An Interactive Visual Framework for Detecting Clusters of a Multidimensional Dataset*

Debangshu Bhadra and Ashim Garg

Department of Computer Science and Engineering

201 Bell Hall

University at Buffalo

Buffalo, NY 14260

{dbhadra, agarg}@cse.buffalo.edu

Abstract

A key problem in data mining applications is that of detecting clusters, i.e., groups of closely related data items, of a multidimensional dataset. Users are adept at detecting clusters in visually presented information. We, therefore, present an interactive framework that allows a user to visualize a multidimensional dataset, and detect clusters in it by repeatedly changing its visualization using interactive operations such as parameter re-adjustment, zooming, subset selection, and cluster probing. Our experiments also suggest that animation provides important clues in detecting clusters.

1 Introduction

Data Mining, also called *Knowledge discovery in databases*, can be defined as the extraction of previ-

*Research supported in part by the NSF CAREER Award Number IIS-9985136.

ously unknown and potentially useful information from data in databases [22]. Data mining enables the user to explore a database, and discover more meaningful and higher-level relationships between its data-items. Data mining not only helps the user in discovering new knowledge, but also allows her to use the database more effectively in problem-solving tasks. Data mining has numerous applications such as in product development, software engineering, and process control. The growth of the Web has also increased the importance of data mining, because the Web is not only a very large data-repository, which is difficult to search, but also provides several applications, such as developing user-profiles for improving on-line services, where data mining can be useful. As a result, researchers from various fields such as databases, artificial intelligence, and visualization, have worked on data mining problems. Several commercial establishments have also developed data mining tools.

Clustering is a technique commonly used in data mining. It can be defined as the process of grouping physical or abstract objects into classes of similar objects [3]. Clustering imposes a higher-level structure on a dataset by partitioning its objects into classes. This helps the user in understanding and interpreting the dataset.

Given a dataset, we can view its data-items to be distributed in a multidimensional *data space*, whose each dimension correspond to an attribute of the data-items. Typically, the data-items would be distributed non-uniformally in the data space. We may have several pockets of closely placed data-items, as well as, large empty spaces. We can express the problem of detecting clusters of the data set as that of detecting pockets of closely placed data-items in the data space. Detecting these pockets is the goal of the interactive visualization-based technique that we describe in this paper.

Visualization presents information in a visual form so that it can be perceived by the user. Card, Mackinlay, and Schneiderman [2] define *Information Visualization* as: "the use of computer-supported, interactive, visual representation of abstract data to amplify cognition", where cognition is defined as "the acquisition or use of knowledge". Apart from anecdotal evidences, formal studies [17] have also supported the notion that information visualization can help in cognition. Visualization takes advantage of the perceptual capabilities of the user to help in cognition. It helps the user in understanding information in several ways [2]: it allows the user to recognize patterns perceptually, to make inferences perceptually (On the map of USA, Boston is to the right of Chicago, Minneapolis is to the left of Chicago,and all three are roughly at the same latitude. So, we can infer that Boston is further from Minneapolis than is Chicago), to monitor rapidly changing information (such as network traffic) perceptually, to manipulate information through its visualization (changing a UML diagram to change the underlying program), and to localize search (If I am in Brooklyn, and looking for a gas-station, I can just look up at the map of Brooklyn for the nearest gas-station). Special techniques such as separation and layering, dynamic queries [27], focus+context displays [8, 23], and zooming [1] can also be used to enhance the effectiveness of a visualization. Because of its advantages, information visualization has been used as an aid in problem-solving tasks in a variety of applications, such as software engineering [19], social-network analysis, and Web-navigation [19].

Detecting clusters of a dataset is a challenging problem. Part of the challenge is based on the fact that it is difficult to formally define a cluster. On the other hand, users are adept at identifying clusters in visually presented information. Hence, we would like to take advantage of this perceptual capability of the user to help them detect clusters of a dataset.

However, visualizing a dataset is also a challenging problem. Assuming the availability of a two-dimensional display surface, such as computer screen, we can think of visualization as a mapping of data-items from the multidimensional data space to a two-dimensional *visual space*, where each data-item is mapped to a pixel of the visual space. Since, we are mapping from a higher dimensional space to a lower dimensional space, this mapping may not preserve the properties of the dataset. In particular, it is possible that this mapping may "break" a cluster, i.e., map its data-items to distant pixels. It may also create "false" clusters on the display, by mapping distant data-items to nearby pixels. This is partly the reason, why "static" visualization techniques, such as scatter-plots, which focus on creating a single visualization may fail to detect clusters. It is possible that the visualization that they create may break clusters, or create false clusters, and hence, users may not be able to clearly identify the "real" clusters of the data-set from the visualization.

We believe that user-interactivity plays an important role in creating visualizations that are effective in revealing clusters of a dataset. Users should be allowed to repeatedly change and improve the visualizations in real-time, and in the process, discover the clusters of dataset. This makes cluster-detection, a user-driven process, where the user actively guides the visualization system to reach her goal. This aspect of active user-participation is missing in static visualization techniques, which reduces their effectiveness. Several researchers in data mining [3, 21] have commented on the benefits of making data mining, a user-driven process, where the user guides and controls the knowledge discovery process. Our interactive visual framework, which we describe in this paper, is a step in that direction, with focus on detecting clusters.

We have implemented a prototype system based on our framework. Our experiments done using this system also suggest that animation can also provide important clues about clusters. As visualizations are changed in real-time by the user, the data-items also change their positions on the screen in real-time. This creates an animation on the screen. In this animation, clusters typically behave like semi-rigid bodies moving from the old positions to the new positions. This behavior helps the user in identifying them.

Our main contribution is in defining and implementing a comprehensive interactive framework for detecting clusters through visualization. Also, to the best of our knowledge, we are the first to use animation to detect clusters.

The rest of the paper is organized as follows: In Section 2, we describe our framework for interactive visualization of clusters. In Section 3, we present the experimental results. In Section 4, we discuss the role of animation in cluster detection further. In Section 5, we discuss related work on cluster-detection. Finally, in Section 6, we present our conclusions and future work.

2 Our Interactive Visual Framework for Cluster Detection

We present an interactive visual framework for detecting clusters of a dataset. Our framework allows the user to interactively change the visualization of a dataset, and in the process, detect clusters. It provides four interactive operations through which the user can change the visualizations. These are: *Parameter Re-adjustment*, *Zooming*, *Subset Selection*, and *Cluster Probing*.

```
The Interactive Visual Framework:
1. Normalize the dataset so that the attributes of each data-item have
      values between 0 and 1;
2. Create an initial visualization;
3. User Interaction Loop:
      Repeat {
         Change the visualization by applying one of the four interactive
         operations, namely, parameter re-adjustment, zooming, subset
         selection, and cluster probing;
      } Until (clustering information is clear).
```

The *user-interaction loop* given above is the heart of our interactive visual framework. The user first creates an initial visualization of the dataset, and then repeatedly changes the visualization using the four above-mentioned interactive operations. This process continues until the clusters become clear to the user. Thus, even if the clusters are not apparent to the user in the initial visualization, during subsequent iterations of the interaction-loop, they become clearer to her, because of the improvements made by her to the visualization. Also, as shown in Section 4, the animation created by the continuously changing visualization also helps in detecting clusters. It is this aspect of our approach, which makes it more effective for detecting clusters than the static visualization techniques.

We now describe in detail, the construction of the initial visualization, each of the four interactive operations, and the roles they play in detecting clusters. We also describe how they are implemented by our prototype system.

2.1 Initial Visualization

The initial visualization is created by a mapping, which maps the data-items from the multidimensional data space to the two-dimensional visual space.

