VisDB: Database Exploration Using Multidimensional Visualization

Daniel A. Keim

Hans-Peter Kriegel

Ludwig-Maximilians-Universität München
Institut für Informatik
Bericht 9413

August 1994

VisDB: Database Exploration Using Multidimensional Visualization

Daniel A. Keim, Hans-Peter Kriegel

Institute for Computer Science, University of Munich
Leopoldstr. 11B, D-80802 Munich, Germany
{keim, kriegel}@informatik.uni-muenchen.de

Abstract

In this paper we describe the VisDB system, which allows an exploration of large databases using visualization techniques. The goal of the system is to support the query specification process by using each pixel of the display to represent one data item of the database. By arranging and coloring the pixels according to the relevance of the data items with respect to the query, the user gets a visual impression of the resulting data set. Using sliders for each condition of the query, the user may change the query dynamically and receives immediate feedback from the visual representation of the resulting data set. Different visualization techniques are available for different stages of exploration. The first technique uses multiple windows for the different query parts, providing visual feedback for each part of the query and helping the user to understand the overall result. The second technique is an extension of the first one, providing additional information by assigning two dimensions to the axes. The third technique uses a grouping of dimensions and is designed to support a focused search on smaller data sets.

Keywords: Visualizing Large Data Sets, Visualizing Multidimensional Multivariate Data, Data Exploration and Analysis, Visual Query Support, Interfaces to Database Systems

Published in:
IEEE Computer Graphics and Applications, September 1994, pp. 40-49.

1 Introduction

In very large databases with tens of thousands or even millions of data items, it is often a problem to find the data in which a person is interested. Scientific, engineering, and environmental databases, for example, contain large amounts of data that in many cases are collected automatically via sensors and (satellite) monitoring systems. In querying such systems, even users who are experienced in using a database and query system may have difficulties finding the interesting data spots. If the user does not know the data and its distribution exactly, many queries may be needed to find the interesting data sets. The core of the problem in searching huge amounts of data is the process of query specification. With today's database systems and their query interfaces, a person has to issue queries in a one-by-one fashion. Generally, there are no possibilities to slightly change a query or to express vague queries. Most importantly, the user gets no feedback on his query, except the resulting data set containing either no data items and thus no hint for continuing the search, or too many data items and thus too many to look at.

Many approaches have been made to improve the database query interface by providing better feedback in cases of unexpected results. One approach consists of graphical database interfaces that allow the user to browse the data (e.g. FLEX [Mot 90] or GRADI [KL 92]). Another approach uses cooperative database interfaces [Kap 82] that try to give 'approximate answers' in cases where the queries do not provide a satisfactory answer. Such systems use techniques like query generalization that is dropping or relaxing a selection predicate in cases where the original queries fail, and statistical approximations or intensional responses instead of full enumeration in the case of large results (key ideas were presented for the first time in [JKL 77]). Cooperative systems mainly help the user to understand the results and to refine erroneous queries, but do not help to find interesting properties of the data such as functional dependencies, local correlations, or exceptional data items. Another area that relates to our work is the area of information retrieval. In information retrieval, a lot of research has been done to improve recall and precision in querying databases of unstructured data such as (full) text. User-provided relevance assessments of results are e.g. used to re-rank the results or to re-run adapted queries [SM 83].

In the *VisDB* system, the main idea is to support the query specification process by visually representing the result. The tables of (relational) databases may also be seen as sets of multidimensional data with the number of attributes corresponding to the number of dimensions. Often it is not clear which dimensions are independent and which are dependent. In most cases, only a limited number of the dimensions are of interest in a certain context. In the VisDB system, we therefore restrict the number of visualized dimensions to those that are part of the query, i.e. the dimensionality of our visualizations corresponds to the number of selection predicates. The visualization techniques employed in the *VisDB* system are not only useful for supporting simple one table queries (c.f. sections 3-5) but also for supporting complex queries involving nested Boolean expressions and complex join conditions (c.f. section 6).

Many approaches to visualize arbitrary multivariate, multidimensional data have been proposed for various purposes in different application contexts. Many examples can be found in the well-known books of Bertin [Ber 81] and Tufte [Tuf 83]. More recent techniques include shape coding [Bed 90], worlds within worlds [FB 90], parallel coordinates [ID 90], iconic displays [PG 88, BMS 92], dimensional stacking [LWW 90], hierarchical plotting [MGTS 90], and dynamic methods as presented in [MZ 92]. In dealing with databases consisting of tens of thousands to millions of data items, our goal is to visualize as many data items as possible at the same time to give the user some kind of feedback on the query. The obvious limit for any kind of visualization is the resolution of current displays which is in the order of one to three million pixels, e.g. in the case of our 19 inch displays with a resolution of 1024 x 1280 pixels about 1.3 million pixels. Important is the interactiveness of such a system. Empirical studies show that interactive, slider-based interfaces improve efficiency and accuracy in accessing databases considerably [Shn 94]. Equally important is the possibility of getting immediate feedback on the modified query. By playing with such a system, the user may learn more about the data than by issuing hundreds of queries.

2 A New Query Paradigm

In today's database systems, queries are specified in a one-by-one fashion. This is adequate if the user of the database exactly specifies the desired data and accesses a clearly separated data set. For many application areas where databases are used on a regular basis, e.g. accounting, reservation systems, and so on, queries are often based on keys, accessing exactly the desired data. For example, if a person deposits money in a specific account or if all transactions for a specific account are searched for, the resulting data set is clearly separated, and therefore one query is in general sufficient to get the desired data. In other application areas, however, especially those with very large data volumes such as scientific, engineering and environmental databases, it is often difficult to find the desired data. Problems occur if the database contains data different from what the user expects or if the user does not know exactly what to look for. In the latter case, querying the database is like an inexact search. If a query does not provide the desired result, usually the database is queried by another similar query, differing in just one detail. While searching for the desired data, many similar queries are often issued before the user is able to find the desired result.

Many problems in querying a database arise if the user does not know the database system, the data model and query language, or the schema of the database. But even if the user has perfect knowledge in all these domains, i.e. all queries are completely correct (syntactically as well as semantically), queries may have results which do not correspond to the user's intentions. The reason is that the user does not know the specific data in the database. In this case, it is very difficult for the user to estimate the amount of data that will be retrieved, especially for range queries and complex queries with many selection predicates. With our query interface, we use visualization techniques to give the users more feedback on the results of their queries. If for example researchers in environmental science are searching a huge database of test series for significant values, they might be looking for some correlation between multiple dimensions for some specific period of time and some geographic region. Since none of the parameters for the query is fixed, it is in general very difficult to find the desired information. The researchers would probably start to specify one query that corresponds to some assumption and after issuing many refined queries and applying statistical methods to the results, they might find some interesting correlation.

With the *VisDB* system, the query specification process would be much easier. In the beginning, the users would still have to specify one query. Then, guided by the visual feedback, they may interactively change the query according to the impression from the visualized results. In exploring very large databases, the visualization of results coupled with the possibility to incrementally refine the query are an effective way to find the interesting properties of the data. The key idea of the *VisDB* system is to use the phenomenal abilities of the human vision system, which is able to analyze small to midsize amounts of data very efficiently and recognizes immediately patterns in images which would be very difficult (in some cases even impossible) if done by the computer. One major research challenge is to find adequate ways of visually presenting multidimensional data to support the user in analyzing and interpreting the data.

