Interesting is the clear shift in texture over the screen which indicates the functional dependency of the attributes from income and age. More examples of the stick figure visualizations are provided in the comparison of subsection 3.3. Note that in both, the stick figure and the Chernoff face technique, the number of dimensions that can be visualized is limited.

Many other ideas for iconic displays have been developed in recent years. An approach which allows the visualization of an arbitrary number of dimensions is the shape-coding approach [24]. The icon used in the shape coding approach maps each dimension to a small array of pixels and arranges the pixel arrays of each data item into a square or rectangle. The pixels corresponding to each of the dimensions are mapped to grey scale or color according to the data value of the dimension. The small squares or rectangles corresponding to the data items are then arranged successively in a line-by-line fashion.

2.4 Hierarchical and Graph-based Techniques

In addition to the geometric projections and iconic displays, there are two more classes of visualization techniques - hierarchical and graph-based techniques. Well-known representatives of hierarchical techniques are the n-Vision technique (also known as 'worlds within worlds') [57], the dimensional stacking [25], and treemaps [27].

The hierarchical techniques subdivide the k-dimensional space and present the subspaces in a hierarchical fashion. The dimensional stacking technique, for example, subdivides the k-dimensional space into 2D-subspaces. With the exception of treemap, the hierarchical techniques mainly focus on visualizing multivariate functions and are therefore not particularly interesting for data mining. The basic idea of the graph-based techniques is to effectively present a large graph using specific layout algorithms, query languages, and abstraction techniques. Examples of graph-based techniques are Hy^{+} [28], Margritte [29], and SeeNet [30].

3. Evaluation and Comparison of Visual Data Mining Techniques

A central goal of this article is to evaluate and compare the techniques which may be used for visualizing large databases. An evaluation of visualizations is different from evaluating the time performance of a system. In principle, measuring the time performance of a system is relatively easy compared to measuring the expressiveness of visualizations, i.e. the perceptibility of data characteristics. The reason is that evaluating the perception necessarily involves humans, and therefore the results do not only depend on the potential of the visual data mining technique but also on the human performing the analysis.

Before presenting our evaluations, in the following we briefly introduce how data characteristics such as correlations, functional dependencies, and clusters may be identified in our visualizations generated by the generalized spiral and axes techniques. The following description may be used as a guideline for discovering data characteristics by interpreting the visualizations generated by the *VisDB* system. Note, that many of the data mining techniques we developed in working with the *VisDB* system are interactive in nature and are therefore difficult to describe in written form. Nevertheless, in the following we try to present the basic ideas.

Properties that hold for all or most of the data can be deduced from the overall brightness and color distribution of the visualizations. The size of the yellow portion of the visualization indicates the number of data items fulfilling the query predicate for the corresponding attribute. The brightness of the visualization of some attribute indicates the degree of fulfilling the corresponding query predicate, and the overall color distribution shows the distribution of distance values for the corresponding attribute. Interesting are especially sharp borders between colors, which indicate discontinuities in the value range of an attribute.

Usually the visualizations of the attributes are not independent. Using our visual data mining techniques, correlations and functional dependencies between attributes may be identified by the similarity of their visualizations. The more the visualizations of two attributes are similar, the stronger is the correlation or functional dependency. An example is provided in Figure 7 (Footnote 1). From the similarity of the visualization windows, it becomes clear that there is a strong correlation between the attributes MinAngle and RightAngle. Slightly less similar and therefore less correlated are MinAngle and MidAngle, and even weaker is the correlation between MinAngle and MaxAngle. A bit problematic are visualizations that do not show any structure. The human perceptual system may consider such visualizations as being similar although the pixels of the visualizations do not correspond to each other and no correlation between the attributes exists. In experimenting with various data sets, we found that the problem of misleading similarity of visualizations only occurs in visualizations without structure. In cases where existing structure in the data is not revealed by the visualization, the structure can usually be made visible by changing the weighting factors of some attributes. Changing the weighting factors makes it easier to perceive similarities in the visualizations and to determine the corresponding correlations and functional dependencies (cf. Figure 8b).

Another important group of interesting data characteristics are clusters of data items with similar properties. In our visualizations, clusters usually appear as rectangular regions (possibly partial rectangles) which may have different colors for different attributes. Clusters may have a lower dimensionality than the whole data set. Therefore, the clustering may appear only in the visualizations of certain attributes. Figure 8 shows an example of four-dimensional clusters in six-dimensional data.

Note that the visualizations generated by the generalized spiral and axes techniques in general depend on the chosen query region. If the data shall be visualized in a query-independent way, the origin of multidimensional space has to be used as a query region. In this case, only the actual data values are visualized. Changing the query region has a major impact on the resulting visualizations. For example, if the query region is moved away from a cluster, the cluster becomes less visible in the visualization (cf. Figure 9). By interactively changing the query region, different clusters can be made visible. A strategy for varying the query region which has proven useful in our experiments is to start with the full range [min, max] for an attribute and successively restrict the range until the visualizations reveal some interesting properties. In some cases, the inverse of this technique — starting at the minimum value for some attribute and extending the range to the maximum (or vice versa) — is more appropriate. Another possibility is to use a range with a constant extension for some attribute and move it between minimum and maximum value of the attribute. This technique is especially helpful for finding the value range of an attribute, which corresponds to an already discovered cluster in some other dimensions - if such a range exists.

In the remaining portion of this section, we take a twofold approach towards evaluating the perception of visual data mining techniques. The first, more conventional approach is to demonstrate the potential of visualization techniques by using a real application (cf. subsection 3.1). The second, more general approach is a step towards a more application-independent evaluation of visualization techniques. By using test data sets which are generated according to user-provided specifications (cf. subsection 3.2), it becomes possible to perform controlled tests for evaluating the strengths and weaknesses of visualization techniques.

3.1 Real Data

The *VisDB* system has proven useful for exploration tasks in several real databases including a large database of geographical data, a large environmental database, and a large NASA earth observation database. The *VisDB* system has also been used in our molecular biology project for finding possible docking regions by identifying sets of surface points with distinct characteristics, and we will use this example to exemplify the usefulness of the system.

In our strategy for finding possible docking regions, one important step is to partition the molecular surfaces according to geometric properties. For this purpose, we use the properties of the triangulation of the molecular surface, e.g. minimum and maximum angle, minimum and maximum side length, right angle and so on. In partitioning the molecules into regions, it is difficult to find the right combination of value ranges that leads to a meaningful partitioning of the molecule. The *VisDB* system has been successfully used for determining interesting value ranges and combinations of parameters. The *VisDB* system allows the interactive variation of the parameters and provides feedback on the resulting data in the region. In Figure 7, we provide two snapshots from a session with the *VisDB* system. The snapshot shows eleven parameters (X_triangle, Y_triangle, Z_triangle, MinAngle, MidAngle, MaxAngle, RightAngle, Surface, MinSide, MidSide, MaxSide) and the overall result generated from the surface data of the molecule complex 'subtilisin carlsberg with eglin'. Since data exploration is inherently interactive, the provided snapshots reflect the usefulness of the *VisDB* system only to a very limited extent. In the following, we briefly describe some of the interesting aspects that we learned from the visualization.

By the different colors dominating the different portions of the visualization, it can be easily seen that each of the selection predicates has a distinct impact on the overall result. The visualization of MaxAngle, for example, is dominated by darker colors, which means that in general the distances to the specified range for MaxAngle (80°-153°) are quite high. The visualizations of Y_triangle and Z_triangle are much brighter, which means that the specified query ranges for these attributes are less restrictive. Interesting is the partitioning which results in the case of the axes technique (Surface and MaxSide are assigned to the axes). For example, the data items which have high distances (dark colors) for MinSide, MidSide, and MaxSide cluster in the bottom left quadrant, which means that their Surface and MaxSide are smaller than the chosen query ranges. Also easily visible is the correlation between the distances for MinSide, MidSide, and MaxSide, which corresponds to the Pythagorean theorem $(a^2 + b^2 = c^2)$. Although the quantitative relationship between MinSide, MidSide, and MaxSide is not deductible from the visualization, the similarity of the visualizations indicates that there is some relationship. More interesting, however, are data-dependent correlations which may not be predicted a priori. An example is the correlation between the data items that fulfill all query ranges, resulting in the yellow region in the middle of the visualization. The region in the visualization corresponds to surface regions of the molecule, which consist of triangles with similar properties, i.e. a certain combination of value ranges for the parameters.