Formally, let D be an N-dimensional dataset with dimensions denoted by $x_1, x_2, \ldots, x_n$, respectively. Let $x_i(p)$ denote the value of the x_i^{th} coordinate of a point p in the data space. We assume that $N \geq 3$ (for $N < 3$, since the visual space is two-dimensional, the visualization problem is trivial). Let $p(a_1, a_2, \ldots, a_N)$ denote a point p such that the value of the x_i^{th} coordinate of p is equal to a_i. Let S be the two-dimensional visual space, that corresponds to the display surface. Let the two axes of S be called X- and Y-axes, respectively. Let $x(q)$, and $y(q)$ denote the X, and Y coordinates, respectively, of a point q of S. Let $q(a, b)$ denote a point q such that $x(q) = a$, and $y(q) = b$. The initial visualization is a mapping $f : D \rightarrow S$, which maps each data-item $p(x_1, x_2, \ldots, x_N)$ of D to a point $q(x, y)$ of S. A visualization is *parameterized*, if the mapping contains parameters, whose values can be set by the user. The user can change the visualization by changing the values of its parameters. Since, the visualization should change in real-time after each parameter re-adjustment, the mapping should be fast to compute.

We now describe the mapping f used by our prototype system. Since the dataset is first normalized, for each data point p of D, $0 \leq x_i(p) \leq 1$, for each i, where $1 \leq i \leq N$. Let $\vec{ab}$ denotes a vector from point a to point b. f maps each data-item $p(x_1, x_2, \ldots, x_N)$ of D to a point $q(x, y)$ of S, as follows: Let χ be a circle of S with center c and radius a. (The value of a depends upon the area of the display surface.) Let $s_1, s_2, \ldots, s_N$ be N equidistant points on the circumference of χ (see Figure 1). Let

$\alpha_1, \alpha_2, \ldots, \alpha_N$ be constants, where each α_i has a value between -1 and 1. Let q be a point inside circle χ, such that $\vec{cq} = \sum_{1 \leq i \leq N} \alpha_i (4/N)(x_i(p) - 0.5)\vec{cs}_i$. f maps p to point q.

This mapping can be interpreted as follows: It maps the point $(0.5, 0.5, 0.5, \ldots, 0.5)$ of D to the center c of χ. $\alpha_i(4/N)(x_i(p) - 0.5)\vec{cs}_i$ is the vector from c to s_i scaled by $\alpha_i(4/N)(x_i(p) - 0.5)$. $\sum_{1 \leq i \leq N} \alpha_i(4/N)(x_i(p) - 0.5)\vec{cs}_i$ is the resultant vector $\vec{v}$ of the scaled vectors from c to each s_i. q is the point reached from c by following $\vec{v}$. Since $|\alpha_i| \leq 1$, $0 \leq x_i(p) \leq 1$, and $|\vec{cs}_i| \leq a$, we can show that the magnitude of the resultant vector is at most equal to a. Thus, q lies within the circle χ.

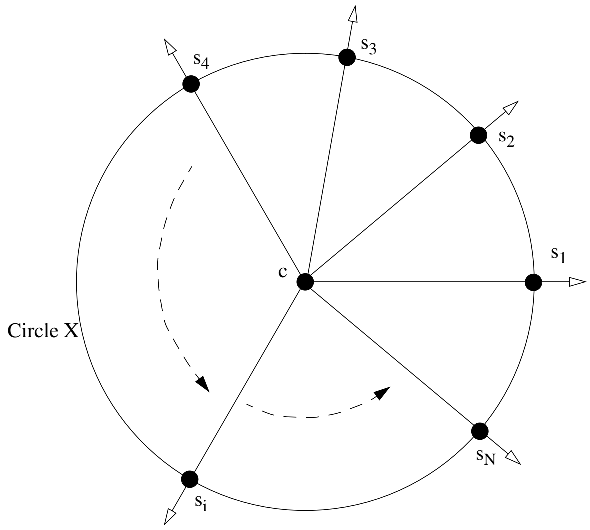


Figure 1: The mapping used by our prototype system for creating its visualizations. Here, χ is a circle with center c and N equidistant points $s_1, s_2, \ldots, s_N$ on its circumference.

$\alpha_1, \alpha_2, \ldots, \alpha_N$ are the N parameters of this mapping. The users can change the visualization by changing the values of these parameters.

This mapping is easy to implement, and as shown by our experiments, works well within our interactive framework for detecting clusters. It is fast to compute, and hence, creates new visualizations quickly on parameter re-adjustments, and was able to reveal the clusters of the datasets used in our experiments.

2.2 Interactive Operations

2.2.1 Parameter Re-adjustment

The parameter re-adjustment operation allows the user to improve a visualization by changing its parameters. Our prototype system provides several sliders, one for each parameter, through which the user can change the values of the parameters (see Figure 2).

Each parameter re-adjustment should change the visualization in real-time. In fact, during a continuous sequence of parameter re-adjustments, to give the perception of smooth and continuous change in the visualization to the user, the new visualizations should be created in less than 100 milliseconds [27]. Thus, it is important that the mapping be one that can be computed quickly. Our experiments suggest that the simple linear mapping used by our prototype system fits this criteria.

2.2.2 Zooming

Zooming [1] allows the user to magnify a portion of the visualization, and view it in greater detail. It may help the user in detecting clusters that may be hidden under other clusters in the visualization. In Section 3, we describe a similar scenario, where zooming helped in discovering a cluster hidden inside another cluster.

In our prototype system, we provide a slider, which allows the user to zoom into, and out of the visualization. After each zooming operation, the visualization should be updated within 100 milliseconds to give a perception of smooth and continuous change to the user.

2.2.3 Subset Selection

If the dataset being visualized is too large, it can create two kinds of problems, which may make it difficult to detect clusters:

1. it may cause an information overload on the user's perceptual and cognitive system, and

2. it may make it difficult to smoothly and continuously update the visualization after a parameter re-adjustment or zooming operation.

Subset selection allows the user to select a subset of the dataset, and detect clusters in it. It helps the user to concentrate only on one part of the dataset, reducing the processing load on both her and the visualization system. It also enables the user to explore the dataset in a divide-and-conquer fashion.

We provide two mechanisms through which the users can perform subset selection:

- *SQL Query:* The users can give an SQL-Query for selecting data-items that satisfy the query. Our prototype system is currently "linked" to an Oracle database-server, which can store a copy of the dataset being visualized. The users can give an SQL-Query through the user-interface of the system. This query will be processed by the database-server, and data-items satisfying the query will be returned to the system.

- *Visual Selection:* The users can also select data-items visually from the current visualization. In our prototype system, they can drag the mouse, create a closed curve on the display (see Figure 2(c)), and click on the "select" button. This will select the data-items lying in the region bounded by the curve.

2.2.4 Cluster Probing

Cluster probing helps the user in determining if a cluster visible on the display is, indeed, a real cluster of the dataset. We provide two kinds of cluster probing operations:

- *Detail-on-demand* [24]: The user can click on any pixel on the display, and see the values of the attributes of the data-items mapped on that pixel (see Figure 2(h)). Our prototype system displays these values in a separate pop-up window. Thus, to probe a cluster visible on the display, the user can randomly select, one-by-one, several pixels in it, and see the values of the attributes of the data-items mapped on to those pixels. If the attributes of the data-items have similar values, then it gives an indication that the cluster is also a real cluster of the dataset.

- *Statistical Analysis*: The user can select a cluster visible on the display, and get statistical data about it, such as the average distance (in data space) between its data-items, and the variance in distances (in data space) between its data-items. Low values for average distance and variance may indicate to the user that the cluster visible on the display is also a real cluster of the dataset. In our system, the user can select a cluster by dragging the mouse, and closing a curve around around it (see Figure 2(g)). The user can then obtain statistical data about the cluster by clicking the "Stat" button. The data will be displayed in a text area provided for it (see Figure 2(g)).

An important aspect of our approach, which makes it successful in detecting clusters, is that even though individual operations (such as cluster probing) may not show the existence of clusters by themselves, it is their combination, with reinforcement provided by animation, that reveals the clusters of the dataset.

3 Experimental Results

We have developed a prototype system that implements our framework. This system is developed in Java, which makes it platform-independent. Figures 2,4,3, and 5 show the user-interface of the system. The user can load a file containing a dataset by clicking on the "load" button. This will create an initial visualization of the dataset, and will also create sliders for adjusting parameters, and for zooming. Since our framework is interactive, where the users explore the dataset by repeatedly changing the visualization, the choice of initial values of the parameters, for creating the initial visualization, is not very crucial. Hence, the system simply sets all the parameters initially to 0.5.