3 Visualizing Large Data Sets of Multidimensional Data

The basic idea of our visualization techniques is to use each pixel of the screen to visualize the data items resulting from a query. As the result of a query, the user does not only get the data items fulfilling the query, but also a number of data items that are approximately fulfilling the query. The approximate results are determined by using distance functions for each of the selection predicates which are combined into the relevance factor. The distance functions are datatype and application dependent and must be provided by the application. Examples for distance functions are the numerical difference (for metric types), distance matrices (for ordinal and nominal types), lexicographical, character-wise, substring or phonetic difference (for strings), and so on. Having calculated the distances for each of the selection predicates, the distances are combined into the relevance factor. For important aspects such as normalizing and weighting the different selection predicates, the formulas we use to calculate the relevance factors, and the heuristics used to reduce the number of displayed data items, the reader is referred to “Appendix: Calculating the Relevance Factors”.

3.1 Basic Visualization Technique

The basic idea for visually displaying the data on the screen is to sort them according to their relevance with respect to the query and to map the relevance factors to colors. The sorting is necessary to avoid completely sprinkled images that would not help the user in understanding the data. One question in designing the system was how to arrange the relevance factors on the screen. We tried several arrangements such as top-down, left-to-right, centered, etc. and found that arrangements with the highest relevance factors centered in the middle of the window seemed to be the most natural ones. The one hundred percent correct answers are colored yellow in the middle and the approximate answers create a rectangular spiral around this region (c.f. figure 1). The colors range from yellow to green, blue and red to almost black and denote the distance from the correct answers. The colorscale used has been determined empirically (see “Appendix: Coloration of the Relevance Factors”). To relate the visualization of the overall result to visualizations of the different selection predicates (dimensions), we generate a separate window for each selection predicate of the query and arrange them next to each other (c.f. figure 2). In the separate windows we place the pixels for each data item at the same relative position as the overall result for the data item in the overall result window. All the windows together make up the multidimensional visualization. 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.

3.2 Mapping two Dimensions to the Axes

We also experimented with other arrangements of the data items on the screen. One straightforward idea was to display the data in 2D or 3D with selected dimensions assigned to the axes. With such arrangements, however, one has the problem that on the one hand many data items may be concentrated in some area of the screen while other areas are virtually empty, and on the other hand many data items are superimposed and therefore not visible. Although 2D- or 3D-visualizations may be very helpful, e.g. in all cases where data have some inherent two- or three-dimensional semantics, we did not pursue this idea for several reasons. One reason was that in most cases the number of data items that can be represented on the screen at the same time is quite limited. This was in contrast to one of our goals, namely to present as many data items as possible on the screen. A second reason was that in most cases where a 2D- or 3D-arrangement of the data really makes sense, systems using such arrangements have already been built. For spatial queries on two-dimensional data, for example, a 2D-visualization is obviously the best support for querying the database and basically all Geographical Information Systems provide such visual representations of the data. For all cases, however, where no inherent two- or three-dimensional semantics of the data exists, our representation can be of great value in providing visual feedback when querying the database. Stimulated by real 2D- or 3D- representations of the data, we decided to improve our interface by including some feedback on the direction of the distance into the visualization. The basic idea is to assign two dimensions to the axes and to arrange the relevance factors according to the direction of the distance; for one dimension negative distances are arranged to the left, positive ones to the right and for the other dimension negative distances are arranged to the bottom, positive ones to the top (c.f. figure 3).

With this kind of representation, we do not represent the distance of data items directly by their location, but we denote the absolute value of the distance by their color and the direction by their location relative to the correct answers (colored yellow). The advantage of this kind of representation is that each data item may be assigned to one pixel and no overlay of data items with the same distance does occur. A problem may arise in some special cases if e.g. no data items exist that have a negative distance for both dimensions but many data items that have a negative distance for one of them and a positive one for the other one.

In this case, the bottom left corner of the window would be completely empty. In the worst case, two diagonally opposite corners of the window may be completely empty and, as a result, only half as many data items as possible are presented to the user. Even in this case, the user gets valuable information on how to change the query to get more or less results. In summary, it may be stated that maximizing the number of data items conflicts with arrangements that have multiple dimensions assigned to the axes.

3.3 Grouping the Dimensions for each Data Item

In both the original arrangement and the 2D-arrangement, the pixels corresponding to the different dimensions of one data item are distributed in different windows for each dimension. In contrast, in the grouping arrangement all dimensions for one data item are grouped together in one area. Each area is arranged in the rectangular spiral-shape according to the combined relevance factor of the considered data items (c.f. figure 4). The coloring of the distances for the different dimensions may be the same as in the original or the 2D-arrangement. The generated visualizations, however, are completely different from those of the original and 2D-arrangements. The visualizations generated using the grouping arrangement consist of only one window with many areas visualizing all dimensions of the considered data items instead of many windows, each providing a visual representation of only one dimension (c.f. figures 4 and 6c). At this point, it should be mentioned that the idea of grouping the dimensions into one area is similar to the shape coding approach described in [Bed 90]. In our approach, however, we do not focus on shape to distinguish the data items, and the criterion and arrangement of the data items is different.

Preliminary experiments show that for the grouping arrangement more pixels per data value are needed. In the case of basic and the 2D-arrangements, one pixel per dimension per data item is used. Empirical tests show that in the case of the grouping arrangement, an area of at least 2 x 2 (better: 3 x 3 or 4 x 4) pixels per dimension per data item is needed for the visualization to provide useful results. This implies that only one-fourth (or even one-ninth or one-sixteenth) of the data items can be displayed on the screen at one point of time, which means that the grouping arrangement is only suitable for a focused search on smaller data sets. Note that additional pixels are needed for the area surrounding each data item. In contrast to the other arrangements, a border is necessary since it would otherwise be impossible to know which pixels belong to which data item. In figures 6 and 7, we show two data sets using all three visualization techniques. Figure 6 shows an eight-dimensional data set with 1,000 data items and figure 7 shows one with 7,000 data items.

Despite the fact that fewer data items may be visualized, the grouping arrangement provides more useful visualizations for data sets with larger dimensionality. In the original and 2D-arrangements, the pixels for each dimension of the data items are only related by their position. For relatively small dimensionality (less than 8 dimensions), it seems to be quite easy for humans to relate the different portions of the screen. The larger the dimensionality becomes, the more difficult it gets to relate the different portions of the visualization and to perceive correlations among them. In the case of the grouping arrangement, it is not necessary for the user to relate different portions of the screen, and it therefore seems to be advantageous for larger dimensionalities.

4 Interactive Data Exploration

In using our query and visualization system, the possibility to modify queries dynamically is important. Since modifications have a direct impact on the visualizations, the user will get immediate feedback on the effects of the changes. The visualizations provide feedback on the amount of data retrieved, on the restrictiveness of the conditions, on the distribution of the distances for each condition and on special areas the user might be interested in. For example, if the yellow region in the middle of each window is getting larger (shrinking), more (less) data items fulfill the condition; if a window is getting darker (brighter), the corresponding selection predicate is getting more (less) restrictive; if the overall structure of a window is changing, the distribution of distances for the corresponding selection predicate is changing, and so on. These visual indicators are of valuable help in understanding the effects of query modifications quickly and learning more about the data in the database, especially in the context of large databases with millions of data items.