If the same triangles are visualized in the 3D-representation of the molecule, the triangles form a double-horn region. Other interesting regions found in the molecular biology application include several types of domes, arcades, caves, and tripods (for details see [51]). The 3D-representation of molecules is very important (e.g., for visualizing the final result), but its usefulness for exploring more than 5 dimensions (3 dimensions of the coordinate system plus one or two additional dimensions denoted by color) is quite restricted. In contrast, our visual data mining techniques are able to present a virtually arbitrary number of parameters, allowing an easy discovery of correlations between them and avoiding the possibly confusing 3D-representation which may occur in case of large molecules due to the high overlap of surface points.

Note that for determining the properties that lead to interesting regions of the molecule, the interactivity of the *VisDB* system is very important. Besides the options which provide values for specific data items and color ranges (for details see [51]), the user may get important additional information by varying the query ranges and weighting factors using the sliders.

3.2 Artificial Data

Many visualization techniques and their potential have been demonstrated by using data from certain application areas. Lacking in all this activity is any quantitative evidence of how effective the techniques are. To get beyond the current demonstrational stage, a basis for evaluation needs to be developed. To progress, we need to know with certainty what is working and which adjustments are leading to an improvement.

Evaluations of the time performance are usually done by using canonical test data sets and standardized testing procedures, so-called benchmarks. Instead of establishing a fixed test data set for evaluating the perception, we use a general test data generation model which can be used for a standardized and quantitative testing of visualization techniques [58]. The model allows the generation of test data sets with characteristics similar to those of real data sets. Unlike real data sets, however, the characteristics of the artificially generated data sets may be varied arbitrarily. We may, for example, vary the correlation coefficient of two dimensions, the mean and variance of some of the dimensions, the location, size, and shape of clusters, etc. Varying the data characteristics in a controlled manner is crucial for evaluating the perceptibility of data characteristics in the visualizations. Controlled test series allow, for example, finding the point where certain data characteristics are perceptible for the first time, or the point where they are no longer perceptible.

The *TestVis* test data generation tool developed at the University of Munich partially implements the test data generation model described in [58] and allows the specification of a wide range of data characteristics. The main focus is on database like test data sets which in general are best described by statistical parameters such as distributions, correlations, functional dependencies, and clusters. The *TestVis* tool allows the specification of data with an arbitrary dimensionality and an arbitrary number of clusters. For each dimension, the user may specify the distribution function or the functional dependency. Also the size and shape of the clusters as well as the properties for each of the cluster dimensions can be specified. The functional dependencies that may be specified using the *TestVis* tool are of the form

$$a_j = (1 + r \times f) \times \left(\sum_{i=1}^{j-1} c_{i1} \times a_i^{c_{i2}} \right)$$

with

- a_j being the functional dependent dimension
- a_i being the data dimensions on which the dimension a_j is functionally dependent,
- r being a randomly generated variable in the range of [-1,1],
- c_{i1} and c_{i2} being user-specified constants, and
- f being a user-specified factor which induces a certain randomness of the data.

To avoid cyclic dependencies, without loss of generality, we assume that dimension a_j only depends on dimensions $a_1, \ldots, a_{j-1}$. A detailed description of the *TestVis* system can be found in [51].

In the following, we provide examples for generated test data sets and the corresponding visualizations. First, we evaluate the perceptibility of data characteristics in visualizations generated by our techniques. Due to space limitations, in this step we only use visualizations generated by using the generalized spiral technique of degree 1. Then, we compare our techniques to the parallel coordinate and stick figure visualization techniques which have been introduced in section 2. In general, the test data sets used in this article consist of a large randomly generated base data set which makes up about two thirds of the data, and one or multiple clusters which have a different dimensionality or are defined by using different distributions and functional dependencies.

Cluster with Different Dimensionality

In Figure 8, we present visualizations of four-dimensional clusters in six-dimensional data. The data set used to generate the visualizations consists of 15,000 data items. Two thirds of the data is generated randomly in the range of [0, 100] for each of the dimensions, and the remaining one third of the data defines three clusters which are generated by inserting additional data items in specific value ranges of the cluster dimensions. As a query region, we use the range [0, 10] for all six dimensions. In the resulting visualization (cf. Figure 8a), the four-dimensional clusters are only vaguely visible. By changing the weighting factors, however, the clusters can be made clearly perceptible (cf. Figure 8b). In general, clusters with lower dimensionality can be made perceptible by setting the weighting factor of one dimension to a significantly higher value than the weighting factors of the other dimensions. The dimension which is chosen to have the higher weight is arbitrary. The clustering in the visualization, however, is much better if a cluster dimension is chosen.

In experimenting with similar test data sets, we found that the extension of the cluster in multidimensional space has only a minor effect on the visualization. More important is the percentage of data items that form the cluster. Small clusters are only perceivable if they are close to the query region and have characteristics which are distinctly different from the remaining data items. The percentage of data items that need to be part of the cluster for the cluster to be perceptible depends on the distinctness between base data and cluster, on the dimensionality of the data, and on the distance between cluster and query region. The latter problem can be resolved, for example, by inverting the ordering of data items in the visualizations, which causes data items with larger distances to be closer to the center and therefore to be more visible.

Cluster with Different Data Distributions

In Figure 9, we present visualizations which are generated by using different data distributions for defining a cluster. The base data set consists of 10,000 data items, uniformly distributed in the range [-10000, 10000] for each of the dimensions. The cluster consists of 1,000 data items and the cluster dimensions differ in the distribution functions used and in their parameters. The parameters of the distribution functions for the cluster dimensions are given in the following table:

	uniform distribution			Gaussian distribution					
attribute	1	2	3	4	5	6	7	8	9
lower limit or mean	-100	-1000	-10000	0	0	0	100	1000	5000
upper limit or standard deviation	100	1000	10000	10	100	1000	1000	1000	5000

As a query region, we use the range [0, 10] for each dimension. Since the query region and the cluster overlap in dimensions one, two, four and five, the cluster is easily visible in the center of the corresponding windows (cf.

Figure 9a). If another query region is used, it is much more difficult to perceive the cluster (cf. Figure 9b). It is interesting that in the visualization there is no clustering visible for the dimension with a large range uniform distribution (dimension three) and only minor clustering occurs for the dimensions with a high standard deviation (dimensions six and nine). This effect becomes even more obvious if the weighting factor is varied (cf. Figure 9c).

Cluster with Functional Dependencies

In Figure 10, we present visualizations of test data which are generated by using different functional dependencies for defining a cluster. The base data set again consists of 10,000 data items, uniformly distributed in different ranges: [0, 1000] for dimensions one to three, [0, 2000] for dimensions four to six and [0, 1000000] for dimensions seven to nine. The cluster consists of 2,000 data items. Dimensions one to three of the cluster are the independent dimensions which are uniformly distributed in the range of [0, 1000]. The functional dependencies of the other dimensions are given in the following table:

<table>
<tr><td>attribute</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr>
<tr><td rowspan="2">functional dependency</td><td colspan="3" rowspan="2">uniform distribution
in the range [0, 1000]</td><td colspan="3">linear dependency
of attribute(s)</td><td colspan="3">quadratic dependency
of attribute(s)</td></tr>
<tr><td>1</td><td>2, 3</td><td>1, 2, 3</td><td>1</td><td>2, 3</td><td>1, 2, 3</td></tr>
</table>

As a query region, we use the origin of the nine-dimensional space. In Figure 10a-c, three visualizations of the data are provided which were produced using different weighting factors and an additional technique which we call ‘color inversion’. Color inversion means that the coloring of pixels is inverted with respect to the color scale. Note that the cluster dimensions which depend on multiple other dimensions are much better perceptible. This is due to the fact that differences of the independent dimensions sum up to higher differences for the dependent dimension, which are then better visible.