The users can create a connection to an Oracle database-server by clicking on the "connect Database" button. By clicking the "Create" button, they can also place a copy of the dataset in the Oracle database-server, so that they can perform subset selection on it by giving SQL-Queries. They can remove this copy by clicking the "Drop" button.

The system also allows the users to select a cluster visible on the screen by dragging the mouse around it and creating a closed curve around it (see Figure 2(c)). Having selected a cluster, the user can perform several kinds of operations: they can get statistical data about it by clicking on the "Stat" button, or they can ask the system to show only the data-items of of the cluster, hiding the remaining data-items, by clicking on the "Show" button (see Figure 2(d)), or they can ask the system to hide the data-items of the cluster, showing only the remaining data-items, by clicking on the "Hide" button.

The users can output the data-items of a cluster visible on the screen, into a file, by clicking on the "Save" button. The users can clear the display, create a visualization again after clearing, exit from the system, undo the affect of the previous command, and get help, by clicking on the "Clear", "View", "Exit", "Restore", and "Help" buttons, respectively. The "Map" button is currently not functional. In a future version of the system, the users can use it to define new mappings for creating visualizations.

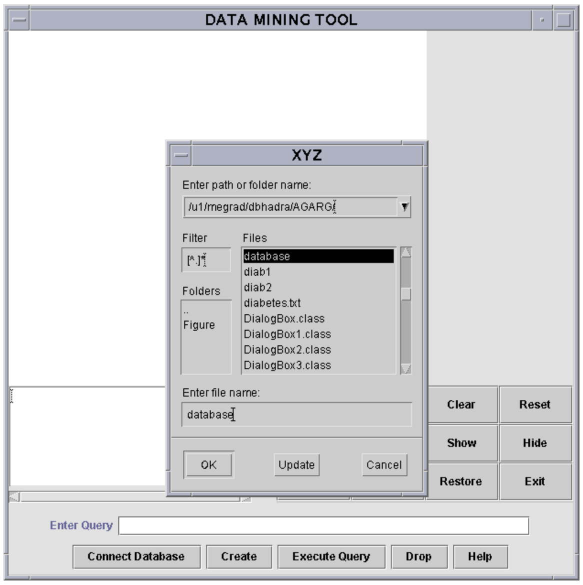
We have tested our system for detecting clusters successfully on several datasets with up to 100,000 data-items in them. In this section, we report the results of our experiments on three datasets. Two of them are synthetic datasets, and one is a dataset from a space-shuttle control application. In Section 4, we present our results on another dataset from a medical application. Results of our experiments, with screen dumps, on some other datasets are available at the URL: `http://www.cse.buffalo.edu/infoviz/datamining/results`.

3.1 Dataset 1

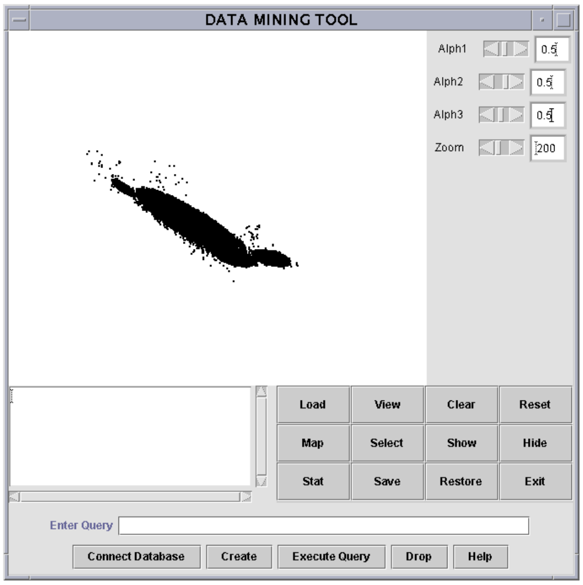
This is a 3-dimensional dataset, called *DS1*, with 100,000 data-items, and 5 clusters. This dataset was used for testing CURE [10] in the paper that appeared in SIGMOD 98. *DS1* is available at the URL: `http://www.bell-labs.com/project/serendip/Dataset1.gz`.

Figure 2 shows the process through which we detected clusters in it using our system. We loaded the dataset from a file, and created an initial visualization, and the sliders for zooming and parameter re-adjustment, by clicking on the "Load" button (Figure 2(a)). The initial visualization is shown in Figure 2(b). We could see 3 clusters in the initial visualization. One of the clusters is very large. We then explored this large cluster in detail by showing only its data-items on the screen, hiding the other data-items, and then zooming into it. This was done by first dragging the mouse around it, completing a closed curve around it (Figure 2(c)), and then clicking on the "Show" button (Figure 2(d)). We zoomed into it by increasing the zoom-factor through the slider. This revealed two denser clusters hidden insider it (Figure 2(e)). We then displayed all the data-items again on the screen, and then interactively re-adjusted the parameters, and changed the zoom through the sliders, which eventually showed all the 5 clusters of the dataset (Figure 2(f)). We also statistically analyzed each cluster visible on the screen, by selecting it (by dragging the mouse around it, completing a closed curve around it) and clicking on the "Stat" button. Low values for average distance, and variation in distance indicated to us that it is, indeed, a real cluster of the dataset (Figure 2(g)). We also randomly sampled each cluster visible on the screen by click on its pixels, and reading the values of the attributes of the data-items mapped to these pixels. (Our system displays these values in a pop-up window (Figure 2(h))). Similarity in the values of these attributes for each cluster further indicated to us that these were the real clusters of the dataset.

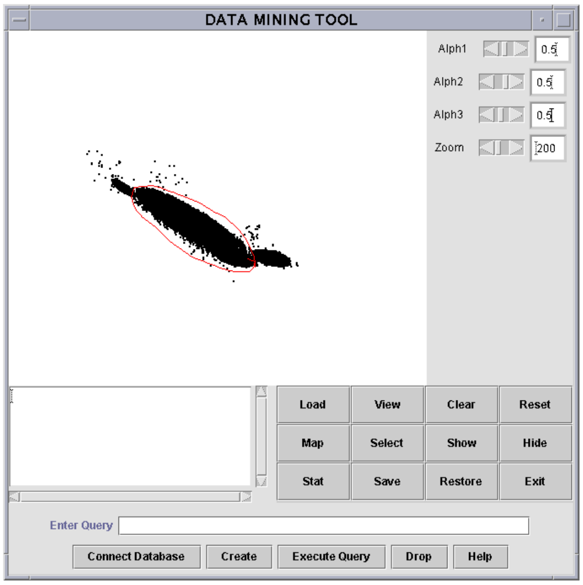
Our experiments also showed that, as we re-adjusted the parameters, the data-items of the same (real) cluster of the dataset typically moved together on the screen, so that the whole cluster gives the appearance of a semi-rigid body in motion. This visual feedback gave us a very strong indication that these data-items, indeed, belonged to the same cluster. We will discuss the help provided by animation in detecting clusters, in a greater detail, in Section 4.