In querying a database using the *VisDB* system, the user initially specifies a query using graphical user interfaces such as GRADI [KL 92] or traditional query languages such as SQL. As a result of the query, the user gets the interactive query and visualization interface of the *VisDB* system, which is divided into the ‘Visualization’ portion on the left and the ‘Query Modification’ portion on the right (c.f. figure 5). In the ‘Visualization’ portion, the resulting data set including a certain percentage of approximate answers is displayed by using one of the visualization methods described in section 3. In the ‘Query Modification’ portion, sliders for modifying the selection predicates and weighting factors as well as some other options are provided. Different kinds of sliders are available for different datatypes and different distance functions. Sliders for numbers, for example, allow graphical manipulations of the lower and upper limits, or of the medium value and some allowed deviation. Sliders for discrete types reflect the discrete nature of the data by allowing only discrete movements of the slider. Sliders for non-metric types (ordinal and nominal datatypes) may be, for example, enumerations of the possible values with the possibility to select each of the values. Special sliders may be designed for special datatypes and special distance functions, e.g. for strings with different distance functions.

Below the sliders, several parameters are listed for each selection predicate, namely the ‘number of results’, the ‘query range’ and ‘weighting factors’, the data values of a ‘selected tuple’, and the data values corresponding to some ‘selected color range’. The possibility of getting the values corresponding to some color or color range for each selection predicate may help the user to understand and interpret the visualization and to modify the query accordingly. To focus on sets of data items with a specific color, it is possible to select some color range in one of the sliders to get only those data items in the corresponding visualization window that have the selected color for the considered attribute. In the other visualization windows, the same data items are displayed, allowing the user to easily compare the values of the other attributes of those data items. Also helpful for the user to understand the visualization and to find interesting data spots is the possibility to select a specific data item in one of the visualization windows, highlight it in all visualization windows, and display the values for the attributes in the ‘selected tuple’ field. The user may use this option to focus on exceptional data items or to get an example of a data item from an interesting region in one of the windows. Below the color spectrum for the overall result, there are fields for the number of data items in the database, the number of data items being displayed in the visualization window (absolute value and percentage) and the number of resulting data items presented to the user. Using a slider, the user may change the percentage of data being displayed or the allowed range, in which case the percentage is determined using the heuristics described in “Appendix: Calculating the Relevance Factors”. Changing the percentage of data being displayed may completely change the visualization since the distance values are normalized according to the new range.

In the normal mode, the system recalculates the visualization after each modification of the query. The user may also switch to an ‘auto recalculate off’ mode where queries are only recalculated on demand. This option is useful for large databases with many data items or if complex distance functions are used, because the re-calculation for each modification may need a considerable amount of time. Other menu options allow the user to choose different distance or combinator functions, to select a different visualization technique or a different slider type, to add or delete selection predicates, to extend the query, or to issue a new query.

5 Examples

In figures 5 - 8, several visualizations of query results are displayed. Figure 5 has been generated by using surface point data from a large molecule complex (subtilisin carlsberg with eglin). In our molecular biology project, the *VisDB* system has been used to find possible docking regions by identifying sets of surface points with distinct characteristics. In evaluating our visualization techniques, we currently explore other data sets including a large database of geographical data, a large environmental database, a NASA earth observation database, and artificially generated data sets. Artificial data sets are crucial for comparing different visualization techniques to find their strengths and weaknesses [BKP 94]. Being able to vary the number of data items, the number of dimensions, and the properties of the data (e.g. distribution of each dimension and the number and size of clusters) is important for doing controlled comparisons. The visualizations displayed in figures 6 - 8 use artificially generated data consisting of a uniformly distributed base data set and multiple clusters. The data set used for figure 6 consists of 1,000 eight-dimensional data items with five clusters. The data set used for figure 7 is similar except that it consists of 7,000 data items. Figure 8 is generated from a database with 100,000 five-dimensional data items containing five clusters. In the visualization, many regions of different colors are clearly identifiable and denote clusters of data items with a comparable distance. Interesting are the correlations between the windows for the different selection predicates. Often regions that have some specific color in the section for one selection predicate have some different color in the section for another selection predicate. Sometimes even an inverse correlation can be found, as in case of the green versus red regions for selection predicates 2 and 3 in figure 8. Another interesting observation in comparing the visualizations a and b in figures 7 and 8 is that colored regions of the basis visualization technique often cluster in one quadrant of the 2D-arrangement (c.f. brown region for selection predicate 8 in figure 7). This provides additional information on the position of the cluster with respect to the two dimensions that have been assigned to the axes and may help the user in modifying the query. Also interesting, but not easily identifiable in the printed version of our visualizations are hot spots, i.e. single exceptional data items in regions which are otherwise homogeneous. Much of the information the user may get out of the visualization is related to the semantics of the data. Due to space limitations we do not elaborate on these aspects, since we would have to introduce the schema and the instances of the databases used; in the case of the artificial data sets, this would require at least a specification of the base data set and all clusters.

Our query and visualization system is not only useful for data mining tasks such as finding correlations between different dimensions, finding groups of similar data, and finding hot spots, but also for other tasks such as similarity retrieval, finding adequate query parameters and weighting factors, or finding correspondences in different databases. Finding similar parts in a large CAD database is an example for the first two of these tasks. In a CAD database of 3D-parts, it is not obvious how similarity can be formally described. Usually, there are many parameters (in a concrete application in mechanical engineering we had 27 parameters) describing the parts, and each of them might be important for a part to be similar. In searching for similar parts in traditional CAD databases, a query is issued by using fixed allowances for some of the parameters. As a result of the query, the user only gets the information concerning whether a data item fulfills all allowances or not. However, the user might miss a part that exactly fits in all but one parameter and just misses fulfilling the query due to that single parameter. Therefore, in similarity retrieval, it seems to be important to provide approximate responses and to allow the user to adjust the allowances and weighting parameters. Our system provides features that exactly support these tasks, making it a promising candidate to be used in similarity retrieval. Another example of an interesting application of our system is in multi-database systems where it is often a problem finding corresponding data items in multiple independent databases. If a distance function for the two attributes to be joined can be defined, our system may help the user to identify closely related data items and to find adequate parameters for approximately joining the databases.

6 Visualizing the Results of Complex Queries

In addition to simple one-table queries with all selection predicates being connected by the same Boolean operator, our visualization techniques have also been used to support complex queries i.e. queries with the selection predicates being arbitrarily connected (nested ANDs and ORs), multi- table queries, and some types of nested queries (for details see [KKS 94]). Complex queries are supported by using multiple layers of windows for different parts of the query, giving the users visual feedback for each part of the query and helping them to understand the overall result. This is sufficient for queries with nested Boolean operators, but in order to support multi-table and nested queries, a mechanism for joining tables and dealing with the cross product is necessary.