To use the described data exploration and visualization techniques as effectively as possible, it is necessary to have a global data exploration strategy. Some of our experiences, derived from an intensive use of our system, are briefly described in the following: If a user has no information about the data, it is best to start with the origin of the coordinate system as a query region and all attributes having the same weighting factor. The following steps are largely guided by the visual feedback the user gets from the visualizations. If the user gets hints for correlations, functional dependencies, or any kind of clustering, the user will try to verify the hypotheses. For this purpose, the user may change, for example, the query region and/or the weighting factors. If there are no hints for interesting data properties in the visualizations, the user may use a higher weighting factor for an arbitrary dimension, use the color inversion option, change the percentage of displayed data items or use different query regions. In this enumeration, the possibilities which according to our experience are most effective are mentioned first.

3.3 Comparison with other Multidimensional Visualization Techniques

In this subsection, we compare our techniques with other visualization techniques for multidimensional data. For the comparison, we use two well-known techniques that have proven useful for database-like data — parallel coordinates and stick figures (cf. section 2). We extended both techniques for the purpose of querying large databases by coloring the stick figure icon and the line segments of the parallel coordinate technique 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 allows the most relevant data items to be easily located and compared. This is especially important in dealing with larger data volumes. We compare the techniques with respect to the number of data items, the number of dimensions, the visibility of certain types of clusters, etc. We also provide examples of visualizations of the real data sets from our molecular biology application.

In Figure 11, we present parallel coordinate visualizations of the test data set containing 15,000 six-dimensional data items with three four-dimensional clusters. In the visualization (cf. Figure 11a), only one cluster is visible. The other two clusters are not visible due to the overlap of data items. By using different query ranges, it is possible to make the other clusters visible. Since the clusters are at different locations of the six-dimensional space, it is impossible to generate parallel coordinate visualizations that show more than one cluster at a time.

With the stick figure visualization, we also have the problem that overlapping data may prevent the visualizations from being useful. If all 15,000 data items are visualized (cf. Figure 11b), it is not possible to find structure in the data. A reduction of the amount of data to 10% makes the visualization more useful (cf. Figure 11c), allowing patterns created by stick figures with similar shapes and colors to become perceptible. Note that many of the displayed stick figures have a similar shape, which means that they are similar with respect to the dimensions that are assigned to the limbs of the stick figure icon. The similarity suggests that these dimensions belong to a cluster. In case of the stick figure technique, it is important which dimensions are assigned to the axes. If the right dimensions (e.g., the cluster dimensions) are chosen to be assigned to the axes, clusters may become visible due to the high density of stick figures in certain regions. In this case, however, the density of stick figures at certain locations, and not the multidimensionality of the stick figure icon, allows the user to find the clusters, which means that the same effect can be achieved by using scatterplot diagrams. To be able to use the multidimensionality of the stick figure icon, the overlap of data items needs to be avoided by reducing the number of data items.

In Figure 12, we present parallel coordinate and stick figure visualizations of the test data set where the cluster is defined using different distribution functions (cf. Figure 9). As in Figure 9, we use the origin of the coordinate system as a query region. Since the cluster is defined symmetrically around the origin, the cluster and its properties are easily visible in the parallel coordinate visualization (cf. Figure 12a). Interesting are the effects of the different distributions. To recall, the cluster is defined such that dimensions one to three have uniform distributions with different ranges, dimensions four to six have Gaussian distributions with different standard deviations, and dimensions seven to nine have Gaussian distributions with different means. Note that for dimensions six and nine the value range of the cluster is larger than the value range of the base data set. Again, the visibility of a cluster and its properties largely depends on the closeness of cluster and query region.

The appearance of the stick figure visualization largely depends on the dimensions which are assigned to the axes. For the visualization presented in Figure 12b, dimensions five and six are assigned to the axes. Since for both dimensions the cluster is defined as Gaussian distribution with different standard deviations, the shape of the cluster is oval in the visualization. The cluster, however, is only visible since the query region is in the middle of the cluster and since the more relevant data items are overlaid and colored differently. Note that most of the data items which are light green have a similar shape. This means that they are similar with respect to the dimensions that are assigned to the limbs of the stick figure icon.

We also used real data sets to compare our visual data mining techniques with the parallel coordinate and stick figure techniques. In the following, we provide parallel coordinate and stick figure visualizations of our molecular surface data. In the parallel coordinate visualization (cf. Figure 13), hot spots such as data items with negative distances (cf. dimensions five, six, and seven) are easily perceptible. Also easily observable are the range and distribution of distance values relative to the query region. The distances from the query region for dimensions one to seven are in their majority either positive (1 and 4-7) or negative (2 and 3). The visualization further reveals that dimensions four to seven are discrete in nature, which is most easily observable for dimension six. Note that even for relatively small numbers of data items (the considered molecule consists of 2,560 triangles), there is a high degree of overlap in the visualization.

In case of the stick figure visualization (cf. Figure 14), the expressiveness of the visualization largely depends on the assignment of dimensions to the axes. In cases of high overlay, the information that can be deduced from the visualization is mainly restricted to the distribution of data items with respect to the dimensions that are assigned to the axes. The direct mapping of two dimensions to the axes, however, also has a great advantage. For all data sets which have dimensions with inherent 2D- or 3D-semantics, the mapping is of great help to relate the visualizations to the real world entities. In case of the molecule surface data, there are inherent 3D-semantics defined by the first three dimensions. If two of those dimensions are mapped to the axes, the resulting visualization is a projection of the 3D-surface of the molecule onto those two dimensions. In Figure 14, we provide example visualizations of XZ-, YZ, and XZ-projections of the data.

In Table 1, we summarize the results of comparing our techniques to the parallel coordinate and stick figure visualization techniques and its colored derivatives. Our generalized spiral and axes techniques turn out to be the most useful techniques for visualizing very large amounts of data. Both techniques avoid overlapping data items and provide good results for most types of data characteristics. The only exceptions are data sets which have inherent 2D- or 3D-semantics. The stick figure technique is best suited for data sets which have a limited dimensionality and inherent 2D- or 3D-semantics, or at least a regular distribution of values for the two dimensions that are assigned to the axes. The stick figure technique helps to discover most types of clusters, but the usefulness of the generated visualizations largely depends on the choice of the axes dimensions. The parallel coordinate visualization technique is very useful for relatively small data sets with large dimensionality. It is especially helpful for providing an overview of the distribution of distances and for discovering exceptional data items (hot spots). The colored derivatives of the parallel coordinate and stick figure techniques help to make the data apparent, which are more relevant with respect to the query. With respect to the task of visualizing the most relevant data, they are therefore an improvement over the original parallel coordinate and stick figure techniques. At the same time, however, the color may distract the user's attention from the multidimensionality of the display, especially in case of the stick figure visualization.

	Generalized Spiral Technique	Generalized Axes Technique	Stick Figures	Colored Stick Figures	Parallel Coordinates	Colored Parallel Coordinates
max no. of data items[1]	++	++	o	+	o	+
max no. of dimensions[2]	+	+	o	o	+	+
overlapping data items	++	++	--	-[3]	--	-[3]
2D-/3D-semantics	-	-	++	++	-	-
hot spots	+	+	o	+	+	++
clusters	+	++	o	o	o	o
distributions	+	+	o	+[4]	o	+[4]
funct. dependencies	+	+	o	+[4]	o	+[4]

1. Only limited by the number of pixels of the display in the case of the generalized spiral and axes techniques. Limited to about 1,000 in case of parallel coordinate and stick figure techniques, and to about 5,000 for the remaining techniques.
2. Arbitrary in case of spiral, axes, and parallel coordinate techniques, but limited to a fixed number (ten dimensions for our stick figure icon) in case of the stick figure technique.
3. Due to coloring and overlay, the more relevant data items are visible in the colored derivatives.
4. Due to coloring and overlay, distribution and functional dependency clusters which are close to the query region are more easily perceptible in the colored derivatives of parallel coordinate and stick figure technique.

Table 1: Comparing the Generalized Spiral and Axes Techniques with Parallel Coordinate and Stick Figure Visualization Techniques

Since none of the techniques performs best for all different kinds of tasks, we believe that it is important to use different visual data mining techniques in parallel. This allows the exploitation of the strengths and avoids the weaknesses of the techniques. For example, in using the *VisDB* system for exploring real data sets, it turned out to be quite effective to use our spiral and axes techniques to reduce the amount of data, and then switch to the stick figure, and parallel coordinate techniques to further explore the remaining, much smaller data set. Comparing the visualizations generated by different visualization techniques is also very interesting, since it allows a correlation of the specific features of each visualization, providing more information than each of the visualizations independently.