DATA MINING TOOL
XYZ
Enter path or folder name:
/u1/megrad/dbhadra/AGARG/
Filter
Files
database
diab1
diab2
diabetes.txt
DialogBox.class
DialogBox1.class
DialogBox2.class
DialogBox3.class
Folders
Figure
Enter file name:
database
OK
Update
Cancel
Clear
Reset
Show
Hide
Restore
Exit
Enter Query
Connect Database
Create
Execute Query
Drop
Help

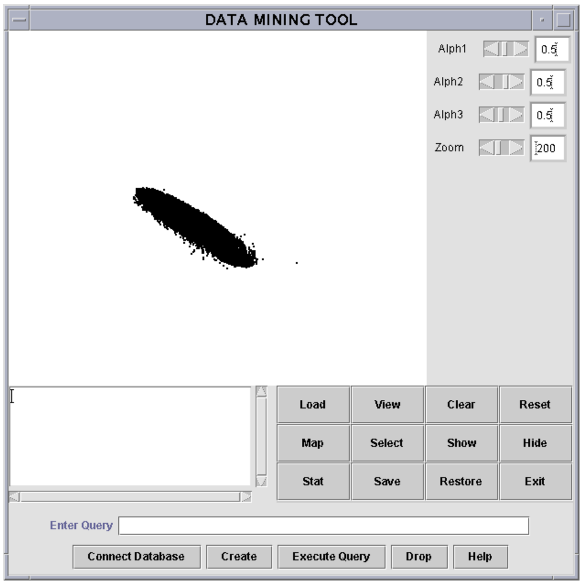
(a)


DATA MINING TOOL
Alph1 0.5
Alph2 0.5
Alph3 0.5
Zoom 200
Load
View
Clear
Reset
Map
Select
Show
Hide
Stat
Save
Restore
Exit
Enter Query
Connect Database
Create
Execute Query
Drop
Help

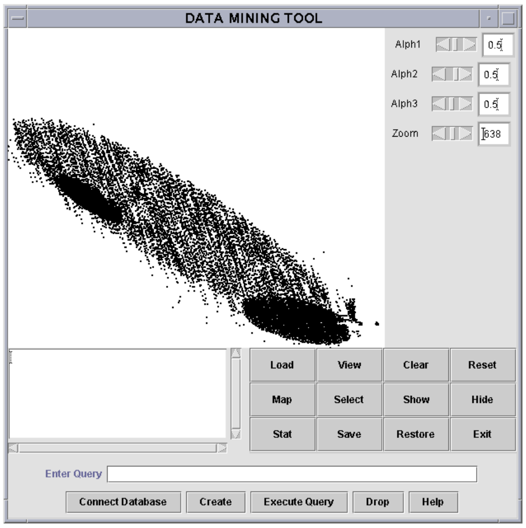
(b)

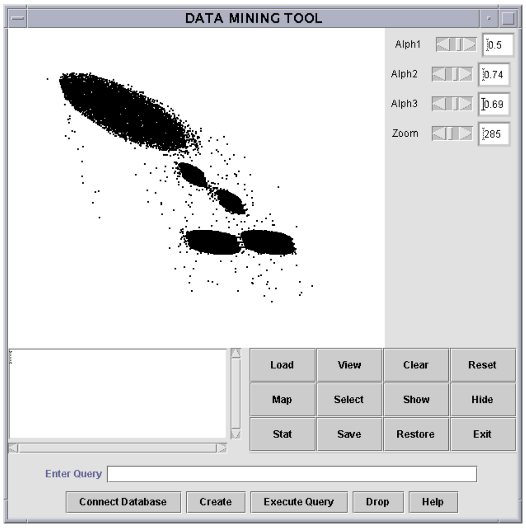

DATA MINING TOOL
Alph1 0.5
Alph2 0.5
Alph3 0.5
Zoom 200
Load
View
Clear
Reset
Map
Select
Show
Hide
Stat
Save
Restore
Exit
Enter Query
Connect Database
Create
Execute Query
Drop
Help

(c)

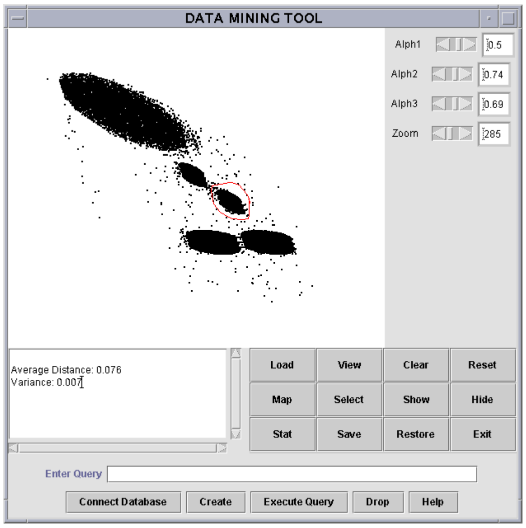

DATA MINING TOOL
Alph1 0.5
Alph2 0.5
Alph3 0.5
Zoom 200
Load
View
Clear
Reset
Map
Select
Show
Hide
Stat
Save
Restore
Exit
Enter Query
Connect Database
Create
Execute Query
Drop
Help

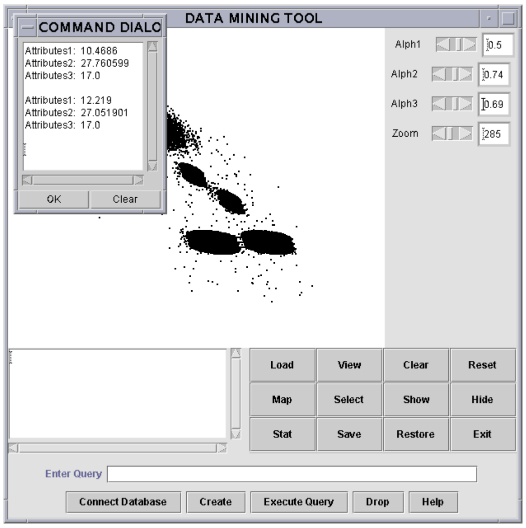
(d)





(e) (f)





(g) (h)

Figure 2: Interactive visual detection of clusters in the dataset DS1 [10]: (a) Loading of the dataset from a file called *database*; (b) Creation of the initial visualization; (c) Selecting a cluster-like formation on the screen by dragging the mouse around it, and completing a closed curve; (d) Hiding the remaining data-items by pressing the "Show" button; (e) Zooming in to the selected cluster-like formation, to reveal two denser cluster hidden inside it; (f) Re-adjusting parameter and changing zoom, to reveal all the 5 clusters of the dataset; (g) Selecting a cluster and clicking on the "Stat" button to display statistical information about it in the text area on the bottom-left part of the user-interface; and (h) Clicking on two randomly selected pixels, to show, in a pop-up window, the values of the attributes of the data-items mapped to these pixels.

3.2 Dataset 2

This dataset, which we call *Shuttle*, is 10-dimensional, and consists of 43,500 data-items and 7 clusters. It contains data related to space-shuttle control and is available at the URL: `http://www.ncc.up.pt/liacc/ML/statlog/datasets/shuttle/shuttle.doc.html`. Of its seven clusters, one is very large with approximately 80% data-items, two are moderately large with approximately 16% and 5.6% data-items, respectively, one is fairly small with approximately 0.3%data-items, and the remaining three are extremely small with just 0.09%, 0.01% and 0.03% data-items, respectively.

The initial visualization for *Shuttle* showed 4 clusters (Figure 3(a)). Parameter-readjustment showed 5 sufficiently large clusters, with one particularly very large cluster(Figure 3(b)), and several very small clusters. We then explored this larger cluster in a greater detail, by selecting it, and hiding the rest of the data-items(Figure 3(c)). However, even on zooming, and re-adjusting parameters several times, the cluster remained “monolithic”, and we could not find any other clusters hidden inside it. More over, in the animation created by the continuous re-adjustment of parameters, the whole cluster tended to move like a single semi-rigid body. Selecting this cluster and statistically analyzing it showed low values for average distance and variance (Figure 3(d)), which gave further evidence that this cluster is a real cluster. So we concluded that this was also a real cluster of the dataset.

Counting the data-items of the 5 clusters that we detected, we found that they had approximately 80%, 15%, 2.8%, 2.5%, and 0.3%, data-items, respectively, in them. Thus, we were able to detect 4 of the 7 clusters of the dataset (the two clusters with 2.8%, and 2.5% together roughly form the single cluster with 5.6 % items). The other three clusters, which we could not detect, are very small with just 0.09%, 0.01% and 0.03% data-items, respectively, in them.