Our idea for supporting multi-table queries is to consider all data items of the cross product that approximately fulfill the join condition. As for all other selection predicates, the user obtains a separate window for the join condition with all data items of the cross product that fulfill the join condition being yellow and the others being colored according to their distance. In some cases, e.g. if the tables are connected by foreign keys which are designed to connect related data items, this may not be helpful because the distances on foreign keys may not have any semantics. In such cases, only those data items that fulfill the join condition are considered and no visualization for the join condition is generated. In many other cases, however, it is helpful to consider data items that approximately fulfill join conditions as well. For joins on numerical attributes, for example, the numerical difference between the considered data items from the two relations is used as an approximation of the join condition to be fulfilled. In a similar way, the distances for non-equijoins ($a1 < a2$) or parametrized (non-equi)joins ($a1 - a2 < c$) are determined.

In the case of nested queries, separate visualizations are provided for each of the selection predicates including the subqueries involved. In the visualization corresponding to the overall result of a subquery, the user gets yellow in case the subquery condition is fulfilled and otherwise the color corresponding to the distance of the data item most closely fulfilling the subquery condition. The data item most closely fulfilling the subquery condition can be determined by the minimum distance in performing an approximate join of the inner and the outer relation(s). Instead of displaying a single value for the whole subquery, there might be an option to select a single data item and get the complete subquery with all its selection predicates including the join of inner and outer relation(s) presented in a separate window.

7 Implementation

The visualizations presented in figures 5 - 8 are screen dumps from working with the *VisDB* system. The *VisDB* system is implemented in C++/MOTIF and runs under X-Windows on HP 7xx machines. The current version is main memory based and allows an interactive database exploration for databases containing up to 50,000 data items (on HP 735 workstations). This is very encouraging since we have not yet spent time optimizing our algorithms. 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 with the database system. In the future, we plan to implement the *VisDB* system on a parallel machine that will be able to support interactive query modifications even for mid-size to large amounts of data and complex distance functions.

8 Future Extensions

Inspired by using our prototype, we already have several ideas to extend our system. One extension is the automatic generation of queries that correspond to some specific region in one of the visualization windows. The region may be graphically identified by the user. The system should then try to find adequate selection predicates that provide the desired data items as a result. Another idea is to generate time series of visualizations corresponding to queries that are changed incrementally. By changing the query, different portions of multidimensional space can be visualized, allowing even larger amounts of data to be displayed. To further improve our system, we intend to apply it to many different application domains, each having its own parameters, distance functions, query requirements and so on. In addition to real world data, we will also use artificially generated data sets which allow controlled studies on the effectiveness of our visualization techniques.

9 Conclusions

One of the big challenges that researchers in the visualization area are currently facing is to develop visualization techniques that are adequate to explore very large amounts of arbitrary multidimensional data. The task is to efficiently retrieve interesting data sets, i.e. hot spots, clusters of similar data, or correlations between different dimensions. Our approach to support these 'data mining' tasks combines traditional database querying and information retrieval techniques with new techniques of visualizing the data. Our *VisDB* system allows visualization of the largest amount of data that can be displayed at one point of time on current displays, providing valuable feedback in querying the database and allowing the user to find results which would otherwise remain hidden in the database. The interactivity of the system allows focusing on interesting data, providing a promising way to explore databases efficiently.

We believe that query and visualization systems like ours are of high value for many applications. They may be the starting point for new ways to visually solve problems that have proven to be very difficult. Querying of large databases is just one example.

Acknowledgments

At this point, we want to thank all people who contributed to the *VisDB* system, especially Thomas Seidl who implemented the first prototype of the system and Juraj Porada who implemented the current version of the *VisDB* system.

References

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

[BMS 92] Bergeron R. D., Meeker L. D., Sparr T. M.: *'A Visualization-Based Model for a Scientific Database System',* in: Focus on Scientific Visualization, eds: Hagen H., Müller M., Nielson G., Springer, 1992, pp. 103-121.

[BKP 94] Bergeron D., Keim D. A., Pickett R.: *'Test Data Sets for Evaluating Data Visualization Techniques',* in: Perceptual Issues in Visualization, Springer, 1994.

[Ber 81] Bertin J.: *'Graphics and Graphic Information Processing'*, Berlin, 1981.

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

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

[JKL 77] Joshi A. K., Kaplan S. J., Lee R. M.: *'Approximate Responses from a Data Base Query System: Applications of Inferencing in Natural Language'*, Proc. 5th Int. Joint Conf. on Artificial Intelligence (IJCAI), Boston, MA, 1977, pp. 211-212.

[Kap 82] Kaplan S. J.: *'Cooperative Responses from a Portable Natural Language Query System'*, Artificial Intelligence, Vol. 19, 1982, pp. 165-187.

[KKS 94] Keim D. A., Kriegel H.-P., Seidl T.: *'Supporting Data Mining of Large Databases by Visual Feedback Queries'*, Proc. 10th Int. Conf. on Data Engineering, Houston, TX, 1994, pp. 302-313.

[KL 92] Keim D. A., Lum V.: GRADI: *'A Graphical Database Interface for a Multimedia DBMS'*, Proc. Int. Workshop on Interfaces to Database Systems, Glasgow, England, 1992, in: Lecture Notes in Computer Science, Springer, pp. 95-112.

[HL 92] Herman G.T., Levkowitz H.: *'Color Scales for Image Data'*, Computer Graphics and Applications, 1992, pp. 72-80.

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

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

[Mot 90] Motro A.: *'FLEX: A Tolerant and Cooperative User Interface to Databases'*, IEEE Trans. on Knowledge and Data Engineering, Vol. 2, No. 2, 1990, pp. 231-246.

[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.

[PG 88] Pickett R.M., Grinstein G.G.: *'Iconographic Displays for Visualizing Multidimensional Data'*, Proc. IEEE Conf. on Systems, Man and Cybernetics, Beijing and Shenyang, China, 1988.

[Shn 94] Shneiderman B.: *'Dynamic Queries for Visual Information Seeking'*, to appear in: IEEE Software, 1994.

[SM 83] Salton G., McGill M. J.: *'Introduction to Modern Information Retrieval'*, McGraw-Hill, New York, 1983.

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

Appendix: Calculating the Relevance Factors

Calculating the Distance

The first step in calculating the relevance factor for each data item is to determine the distance between attribute and corresponding query values. The distance functions used in this step are data type and application dependent. In some cases, even for a single data type multiple distance functions may be useful. For number 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 non-metric types such as enumerations with a non-interpretable distance between values (ordinal types e.g. *grades*) or with non-comparable values (nominal types e.g. *professions*), there is no obvious way to determine the distance. For ordinal types, the distance may be defined by some domain-specific distance function or by a distance matrix containing the distances for all pairs of values. A distance matrix may also be useful for nominal types but, in some cases, even a constant value may be an adequate distance. For the data type *string,* there are many possibilities to calculate the distance. Depending on the application and the context of the retrieval, the user may want to choose between lexicographical difference, character-wise difference, substring difference or even some kind of phonetic difference.