4. Implementation and Performance Evaluation

All visualization techniques presented in the previous sections are implemented as part of the visualization and data analysis system *VisDB*. The interactive interface of the *VisDB* system allows the user to arbitrarily switch between the techniques, and to interactively modify the query, weighting factors, and percentage of displayed data items. Additional features include the possibility to select pixels to get the corresponding data values presented in certain fields. Details about the system are described in [51].

The *VisDB* system is implemented in C++/MOTIF and runs under X-Windows on HP 7xx machines. 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 applications. 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 purposes.

To evaluate the time efficiency of the *VisDB* system, we performed several test series using artificially generated test data sets with random distributions. For the test series we used an HP 735 machine with 128 Mbyte of main memory. In performing the test series, the machine was used exclusively by the *VisDB* system, and therefore, the measured CPU-times directly correspond to the elapsed time (reaction time of the system).

The steps in calculating the visualizations are presented in Figure 15. After loading the data into main memory, the *VisDB* system calculates the distances, normalizes and combines them, determines the desired percentage of data items with lowest overall distances, and sorts them according to their overall distance. The steps are executed one after another (no pipelining is used in our sequential implementation), and therefore the overall time (T_{Calc}) is the sum of the time for each of the steps. T_{Calc} depends on the number of data items (n), the number of dimensions (k), and the number of displayed data items (d).

$$T_{Calc}(n, k, d) = T_{CalcDist}(n, k) + T_{NormAttr}(n, k) + T_{CalcComb}(n, k) + T_{Cut}(n, d) + T_{NormResult}(d, k) + T_{Sort}(d)$$

The calculation performed in steps two, three, and four has to be done for each data value and therefore $T_{CalcDist}$, $T_{NormAttr}$ *and* $T_{CalcComb}$ only depend on the product of the number of data items and the number of dimensions [$O(n*k)$]. In steps five, six, and seven, the desired number of displayed data items (d) has to be considered. In step 5, the d data items with lowest overall distances are determined by computing the (d/n)-quantile which can be done in linear time using a bottom-up heap with d elements, inserting the remaining $(n-d)$ elements into the heap. Since the heap contains at most d elements and since d is a constant, the heap can be built in linear time. The actual time complexity however varies according to the distribution of the data. The worst, average, and best case time complexities of step 5 are presented graphically in Figure 16. The next step, normalizing the distances [0, 255], depends on the number of data items that need to be normalized, multiplied by the number of dimensions [$O(d*k)$]. The final sorting step can be performed most efficiently using the bucket sort algorithm. The bucket sort is ideal for our application since we use a linear mapping of values to colors and since the sorting granularity is limited. For our purpose, it is sufficient to use a bucket sort where the number of buckets correspond to the number of colors. The time complexity of the bucket sort algorithm is $O(d + b)$ [59] where b is the number of buckets. Since in our case b is constant ($b = \#colors = 256$), the complexity is linear [$O(d)$]. Note that again the complexity is independent of the data and their distribution.

The overall time complexity of our algorithm is the sum of the time complexities for each of the steps $[O(n \times k) + O(n) + O(d \times k) + O(d)]$, which corresponds to

$$O(n \cdot k)$$

since $d \leq n$. Note that for a large number of data items and constant screen resolution, the worst and best case time complexity are in the same order of magnitude. The time measurements taken from empirical test series confirm the linear dependency on n and k (cf. Figure 17). In our empirical tests, we also compared the time portions which are needed for each of the steps. From their graphical representation (cf. Figure 18), it becomes obvious that calculating the distances ($T_{CalcDist}$) and calculating the combination ($T_{CalcComb}$) are the most time-consuming steps which use more than 80% of the overall time. Fortunately, these steps are ideally suited for parallelization. For details, the reader is referred to [51].

5. Conclusion

Visual data mining techniques are useful for the exploration and analysis of large databases to find interesting data clusters and their properties. Our approach to data mining aims at an adequate support of the human by the computers, and combines database query and information retrieval techniques with new types of visualization techniques. Using our visual data mining techniques, tens to hundreds of thousands of data items with an arbitrary dimensionality can be visualized on the screen at the same time. Five different visualization techniques — all implemented as part of the *VisDB* system —support the user in different phases of the data exploration process. The results of using our visual data mining techniques in different areas show that visual data mining techniques are of high importance for a wide range of applications including data mining tasks (finding correlations between attributes, finding groups of similar data, and finding hot spots) and similarity retrieval (finding an adequate combination of value ranges). For a comparison of our techniques with the parallel coordinate and stick figure techniques, we use real data from a molecular biology application and artificial test data sets generated according to a systematic test data model. We show that our techniques are useful for visualizing a wide range of data characteristics and are superior to the other techniques with respect to the amount of data that can be visualized at one point of time.

Future work includes further evaluations that need to determine which techniques are most appropriate for specific types of correlations, clusters, and functional dependencies. The evaluations are also necessary as basis for improving existing visual data mining techniques and for visualizing even larger amounts of data. Also important is to directly interface our system with commercially available database systems. As first investigations show, current database systems are only useful to a limited extend for supporting interactive visualization systems such as the *VisDB* system. Database systems support high transaction rates and a fast search of specific data items, but most of them do not provide a sufficient performance for range queries on multiple attributes which are required by our system. A possible solution of this problem is the use of multidimensional data structures. Future work needs to determine which multidimensional data structures are best suited for our application. Another problem of current database systems is that the queries are executed separately and no support for incrementally changing queries is provided, which would be needed for real interactivity of the *VisDB* system in dealing with very large databases containing millions of data items. This brief enumeration of problems demonstrates that developing a secondary storage based version of the *VisDB* system using commercially available database systems poses many interesting research questions which need to be solved.

Footnotes

Affiliation of authors

Institute for Computer Science, University of Munich

Oettingenstr. 67, D-80538 München, Germany

e-mail: {keim, kriegel}@informatik.uni-muenchen.de

Footnote 1 on page 9

The quality of the B/W version of our visualizations is rather bad compared to the quality of the color visualizations on the screen. Structures in the visualizations which are easy to perceive in the color version might therefore be difficult to perceive in the B/W version. A color postscript version of the paper may be obtained from our ftp-server (URL: 'ftp://arcadia.informatik.uni-muenchen.de/pub/local/dbs/pubs/TKDE96.ps'). Readers who do not have access to the world wide web may obtain a color paper version upon request from the authors.

References

[1] W. J. Frawley, G. Piatetsky-Shapiro, C. J. Matheus: *'Knowledge Discovery in Databases: An Overview',* in: Knowledge Discovery in Databases, AAAI Press, Menlo Park, CA, pp. 1-27, 1991.

[2] G. Dunn, B. Everitt: *'An Introduction to Mathematical Taxonomy',* Cambridge University Press, Cambridge, MA, 1982.

[3] T. Gaasterland, P. Godfrey, J. Minker: *'An Overview of Cooperative Answering',* Journal of Intelligent Information Systems, Vol. 1, pp. 123-157, 1992.

[4] T. M. Anwar, H. W. Beck, S. B. Navathe: *'Knowledge Mining by Imprecise Querying: A Classification-Based Approach',* Proc. 8th Int. Conf. on Data Engineering, Tempe, AZ, pp. 622-630, 1992.

[5] A. Motro: *'FLEX: A Tolerant and Cooperative User Interface to Databases',* IEEE Transactions on Knowledge and Data Engineering, Vol. 2, No. 2, pp. 231-246, 1990.

[6] G. Salton, C. Buckley: *'Term-Weighting Approaches in Automatic Text Retrieval',* Information Processing and Management, Vol. 24, No. 5, pp. 513-523, 1988.

[7] H. P. Frei, S. Meienberg: *'Evaluating Weighted Search Terms as Boolean Queries',* Proc. GI/GMD-Workshop, Darmstadt 1991, in: Informatik-Fachberichte, Vol. 289, pp. 11-22, 1991.