3.3 Dataset 3

This dataset is 5-dimensional, has 10,000 data-items, and has 6 clusters. It is generated using the program *DataGen*, which is available at the URL: `http://www.datasetgenerator.com/`. It is a

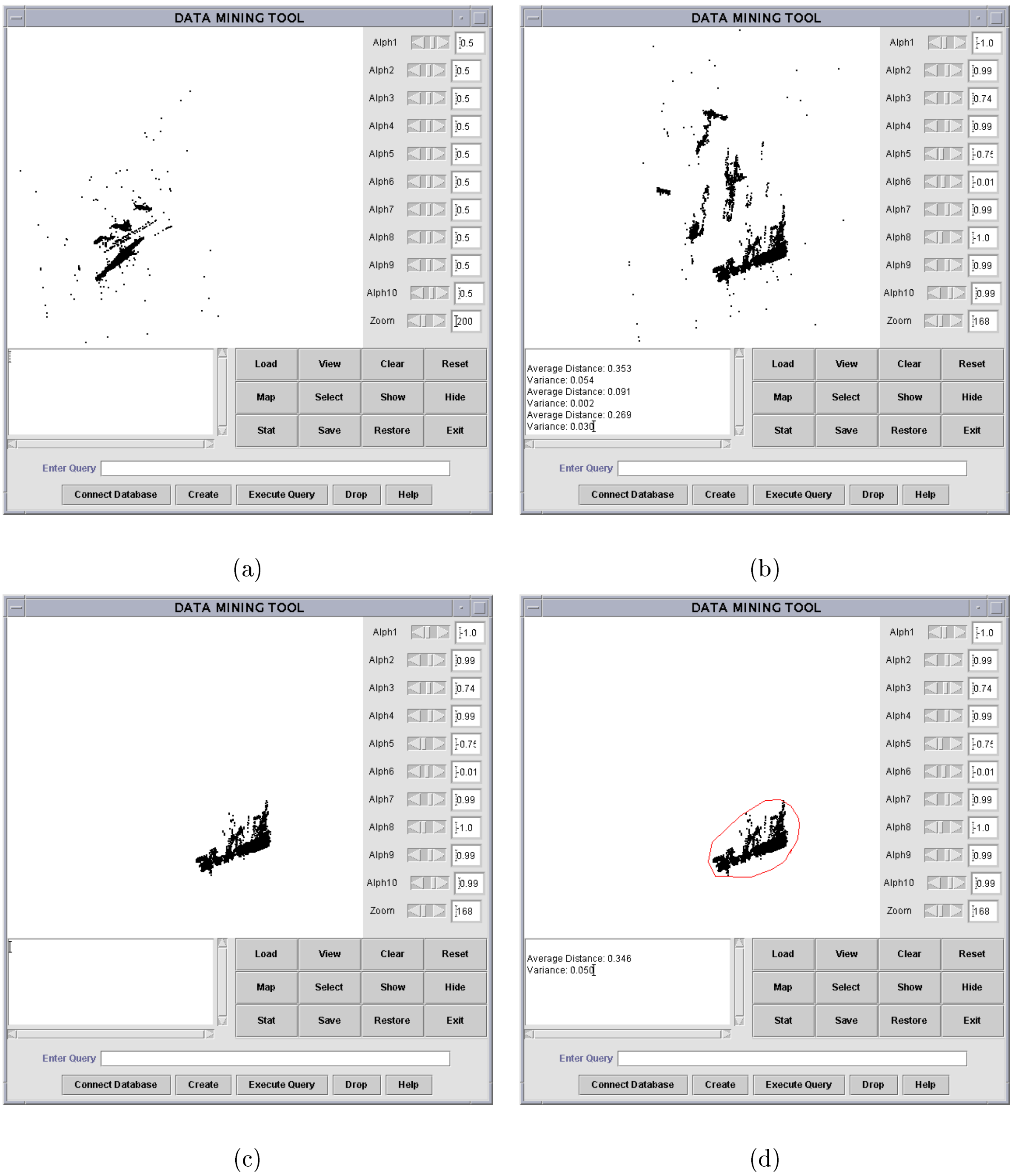


Figure 3: Interactive visual detection of clusters in the dataset *Shuttle*: (a) Initial Visualization; (b) Parameter re-adjustment reveals 5 sufficiently large clusters, with one very large cluster, and several very small clusters; (c) selecting the large cluster and hiding the remaining data items (d) Selecting the large cluster and clicking on the "Stat" button to display statistical information about it.

program originally developed by Gabor Melli, and has been used by the researchers in data mining and machine learning to test their algorithms.

DataGen allows the user to generate a dataset that conforms to her constraints. For example, the user can place constraints on datatypes, domain, and range of the attributes (called columns by *DataGen*) of a dataset. Constraints between various attributes can also be defined. The user can also define the distribution of data-items in a dataset.

Our dataset was generated using the following constraints (rules):

(14.4%) c1 <- A<=1 & B<=3 & C<=3 & D>=10 & E>=146

(13.5%) c1 <- A<=3 & C>=4 & D>=9 & E>=147

(13.3%) c1 <- A<=1 & B>=4 & C>=2 & D<=1 & E<=2

(12.2%) c2 <- A<=3 & D<=3 & E>=147

(10.6%) c2 <- C>=2 & D>=11 & E<=3

(8.8%) c2 <- A<=2 & B>=4 & C<=1 & D>=9 & E>=145

(7.2%) c3 <- A>=10 & B>=4 & C<=1 & E>=145

(5.8%) c3 <- A>=10 & B<=3 & C<=3 & D<=2 & E<=2

(4.9%) c4 <- A>=9 & B>=4 & C<=2 & D<=2 & E<=2

(3.6%) c4 <- A>=9 & B<=1 & C<=3 & E>=147

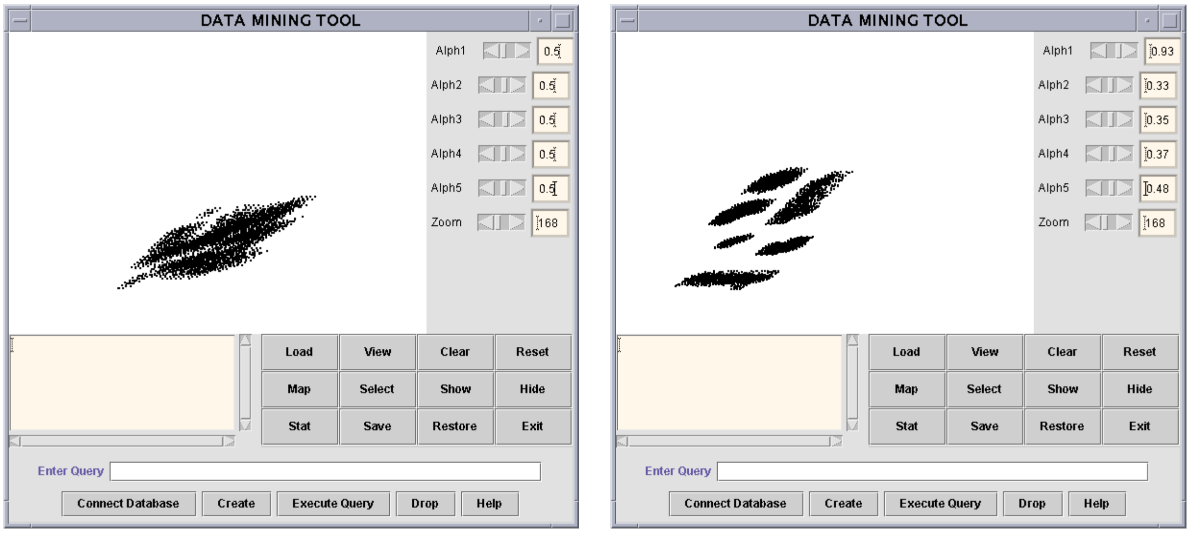
(2.5%) c5 <- A>=11 & B>=6 & C>=4 & D<=1 & E>=147

(1.4%) c5 <- A>=10 & B<=3 & C>=4 & D>=11 & E>=146

(1.1%) c6 <- A>=11 & B>=6 & C<=1 & D>=10 & E<=1

(0.8%) c6 <- A<=1 & B<=3 & C>=3 & D<=1 & E<=1

Here, A, B, C, D, and E are the 5 dimensions of the dataset, and each ci, where $1 <= i <= 6$, is a cluster. The first three rules say that c1 consists of data-items that satisfy the constraints: (A<=1 AND



(a) (b)

Figure 4: Interactive visual detection of clusters in the dataset generated by DataGen: (a) Initial Visualization; (b) Re-adjusting parameters reveals all the six clusters of the dataset.