Combining Distances into the Relevance Factor

The next step in calculating the relevance factor is the combination of the independently calculated distances of the different selection predicates. This, however, is not straightforward because the distances for the different selection predicates have to be considered with respect to the distances of the other selection predicates, and the combined distance must be defined and must be meaningful globally. One problem is that the relative importance of the multiple selection predicates is highly user and query dependent. This problem can only be solved by user interaction since only the user is able to determine the priority of the selection predicates. Therefore, it is necessary to obtain weighting factors (w_j , $j \in 1, \ldots, \#sp$) representing the order of importance of the selection predicates from the user. A second problem is that the values calculated by the distance functions may be in completely different orders of magnitude (e.g. in a medical application, a distance of 1g/dl for Haemoglobin may be very high and a distance of 1,000 Erythrocytes per dl may be very small). This problem can be solved by a normalization of the distances. A simple normalization may be defined as a linear transformation of the range $[d_{min}, d_{max}]$ for each selection predicate to a fixed range (e.g. [0, 255]). For combining the independently calculated and normalized distances of multiple selection predicates into a single distance value, we use numerical mean functions such as the weighted arithmetic mean for 'AND'-connected condition parts and the weighted geometric mean for 'OR'-connected condition parts. More exactly, for each data item x_i the combined distance is calculated as:

$$\text{Combined Distance}_i = \sum_{j=1}^{\#sp} w_j \times d_{ij} \quad \text{in case of 'AND'},$$

$$\text{Combined Distance}_i = \prod_{j=1}^{\#sp} d_{ij}^{\,w_j} \quad \text{in case of 'OR'}.$$

After calculating the combined distance for the whole condition, the relevance factor is determined as the inverse of that distance value. The relevance factor combines the information on how well a data item approximates the query into one value representing the relevance of the data item with respect to the query. At this point it should be mentioned, that for special applications other specific distance functions such as the Euclidean, L^p or the Mahalanobis distance in n-dimensional space may be used to combine the distance values of multiple selection predicates.

Reducing the Amount of Data to be Displayed

Since the number of data items in the database may be much higher than the number of data items that can be displayed on the screen, we had to find adequate heuristics to reduce the amount of data and to determine the data items whose distance should be displayed. The most exact way is to use a statistic parameter, namely the α-quantile. The α-quantile is defined as the lowest value ξ_α such that

$$F(\xi_\alpha) = \int_{-\infty}^{\xi_\alpha} f(x)\,dx = \alpha ,$$

where $0 \le \alpha \le 1$, *F(x)* is the distribution and *f(x)* the density function.

Let r be the number of distance values that fit on the screen, #sp be the number of selection predicates, and n be the number of data items in the data base. Then only data items with an absolute distance in the range $[0, \frac{r}{n \times (\#sp + 1)}$-quantile] are presented to the user. If negative and positive distance values are used, then the range of values presented to the user is given by [α_0*(1-p)-quantile, (α_0*(1-p) + p)-quantile] where $p = \frac{r}{n \times (\#sp + 1)}$ and α_0 is determined by α_0-quantile = 0. In the special case of two dimensions assigned to the two axes (c.f. section 3.2), correspondingly the combined α-quantiles for two dimensions may be used. For the grouping arrangement (c.f. section 3.3), the number of data items that can be displayed on the screen is lower since multiple pixels are needed per data value.

Appendix: Coloration of the Relevance Factors

Visualizing the relevance factors using color corresponds to the task of mapping a color scale to a single parameter distribution. The advantage of color over gray scales is that the number of just noticeable differences (JNDs) is much higher. The main task is to find a path through color space that maximizes the number of JNDs but, at the same time, is intuitive for the application domain [HL 92].

In designing the system, we experimented with different colormaps. We found that the coloration has a high impact on the intuitivity of the system. The user, for example, may implicitly connect good answers with light colors and bad answers with dark colors, or the user may be accustomed to green colors for good answers and red colors for bad answers (like the colors used for traffic lights). We tried many variations of the colormap to enhance the usefulness of our system and experimentally found that for our application, a colormap with quite constant saturation, an increasing value (intensity) and a hue (color) ranging from yellow over green, blue and red to almost black are a good choice to denote the distance from the correct answers. The color model used in the *VisDB* system is a variation of the HSV model. Instead of the hexcone used in the HSV model, we use a circular cone with the intensity being defined as the Euclidean distance to black and the saturation being defined as the Euclidean distance to the gray axis. In the HSV model, both parameters are determined by using the maximum of (r, g, b). In contrast to color scales generated according to the HSV model, our model provides color scales whose lightness ranges continuously from light to dark colors.

Since the usefulness of colormaps varies depending on the user and the application, we allow the users to define their own colormaps and use them instead of our standard colormap.