[8] C. J. Matheus, P. K. Chan, G. Piatetsky-Shapiro: *'Systems for Knowledge Discovery in Databases',* IEEE Transactions on Knowledge and Data Engineering, Vol. 5, No. 6, pp. 903-913, 1993.

[9] J. S. Park, M.-S. Chen, P. Yu: *'An Effective Hash-Based Algorithm for Mining Association Rules'*, Proc. ACM SIGMOD Int. Conf. on Management of Data, San Jose, CA, pp. 175-186, 1995.

[10] R. Agrawal, R. Srikant: *'Fast Algorithms for Mining Association Rules'*, Proc. 20th Int. Conf. on Very Large Databases, Santiago, Chile, pp. 487-499, 1994.

[11] J. Han, Y. Cai, N. Cercone: *'Data-driven Discovery of Quantitative Rule in Relational Databases',* IEEE Transactions on Knowledge and Data Engineering, Vol. 5, No. 1, pp. 209-240, 1993.

[12] R. T. Ng, J. Han: *'Efficient and Effective Clustering Methods for Spatial Data Mining',* Proc. 20th Int. Conf. on Very Large Data Bases, Santiago, Chile, pp. 144-155, 1994.

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

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

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

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

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

[18] A. Inselberg: *'The Plane with Parallel Coordinates, Special Issue on Computational Geometry',* The Visual Computer, Vol. 1, pp. 69-97, 1985.

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

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

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

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

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

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

[25] J. LeBlanc, M. O. Ward, N. Wittels: *'Exploring N-Dimensional Databases', Proc.* Visualization '90, San Francisco, CA, pp. 230-237, 1990.

[26] G. Robertson, S. Card, J. Mackinlay: *'Cone Trees: Animated 3D Visualizations of Hierarchical Information',* Proc. ACM CHI Int. Conf. on Human Factors in Computing, pp. 189-194, 1991.

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

[28] M. P. Consens, A. O. Mendelzon: 'Hy^+*: A Hygraph-based Query and Visualization System'*, Proc. ACM SIGMOD Int. Conf. on Management of Data, Washington, DC, pp. 511-516, 1993.

[29] V. Vasudevan: *'Supporting High Bandwidth Navigation in Object-Bases',* Proc. 10th Int. Conf. on Data Engineering, Houston, TX, pp. 294-301, 1994.

[30] R. A. Becker, S. G. Eick, G. J. Wills: *'Visualizing Network Data',* IEEE Transactions on Visualizations and Graphics, Vol. 1, No. 1, pp. 16-28, 1995.

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

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

[33] V. Anupam, S. Dar, T. Leibfried, E. Petajan: *'DataSpace: 3-D Visualization of Large Databases',* Proc. Int. Symp. on Information Visualization, Atlanta, GA, pp. 82-88, 1995.

[34] D. A. Keim, H.-P. Kriegel, T. Seidl: *'Supporting Data Mining of Large Databases by Visual Feedback Queries',* Proc. 10th Int. Conf. on Data Engineering, Houston, TX, pp. 302-313, 1994.

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

[36] D. A. Keim, H.-P. Kriegel, M. Ankerst: *'Recursive Pattern: A Technique for Visualizing Very Large Amounts of Data',* Proc. Visualization '95, Atlanta, GA, pp. 279-286, 1995.

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

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

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

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

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

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

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

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

[45] D. A. Keim, H.-P. Kriegel: *'VisDB: A System for Visualizing Large Databases',* Proc. ACM SIGMOD Int. Conf. on Management of Data, San Jose, CA, p. 482, 1995.

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

[47] D. Hilbert: *'Über stetige Abbildung einer Linie auf ein Flächenstück'*, Math. Annalen, Vol. 38, pp. 459-460, 1891.

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

[49] G.T. Herman, H. Levkowitz: *'Color Scales for Image Data'*, Computer Graphics and Applications, pp. 72-80, 1992.

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

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

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

[53] J. Friedman, J. Tukey: *'A Projection Pursuit Algorithm for Exploratory Data Analysis',* IEEE Transactions on Computers, Vol. 23, pp. 881-890, 1974.

[54] H. Chernoff: *'The Use of Faces to Represent Points in k-Dimensional Space Graphically',* Journal American Statistical Association, Vol. 68, pp 361-368, 1973.

[55] R. M. Pickett: *'Visual Analyses of Texture in the Detection and Recognition of Objects',* in: Picture Processing and Psycho-Pictorics, Lipkin B. S., Rosenfeld A. (eds.), Academic Press, New York, 1970.

[56] G. Grinstein, J. C. Sieg, S. Smith, G. M. Williams: *'Visualization for Knowledge Discovery',* Technical Report, Computer Science Department, University of Massachusetts at Lowell, MA, 1991.

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

[58] R. D. Bergeron, D. A. Keim, R. Pickett: *'Test Data Sets for Evaluating Data Visualization Techniques',* in: Perceptual Issues in Visualization, Springer, Berlin, pp. 9-22, 1994.

[59] D. E. Knuth: *'The Art of Computer Programming: Sorting and Searching',* Vol. 3, Addison-Wesley, 1973.

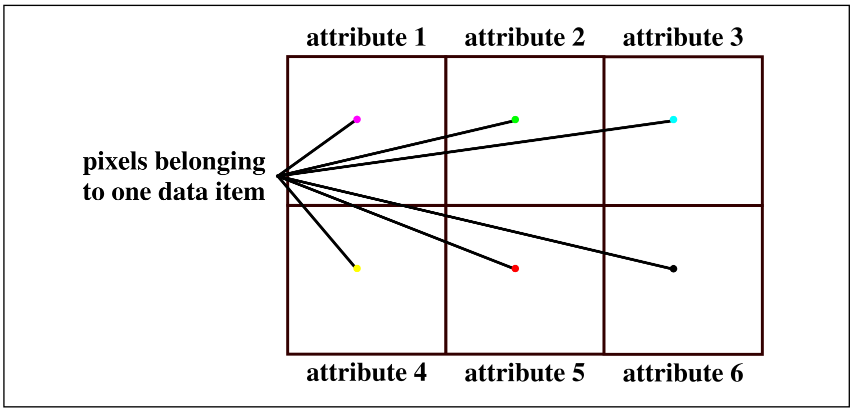
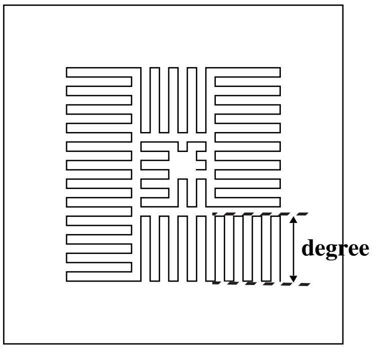
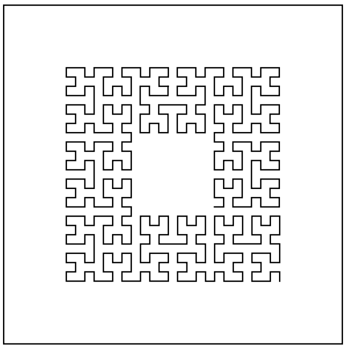