B<=3 AND C<=3 AND D>=10 & E>=146) OR (A<=3 AND C>=4 AND D>=9 AND E>=147) OR (A<=1 AND B>=4 AND C>=2 AND D<=1 AND E<=2). The percentage value before each rule gives the percentage of data-items of the dataset that satisfy that rule. For example, 14.4% of data-items will satisfy the first rule.

The initial visualization for the dataset is shown in Figure 4(a). The initial visualization only shows two clusters. However, by performing a series of parameter re-adjustment operations, we could find all the six clusters of the dataset (See Figure 4(b)). Cluster probing and animation also helped in determining that these were the real clusters of the dataset.

4 Role of Animation in Cluster Detection

A series of continuous parameter re-adjustments, done by moving the sliders, creates an animation in which data-items move to their new positions on the screen from their older positions. During our experiments, we discovered that this animation provides important clues in detecting clusters. We discovered that the data-items of a real cluster of the dataset tend to move together in a similar fashion on the screen during this animation, which helps in identifying the cluster.

We have illustrated this phenomenon in Figure 5, which shows four frames of an animation of the cluster-detection process in a dataset of diabetic patients. It is a 9-dimensional dataset, where each dimension (attribute) corresponds to a characteristic shown by the patient, such as, number of times pregnant, plasma glucose concentration, diastolic blood pressure, triceps skin fold thickness, 2-hour serum insulin, body mass index, diabetic pedigree, and age. It contains 768 data-items (patients), and has two clusters. This dataset is available at the URL: `http://touca.ncc.up.pt//statlog/datasets/diabetes/diabetes.doc.html`.

Figure 5(a) shows the first frame. As we continuously change the value of the parameter α_9, a cluster of data-points moves from the left side of the screen to the right side (see Figure 5(a–d)), and in the end, separates completely from the other cluster. This gives a very strong indication that it is a real cluster. Further verification was then obtained by doing cluster-probing on it.

Mathematically, we can explain this phenomenon as follows: Recall that we use a linear mapping, $\vec{cq} = \sum_{1 \leq i \leq N} \alpha_i (4/N)(x_i(p) - 0.5)\vec{cs}_i$, to map a data-item $p(x_1, x_2, \ldots, x_N)$ to a point q on the display surface. Suppose we change each parameter α_i by a small amount $d\alpha_i$. Using a first-order approximation, $\vec{cq}$, will change by an amount $d\vec{cq}$, which is approximately equal to $\sum_{1 \leq i \leq N} (4/N)(x_i(p) - 0.5)\vec{cs}_i d\alpha_i$.

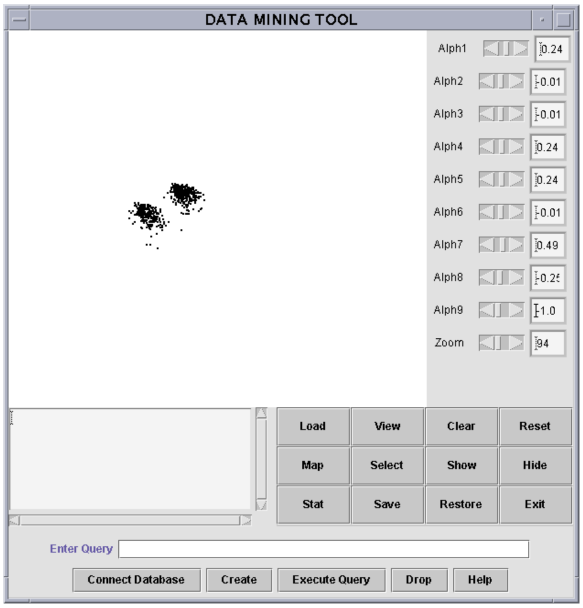
Let p and r be two nearby points of the same cluster, that are also currently mapped to two nearby points a and b, respectively, on the screen. On making a change $d\alpha_i$ in each α_i, points a and b will change their positions by amounts $\sum_{1 \leq i \leq N} (4/N)(x_i(p) - 0.5)\vec{cs}_i d\alpha_i$, and $\sum_{1 \leq i \leq N} (4/N)(x_i(r) - 0.5)\vec{cs}_i d\alpha_i$, respectively. Thus, relative change in positions of a and b will be equal to $\sum_{1 \leq i \leq N} (4/N)(x_i(p) - (x_i(r))\vec{cs}_i d\alpha_i$. Since p are r are nearby points in the data space, $x_i(p) - x_i(r)$ will, in general, also be small, and hence, the relative change in the positions of a and b will also be small. Thus, the two of them will appear to be moving together on the screen during parameter re-adjustment. Generalizing this to the whole cluster, if a real cluster is mapped to a cluster-like formation on the screen, then on parameter re-adjustment, the whole cluster-like formation will move as if some semi-rigid body were moving. The ensuing animation will give us a strong indication that the cluster-like formation visible on the screen is indeed a real cluster of the dataset.

We have observed this phenomenon in our other experimental studies also, which has helped us in detecting clusters.

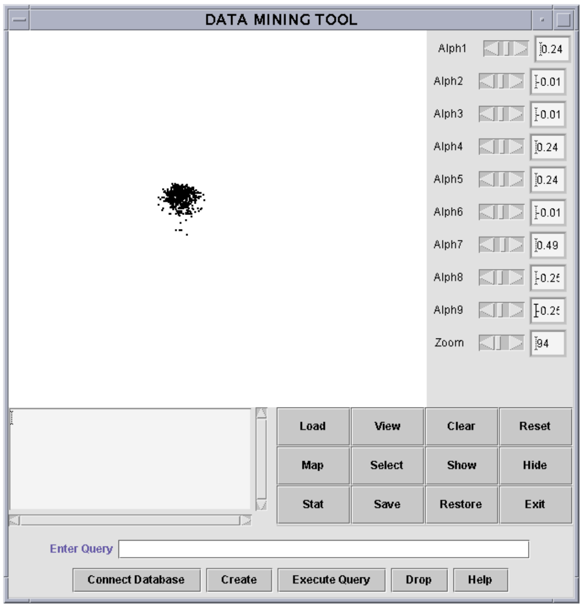
5 Related Work

Cluster-detection in datasets is a well-studied problem. Several automatic detection techniques have been developed. In particular, [13] presents several agglomerative hierarchical algorithms for cluster-detection. CURE [10] is a system that employs random sampling and partitioning to detect clusters. ROCK [11] presents a technique for clustering boolean and categorical data. DBSCAN [6] uses density measure to detect clusters. BIRCH [29] presents a hierarchical clustering based approach that minimizes computational and space requirements. CHAMELEON [14] employs graph-partitioning to detect clusters in a hierarchical fashion. WaveCluster [25] uses wavelets to detect clusters in a spatial dataset. DBCLASD [28] uses a distribution-based approach for cluster-detection in spatial datasets. CLARANS [20] uses randomized search for cluster detection. PAM [15] uses K-Mediods to detect clusters.

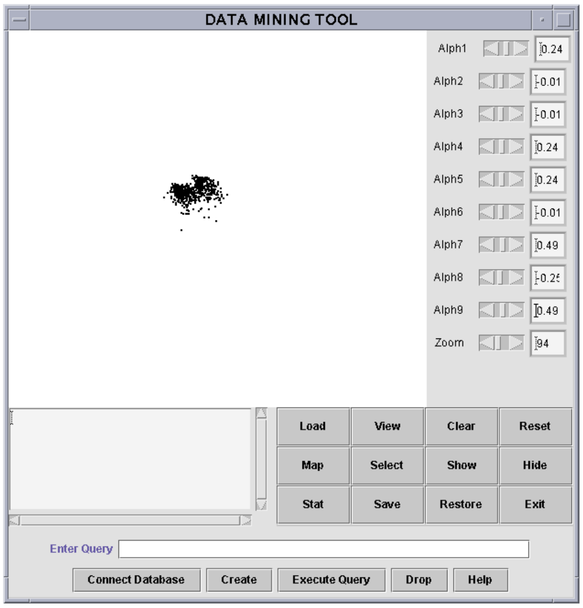
We view our visualization framework to be complementary to these automatic cluster-detection techniques. For example, the user can first run our system to get the clusters, and then pass them to any of these techniques for further analysis. Or, they may first run one of these techniques to obtain clusters, and then visualize and explore them using our framework. A tighter coupling between our framework and some of these techniques is also possible. Namely, during cluster-probing, we can either employ these techniques directly to analyze a selected cluster, or use the mathematical characterization (such as density-based) of a cluster used by them, to statistically analyze the cluster.