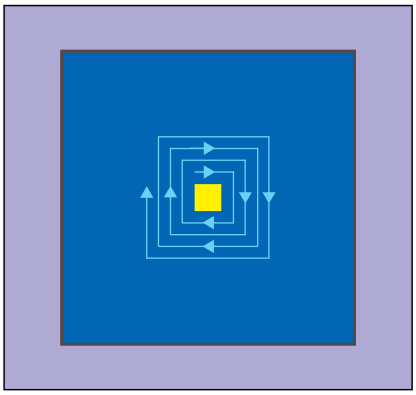


Figure 1:
Spiral Shaped Arrangement
of one Dimension

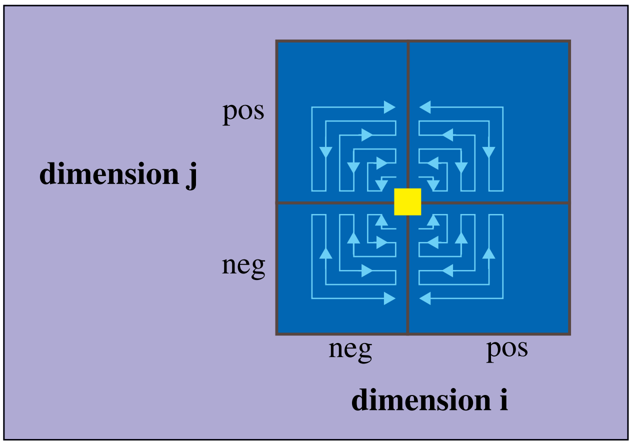


Figure 3: 2D-Arrangement of one Dimension

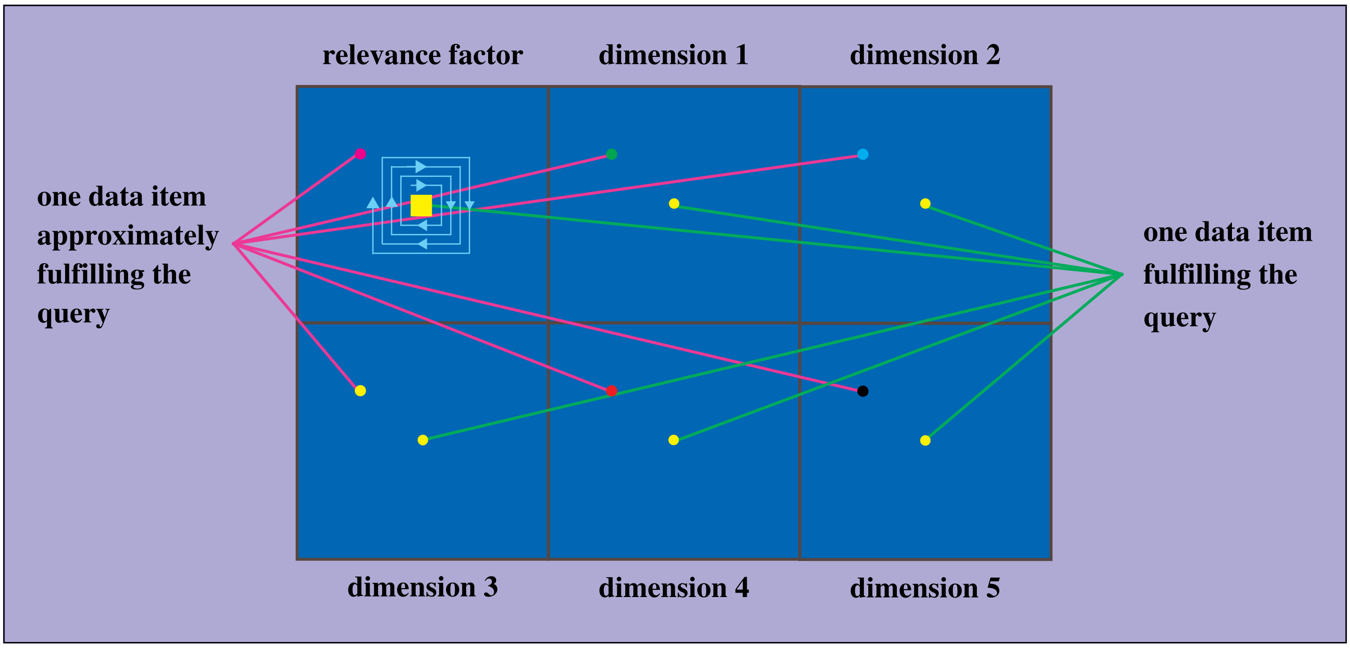


Figure 2: Arrangement of Windows for Displaying five-dimensional Data

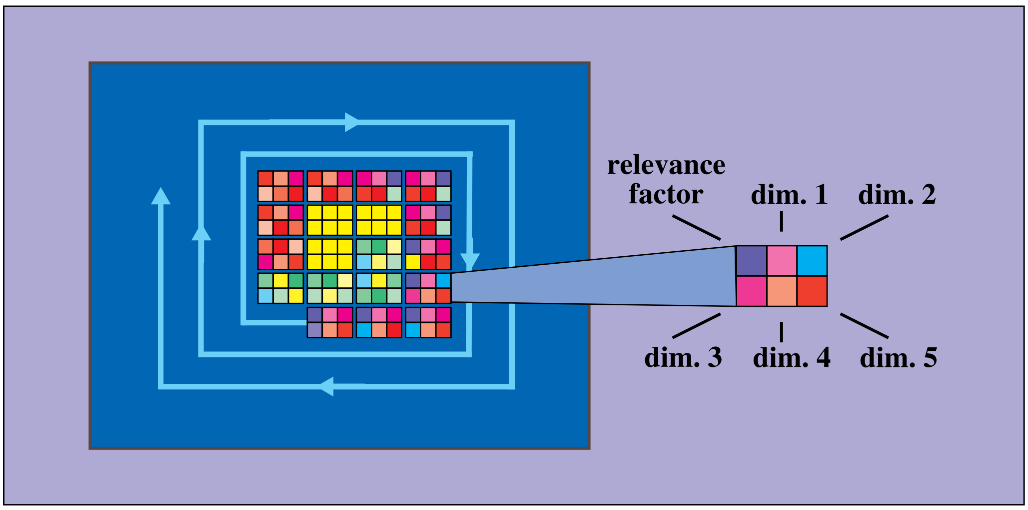


Figure 4: Grouping Arrangement for five-dimensional Data

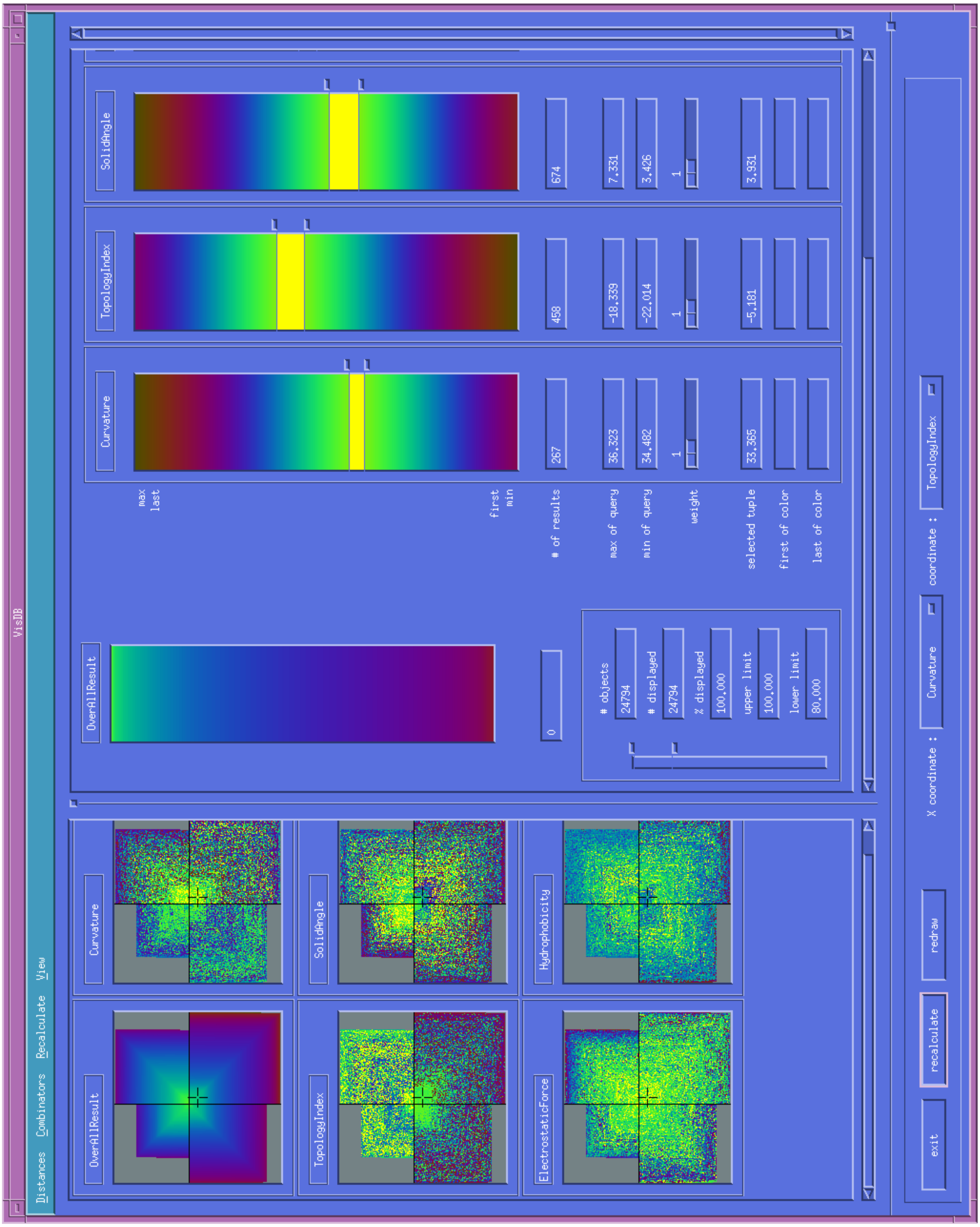
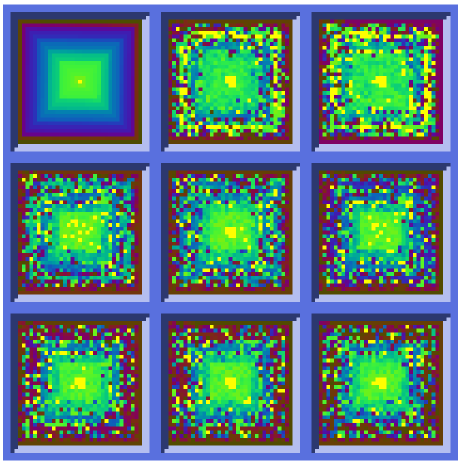
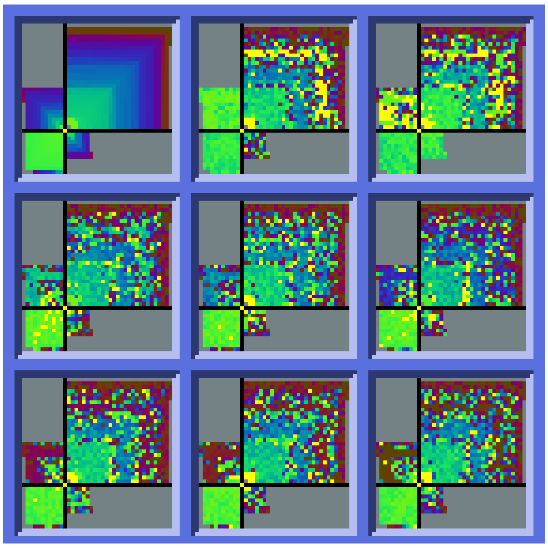