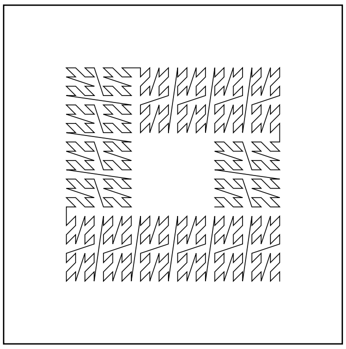
Figure 1: Arrangement of Attribute Windows for Data with Six Attributes



a. Snake-Spiral

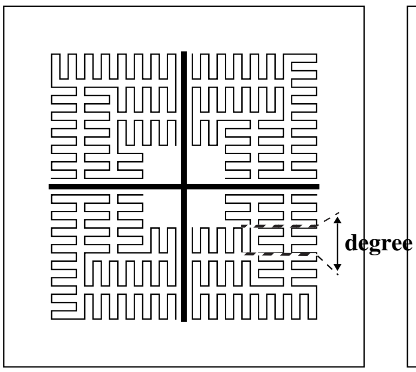


b. Peano-Hilbert-Spiral

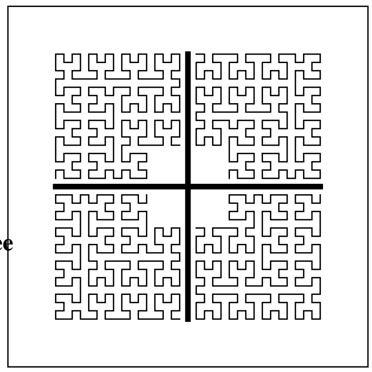


c. Morton-Spiral

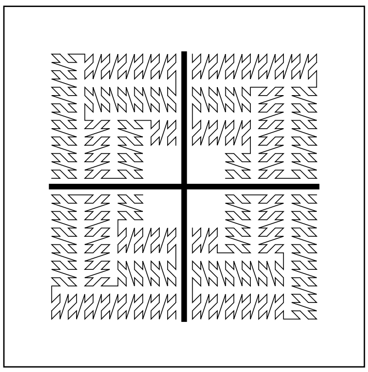
Figure 2: Generalized Spiral Arrangement of one Attribute (degree of local pattern is 8)



a. Snake-Axes

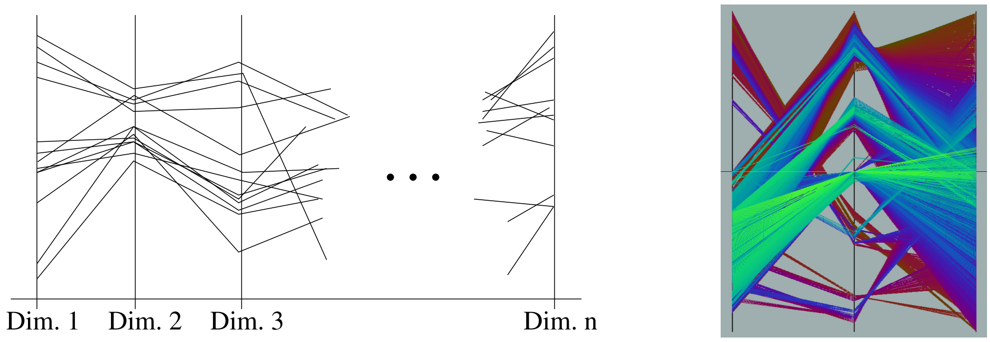


b. Peano-Hilbert-Axes



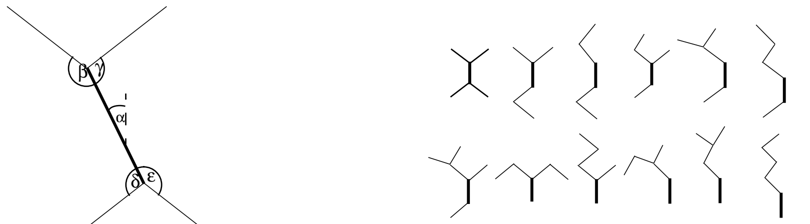
c. Morton-Axes

Figure 3: Generalized Axes Arrangement of one Attribute (degree of local pattern is 4)



a. Technique b. Example

Figure 4: Parallel Coordinate Visualization Technique



a. Stick Figure Icon b. A Family of Stick Figures

Figure 5: Stick Figure Visualization Technique

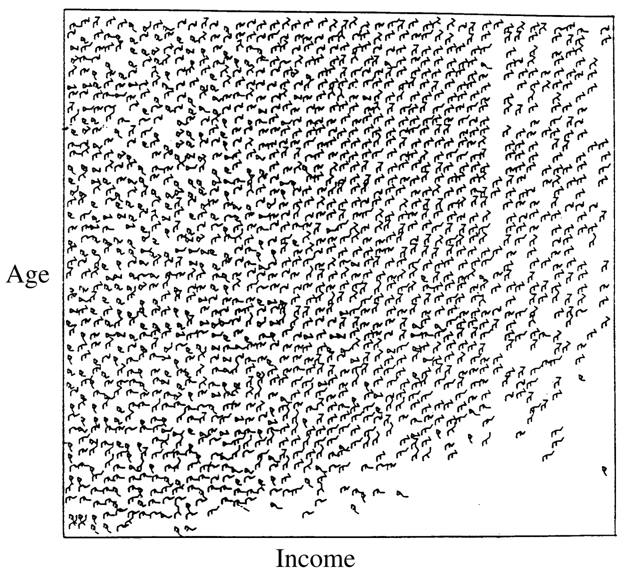
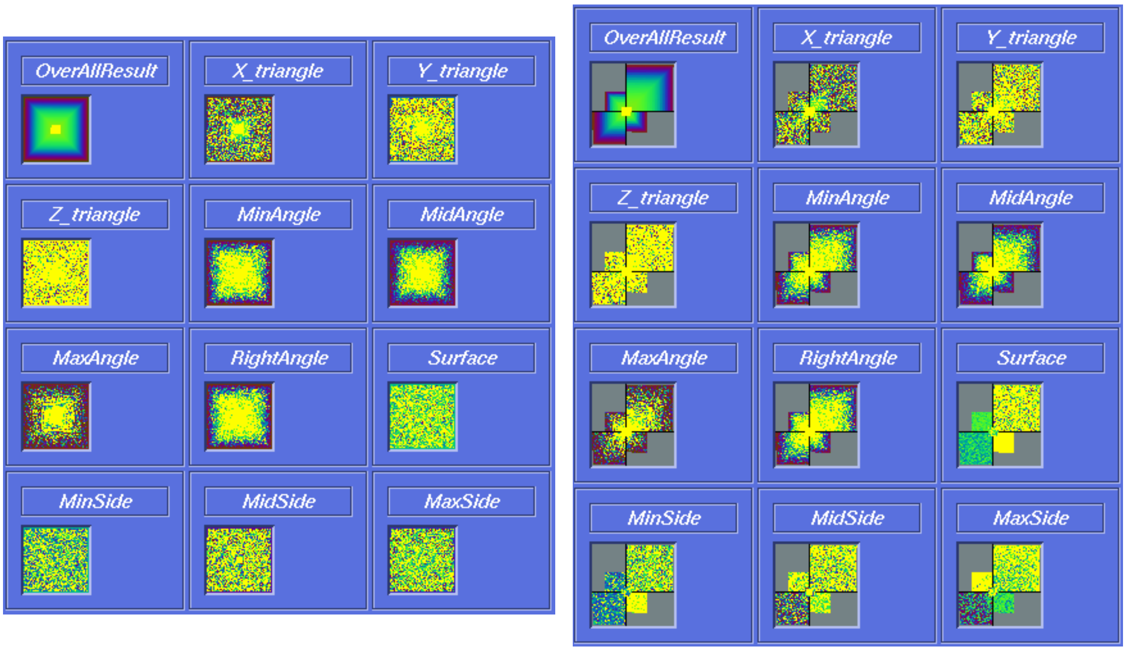


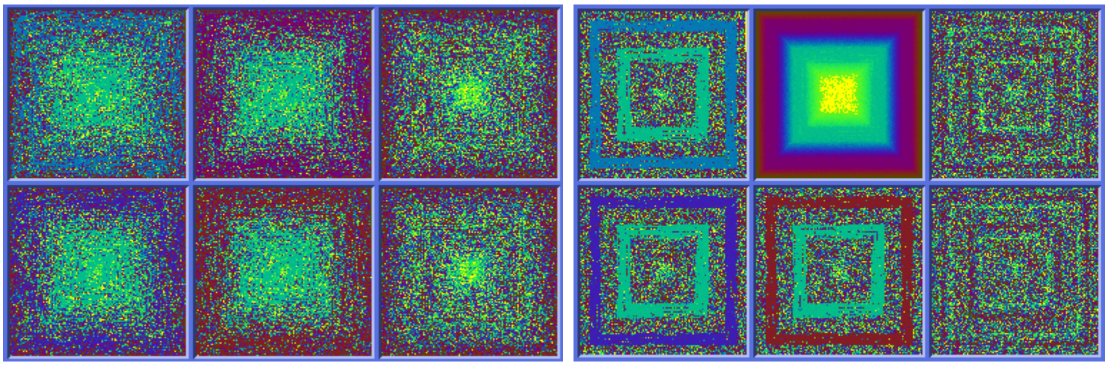
Figure 6: Stick Figure Visualization of Census Data