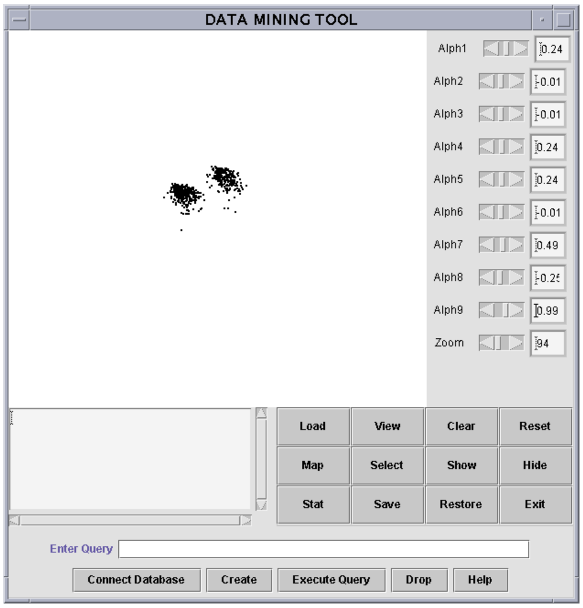
(a)



(b)



(c)



(d)

Figure 5: Four frames in the animation created during the interactive visual detection of clusters in the dataset on Diabetic patients: (a) First Frame; (b) Second Frame; (c) Third Frame; (d) Fourth Frame.

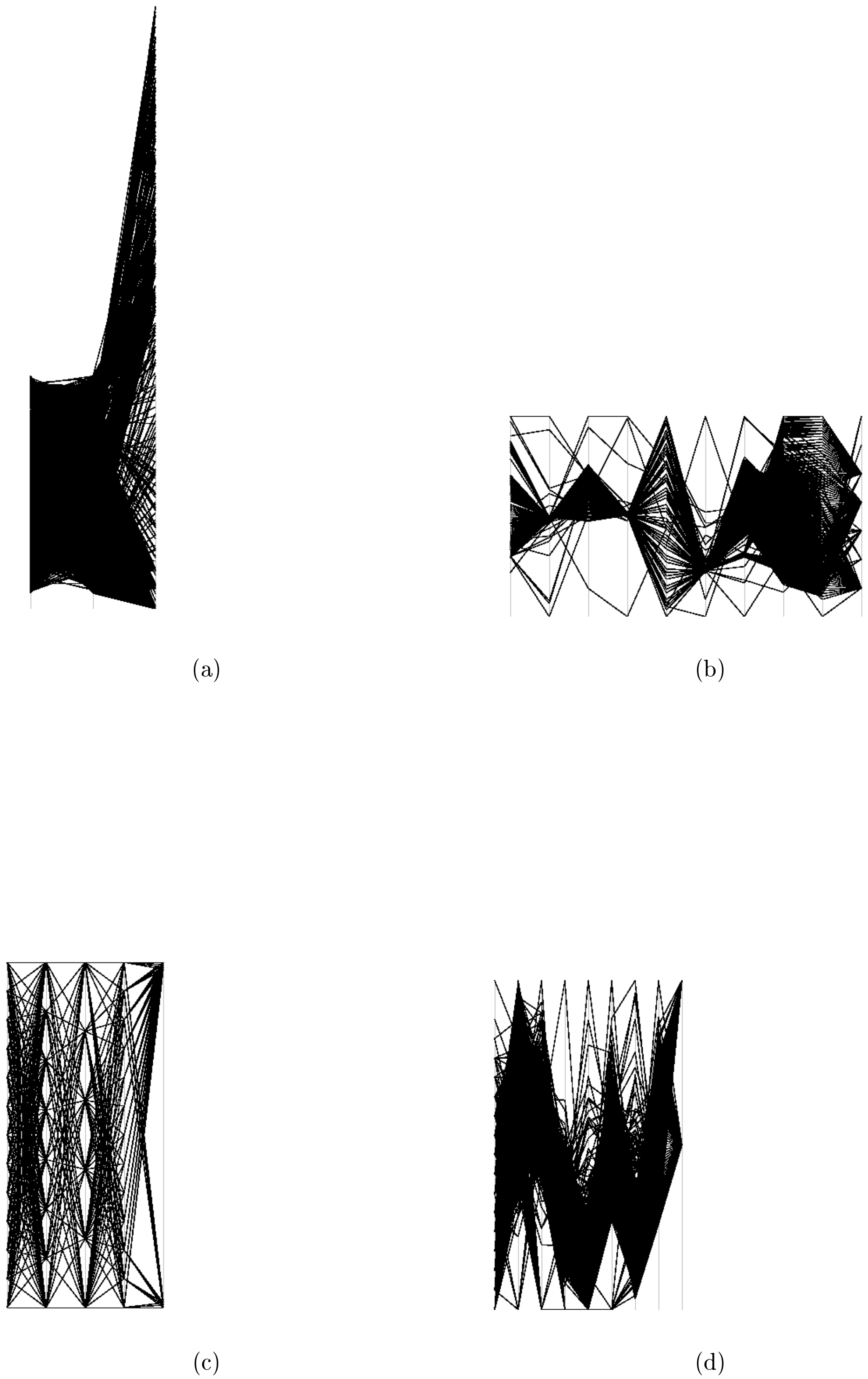
Various kinds of techniques for visualizing multidimensional databases have also been developed. The icon-oriented techniques [4, 26] use icons, whose features encode the values of the attributes of the data objects. Projection-oriented techniques such as Scatterplot matrices, coplots, and prosection [5, 9] visualize multidimensional datasets by creating their projections on various planes. Since these techniques use only projections, they do not generally preserve clustering information in their visualizations. Hierarchical techniques [7, 18] impose a hierarchy on the axes of the N-dimensional space, and are not geared towards cluster-detection. Parallel Coordinates [12] map the axes of the N-dimensional space on to N parallel lines in the plane. It is generally difficult to detect clusters in parallel coordinates, unless the data set is well behaved. VisDB [16] uses a combination of spatial positioning and coloring to visualize data. It can be used to visualize clusters of a dataset, but it imposes a linear-ordering on the data for visualization purposes, and has limited interactive support, which limits its effectiveness in detecting clusters. In Figure 6, and Figure 7, we show a visualization of the datasets of Section 3.1, Section 3.2, Section 3.3, and Section 4, respectively, by Parallel Coordinates, and VisDB, respectively. As can be seen, it is difficult to detect clusters in these datasets through these visualizations, where as by using our interactive visualization framework, we can detect clusters in them, as shown earlier.

6 Conclusions and Future work

We have presented an interactive visualization framework for detecting clusters of a dataset. We have also developed a prototype system based on this framework, and conducted experiments, which demonstrate its effectiveness in detecting clusters. Our experiments also suggest that animation can provide important clues in detecting clusters.

As a part of the future work, we would like to do the following:

- incorporate other visualization techniques, such as separation and layering, coloring, and focus+context displays [8, 23], into our framework to increase its effectiveness,



(a) (b) (c) (d)

Figure 6: Visualization created by Parallel Coordinates [12] of the datasets of: (a) Section 3.1, (b) Section 3.2, (c) Section 3.3, and (d) Section 4, respectively.