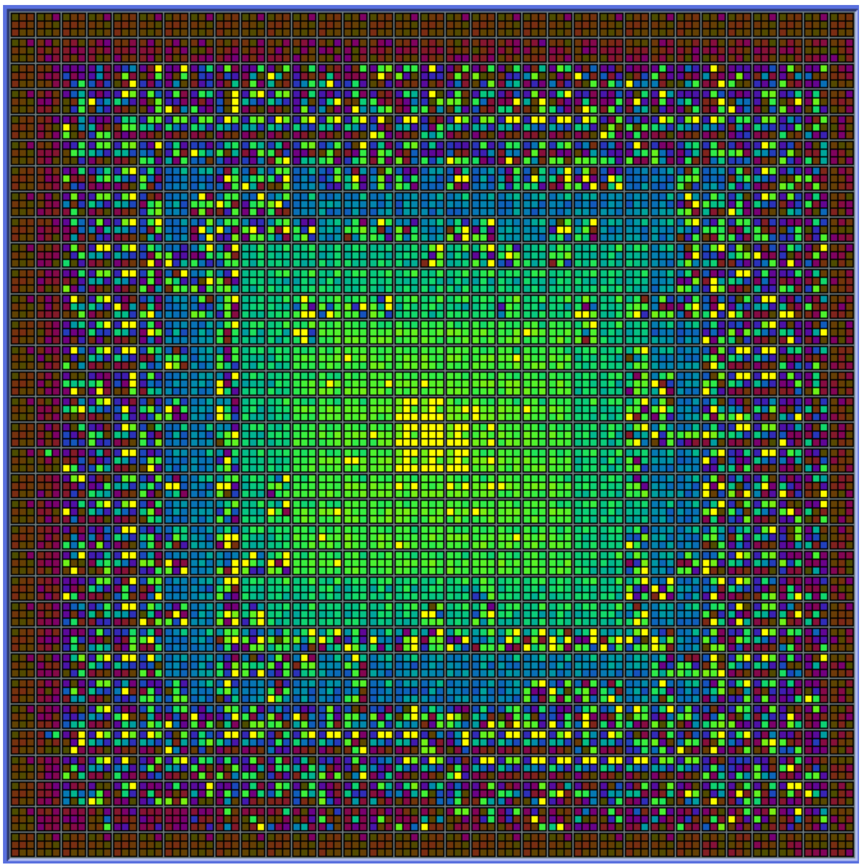
Figure 5: The *VisDB* System



a. Basic Visualization Technique

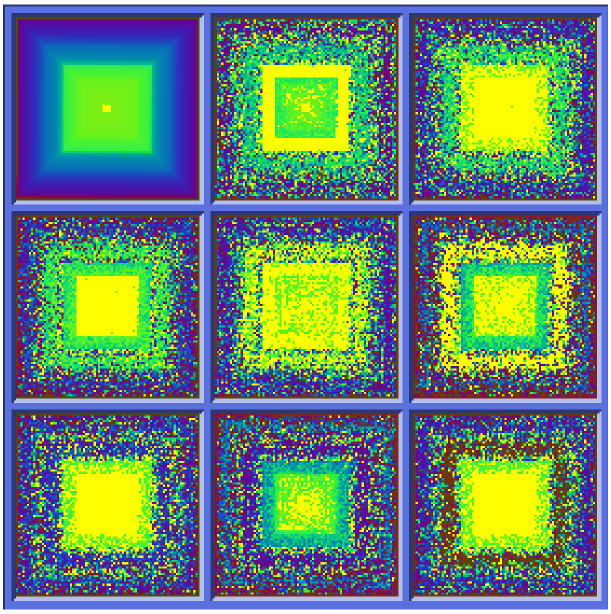


b. 2D-Arrangement

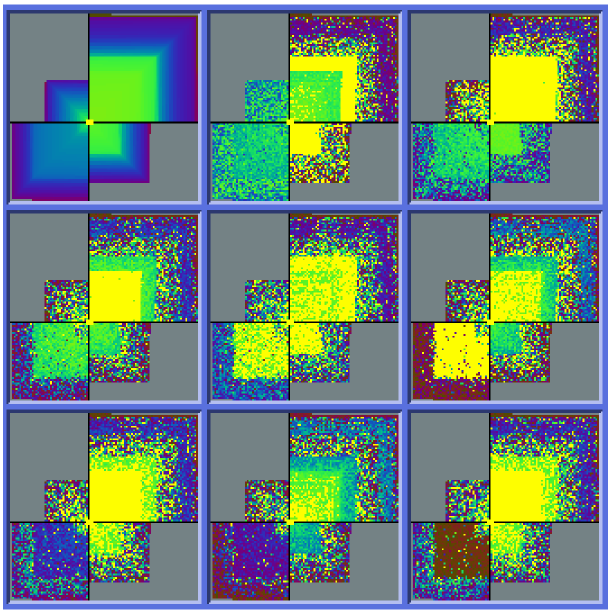


c. Grouping Arrangement

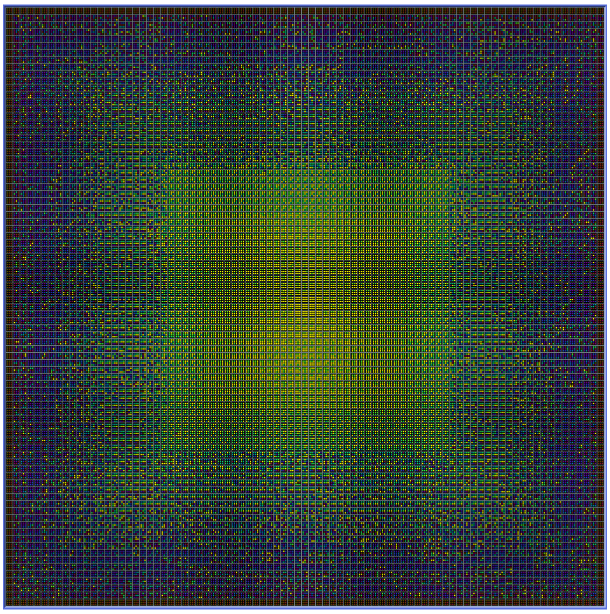
Figure 6: Eight-dim. data displayed with the three different Visualization Methods (1000 Data Items)