(used by permission of G. Grinstein, Institute of Visualization and Perception Research, University of Massachusetts at Lowell; cf. [GPW 89])



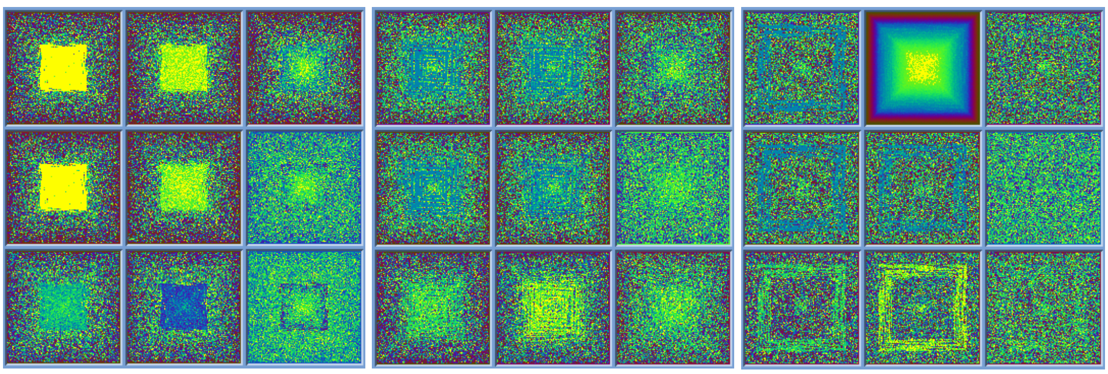
a. Spiral Technique b. Axes Technique

Figure 7: Partitioning a Molecule into Regions by using Properties of the Triangulation



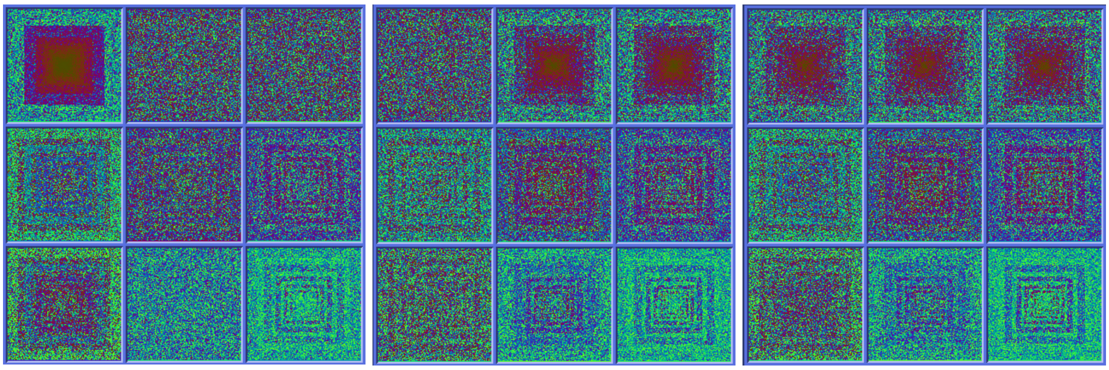
a. Four-dimensional Cluster b. Higher Weight of Attribute 2

Figure 8: Four-dimensional Cluster in Six-dimensional Data



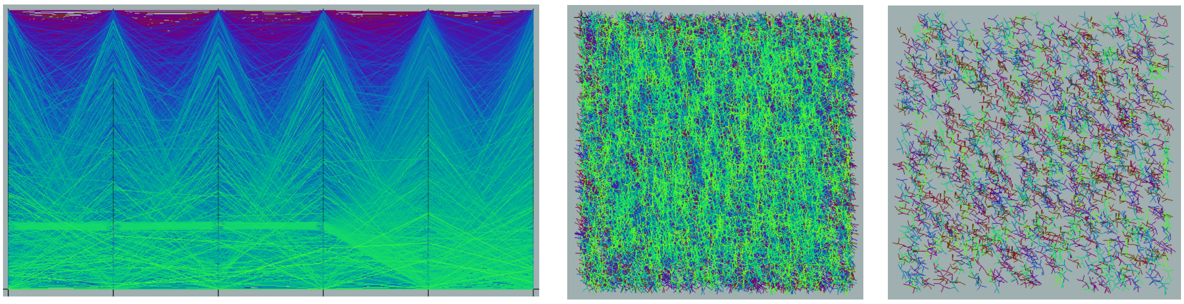
a. Query Region [0, 10] b. Query Region [4900, 5000] c. Higher Weight of Attr. 2

Figure 9: Cluster Defined by Different Data Distributions



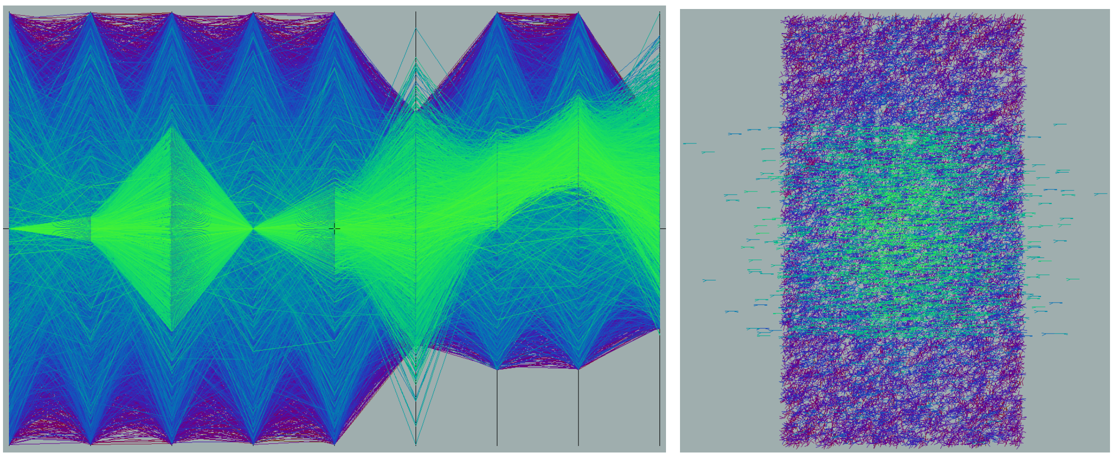
a. Higher Weight of Attr. 1 b. Higher Weight of Attr. 2-3 c. Higher Weight of Attr. 1-3