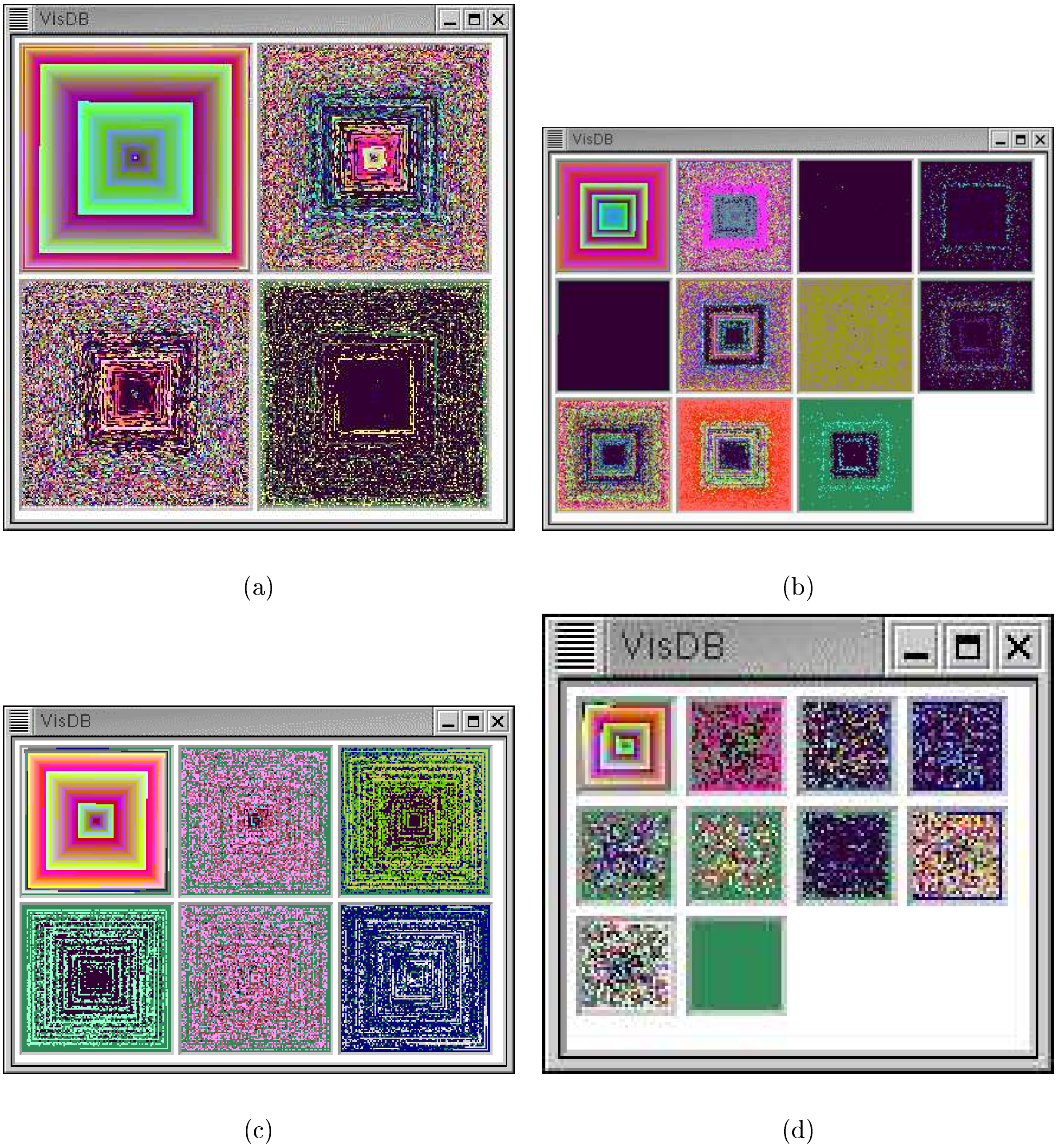


Figure 7: Visualization created by VisDB [16] of the datasets of: (a) Section 3.1, (b) Section 3.2, (c) Section 3.3, and (d) Section 4, respectively.

- explore the integration of automatic cluster detection techniques with our framework, especially in context of the cluster-probing operation,
- allow users to define their own mapping functions for mapping the data-items from the data space to the visual space,
- add more measures, such as density measure, for statistically analyzing clusters, and
- implement our prototype system as a Java-applet.

References

[1] B. B. Bederson, J. D. Hollan, K. Perlin, J. Meyer, D. Bacon, and G. Furnas. Pad++: A zoomable graphical sketchpad for exploring alternate interface physics. *Journal of Visual Languages and Computing*, 7(1):3–31, 1996.

[2] S. K. Card, J. D. Mackinlay, and B. Shneiderman. *Readings in Information Visualization: Using Vision to Think.* Morgan Kaufmann, San Francisco, CA, 1999.

[3] M. Chen, J. Han, and P. Yu. Data mining: An overview from a database perspective. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):866–883, 1996.

[4] H. Chernoff. The user of faces to represent points in k-dimensional space graphically. *Journal American Statistical Association*, pages 361–368, 1973.

[5] W. S. Cleveland. *Visualizing Data.* AT&T Bell Laboratories, Hobart Press, NJ, 1993.

[6] M. Ester, K. H.-P., J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, 1996.*

[7] S. K. Feiner and C. Beshers. Worlds within worlds: Metaphors for exploring n-dimensional virtual worlds. In *Proc. ACM Symp. on User Interface Software and Tech.*, pages 76–83, 1990.

[8] G. Furnas. The fisheye view: A new look at structured files. Technical report, Bellcore, 1981.

[9] G. W. Furnas and A. Buja. Prosection views: Dimensional inference through sections and projections. *Journal of Computational and Graphical Statistics*, pages 323–353, 1994.

[10] S. Guha, R. Rastogi, and K. Shim. CURE: An efficient clustering algorithm for large databases. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data.*

[11] S. Guha, R. Rastogi, and K. Shim. ROCK: A robust clustering algorithm for categorical attributes. In *IEEE Conference on Data Engineering*, 1999.

[12] A. Inselberg and B. Dimsdale. Parallel coordinates: A tool for visualizing multi-dimensional geometry. In *Proceedings of IEEE Visualization'90*, pages 361–375.

[13] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data.* Prentice Hall, 1988.

[14] G. Karypis, E.-H. Han, and V. Kumar. Chameleon: A hierarchical clustering algorithm using dynamic modeling. *IEEE Computer*, page to appear.

[15] L. Kaufman and P. Rouseeuw. *Finding Groups in Data: an Introduction to Cluster Analysis.* John Wiley & Sons, 1990.

[16] D. Keim and H.-P. Kriegel. VisDB: Database exploration using multidimensional visualization. *IEEE Computer Graphics and Applications*, pages 40–49, September, 1994.

[17] J. Larkin and H. A. Simon. Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11(1):65–99, 1987.

[18] A. R. Martin, M. O. Ward, and N. Wittels. Exploring n-dimensional databases. In *Proceedings of IEEE Visualization'90*, pages 230–239.

[19] S. Mukherjee and J. Foley. Visualizing the World-Wide Web with navigational view builder. *Computer Networks and ISDN System*, 1995.

[20] R. Ng and J. Han. Efficient and effective method for spatial data mining. In *Very Large Data bases (VLDB 94)*, pages 144–155, 1994.

[21] R. Ng, L. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained associations rules. In *Proc. SIGMOD*, pages 13–24, 1998.

[22] G. Piatetsky-Shapiro, U. Fayyad, and P. Smith. From data mining to knowledge discovery: an overview. *Advances in Knowledge Discovery and Data Mining*, pages 1–35, 1996.

[23] M. Sarkar and M. H. Brown. Graphical fisheye views. *Commun. ACM*, 37(12):73–84, 1994.

[24] B. Schneiderman. The eyes have it: A task by data type taxonomy for information visualization. In *IEEE Workshop on Visual Languages*, pages 336–343, 1996.

[25] G. Sheikholeslami, S. Chatterjee, and A. Zhang. Wavecluster: A multi-resolution clustering approach for very large spatial databases. In *Proc. VLDB '98*, 1998.

[26] T. M. Sparr, R. D. Bergeron, and L. D. Meeker. A visualization-based model for a scientific database system. *Focus on Scientific Visualization*, pages 103–121, 1993.

[27] C. Williamson and B. Schneiderman. The dynamic homefinder: Evaluating dynamic queries in a real-estate information exploration system. In *Proc. SIGIR Conf., 1983*, pages 339–346.

[28] X. Xu, M. Ester, H.-P. Kriegel, and J. Sander. A distribution-based clustering algorithm for mining in large spatial databases. In *14th International Conference on Data Engineering*, 1998.

[29] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An efficient data clustering method for very large databases. In *Proc. of the 1996 SIGMOD*, pages 103–114.