a. Basic Visualization Technique

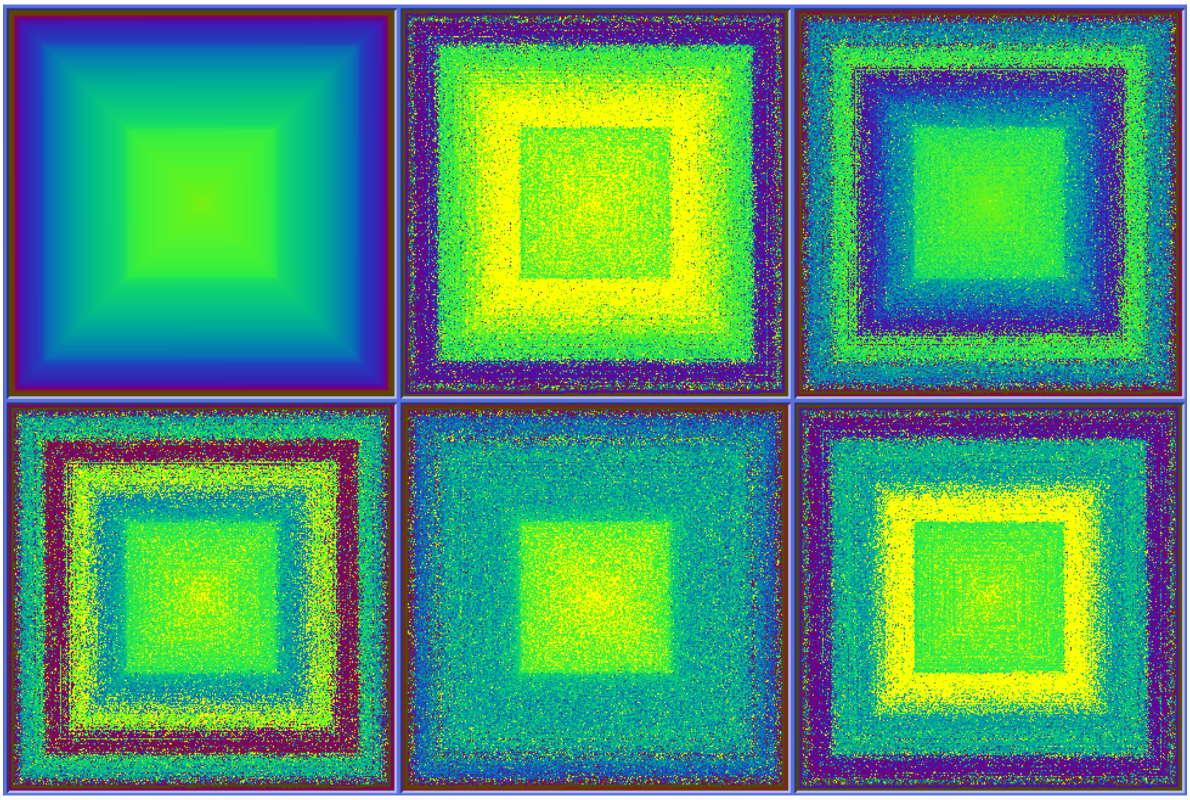


b. 2D-Arrangement

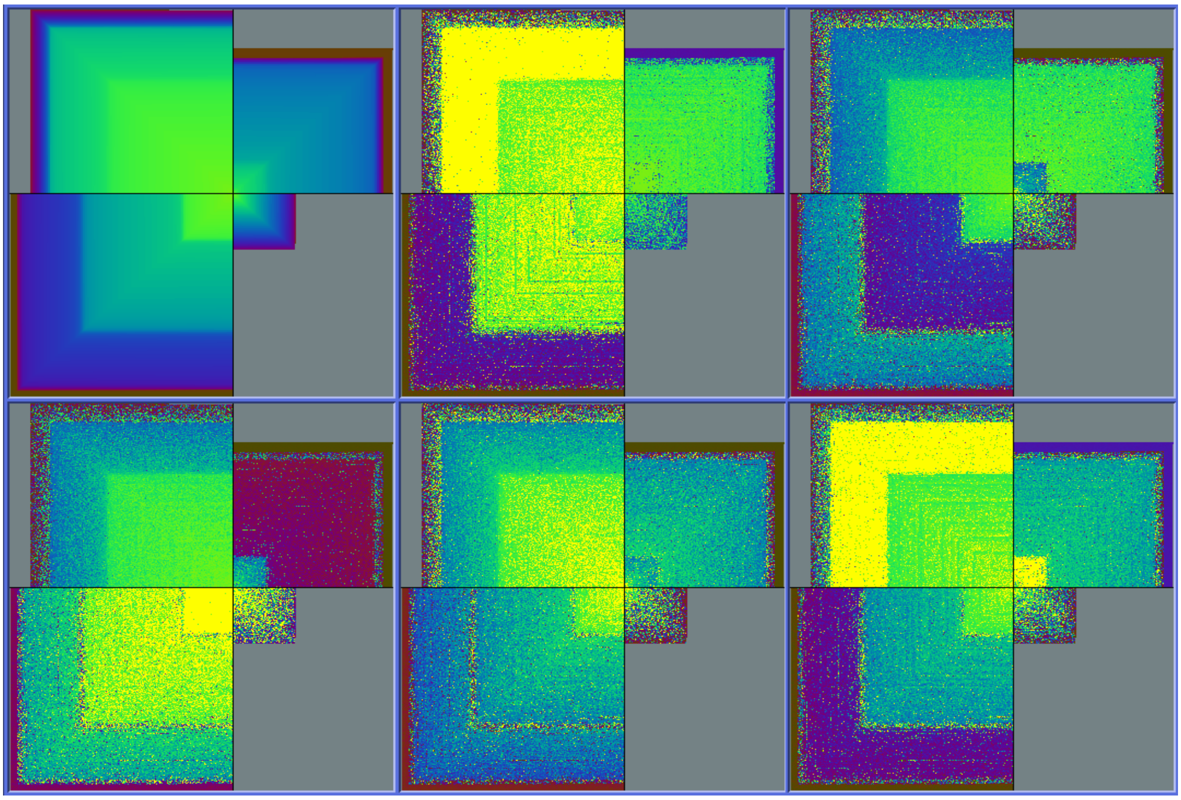


c. Grouping Arrangement

Figure 7: Eight-dim. Data displayed with the three different Visualization Methods (7000 Data Items)



a. Basic Visualization Technique



b. 2D-Arrangement

Figure 8: Five-dimensional Artificially Generated Data Items (100,000 Data Items)