Figure 10: Effect of Color Inversion



a. 4-dim.Clusters b. 100% of the Data c. 10% of the Data

Figure 11: Four-dimensional Clusters in Six-dimensional Data



a. Parallel Coordinate Visualization b. Stick Figure Visualization

Figure 12: Clusters Defined by Different Distributions

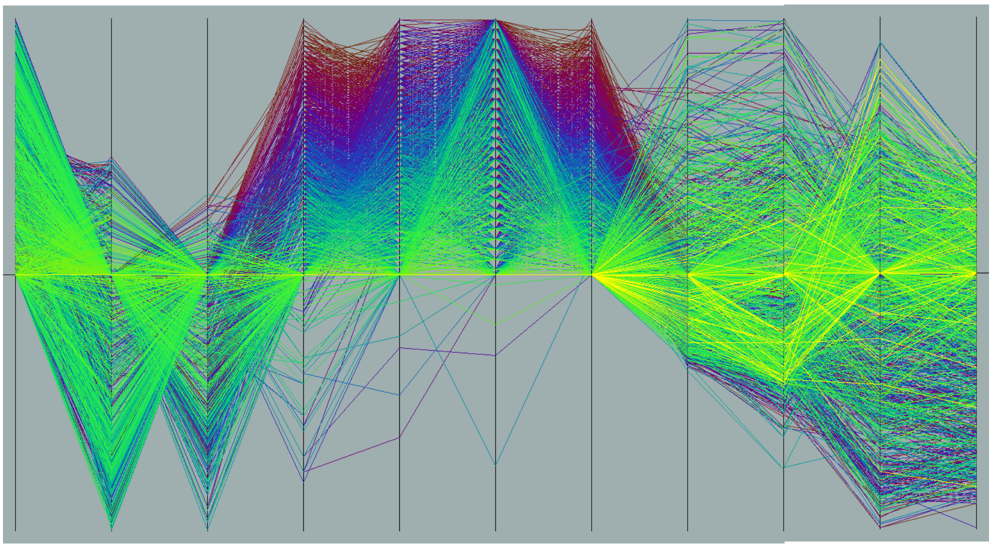
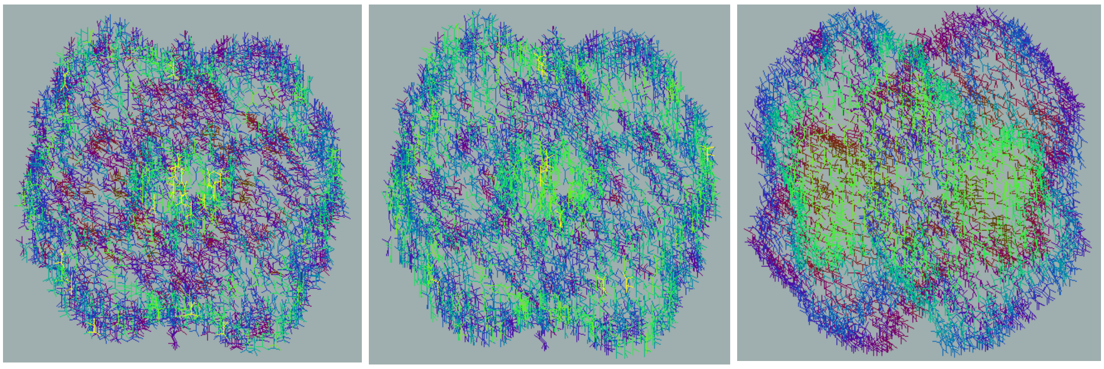


Figure 13: Parallel Coordinate Visualization of the Molecule Surface Data



a. b. c.

Figure 14: Examples for Clusters in the Molecule Surface Data

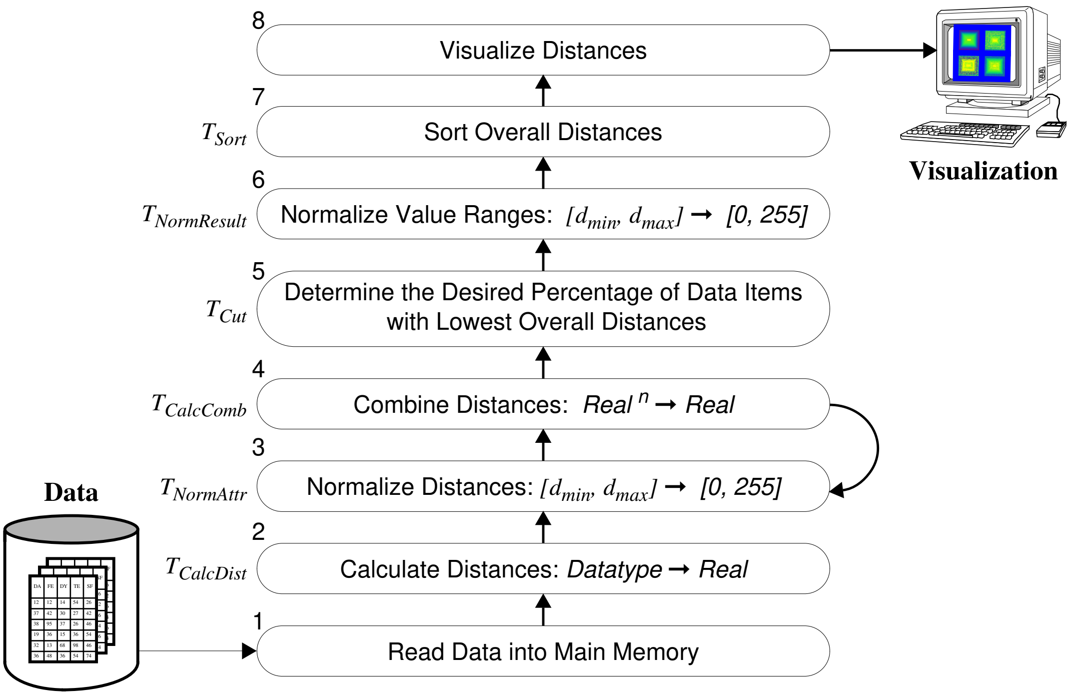


Figure 15: Steps in Calculating the Visualizations

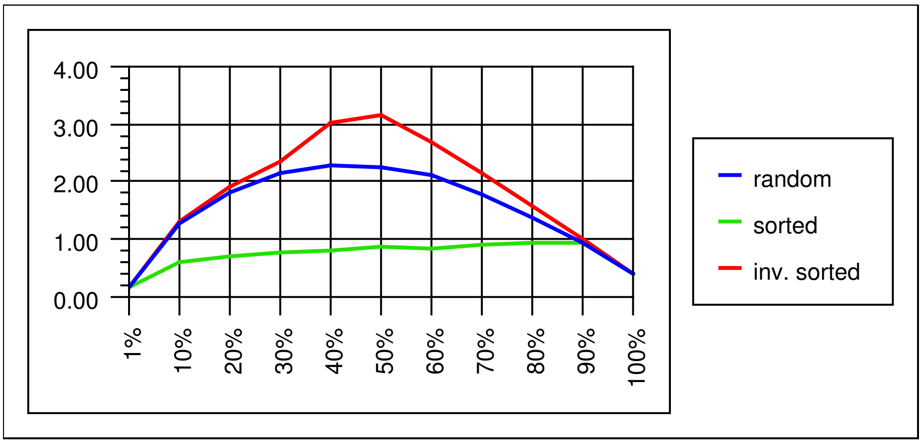


Figure 16: Graphical Representation of Worst, Average, and Best Case for T_{Cut}

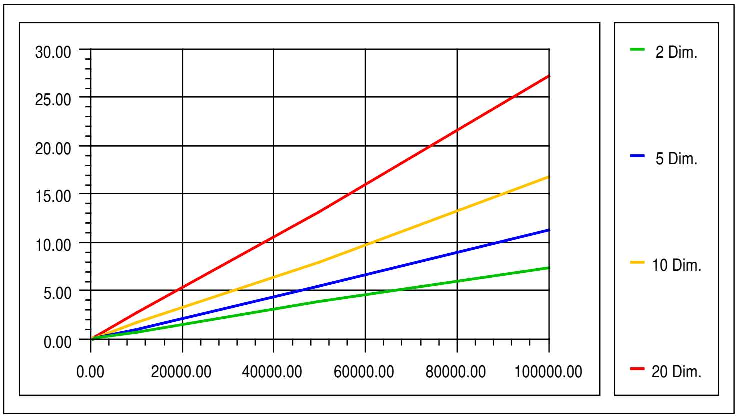


Figure 17: Graphical Representation of T_{Calc}

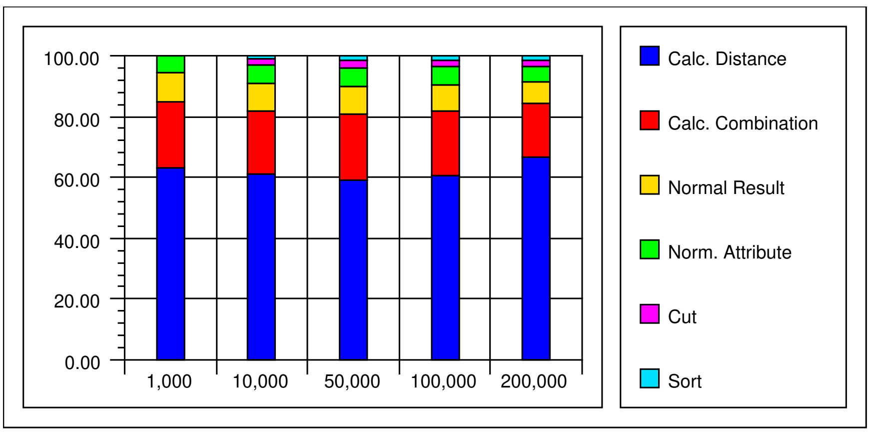


Figure 18: Time Portions Needed for each of the Steps