

Visualization and Data Mining in
an 3D Immersive Environment:
Summer Project 2003

Christian T. Brown,
Harry W. Bullen,
Sean P. Kelly,
Robert K. Xiao,
Steven G. Satterfield,
John G. Hagedorn,
Judith E. Devaney

Abstract

We describe the application of some simple data-mining tools and visualization of the results in the NIST RAVE, an immersive 3D environment. The project builds upon several previous software development efforts, most notably the Glyph ToolBox, a set of tools used for creating three-dimensional glyphs. The visualizations consist of 3D and 2D representations of the data and of the data mining output combined in various layouts designed to aid ease of interpretation. These were in some cases equipped to allow user interactivity in the RAVE. We determined which methods were most and least effective in analyzing the data, providing examples for each, based on our experiences. We also describe the development of new capabilities for the Glyph ToolBox and several other additional visualization tools.

Contents

Introduction	1
1 Introduction	1
1.1 Overview	1
1.2 Background	2
2 Resources and Tools	3
2.1 Glyph ToolBox	3
2.2 NIST RAVE	3
2.3 DIVERSE	4
2.4 Inventor Utilities: ivUtilities	4
2.5 Weka	5
2.6 IDL	5
2.7 UCI Machine Learning Repository	5
3 Tools Created for this Project	6
3.1 ivMuseum	6
3.2 Glyph ToolBox Additions and Modifications	6
3.3 Other Alternatives to the Museum Environment	7
3.4 Visual Component Tools	8
3.5 Dataflow	13
4 Data Sets Analysed	17
4.1 Tic-Tac-Toe	17
4.2 Miles Per Gallon	19
4.3 Housing	21
4.4 Image Segmentation	23
4.5 Forest Cover Type	26

4.6	BUPA Liver Disorder	29
4.7	Road Detection	31
4.8	Wine	33
4.9	Page Block	34
4.10	Spambase	35
4.11	CPU	39
4.12	Solar Flare	43
4.13	Abalone	46
4.14	Glass	51
4.15	Mushroom	56
5	Conclusions	60
5.1	Problems	60
5.2	Effective Techniques	61
5.3	Conclusions	61

Chapter 1

Introduction

The bulk of this work was done by four students from the Montgomery Blair High School in Montgomery County Maryland: Christian T. Brown, Harry W. Bullen, Sean P. Kelly, and Robert K. Xiao. They came to work at NIST in the Scientific Applications and Visualization Group (SAVG) of the Information Technology Laboratory (ITL) during the summer of 2003. In the SAVG we work with many different types of data from researchers throughout NIST. These data sets have become more complex, sometimes with dozens of variables. In order to understand these data sets and to interpret interrelations among variables, we look toward the techniques of data mining and visualization.

1.1 Overview

In this project we bring together data mining tools and various information visualization techniques in the NIST immersive visualization environment. The term "data mining" refers to the search for patterns or correlations that provide some understanding or predictive power. For example, can we write software that will examine hospital patient records and give us guidelines for predicting, on the basis of basic medical history, whether someone is likely to get a certain disease in the future? We use as a testbed a group of existing scientific data sets from the UCI Machine Learning Repository [6]. We intend this project to be an avenue for trying out different ideas and approaches for visualizing the results of data mining operations in an immersive environment.

1.2 Background

SAVG has been visualizing data in collaboration with NIST scientists for years [14, 13]. We have used a variety of two-dimensional (2D) and three-dimensional (3D) techniques, and most recently we have been working within an immersive 3D environment, the NIST RAVE which is described below.

In addition, we have worked with a variety of machine learning techniques and tools in order to derive meaning from large experimental data sets. This work has included the development of our own genetic programming system [10], as well as the use of existing tools such as Weka [16, 4], autotool [8], and C5¹ [12, 1].

This project brings together all of these elements: machine learning tools, 2D and 3D data representations, and immersive visualization.

¹Certain commercial products may be identified in this paper in order to adequately describe the subject matter of this work. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the identified products are necessarily the best available for the purpose.

Chapter 2

Resources and Tools

2.1 Glyph ToolBox

The Glyph Toolbox [5] is a collection of tools, i.e. individual UNIX style command line programs, that can be used to build a polygon-based virtual environment and glyphs for data representation. Each tool outputs an ASCII based file that is machine and rendering independent. The actual display of the ASCII files is handled by a viewing program, such as DIVERSE/diversify, VRML, Open Inventor, etc. In some cases, an additional conversion to another file format may be required.

The key to understanding the Glyph Toolbox is to see that it enables the creation of three-dimensional glyphs both complex and intuitive enough to convey information while maintaining a simple and intuitive system for actually generating the glyphs based on existing data. Glyphs created by the Glyph Toolbox were designed to be used in the NIST RAVE both for constructing virtual environments and for plotting or otherwise modeling data obtained in a variety of areas of research.

2.2 NIST RAVE

SAVG has developed an immersive visualization environment that can be used to gain increased insight into large, complex data sets. Such data sets are becoming more commonplace at NIST, as high performance parallel computing is used to develop higher fidelity simulations, and combinatorial experimental techniques are used in the laboratory. Immersive visualization environments allow the sci-

entist to interactively explore complex data by literally putting oneself inside the data.

The ITL Immersive Visualization Laboratory is a RAVE (Reconfigurable Automatic Virtual Environment) from Fakespace Systems [15]. The two-wall RAVE is configured as an immersive corner with two 8 ft. x 8 ft. (2.44m x 2.44m) screens flush to the floor and oriented 90 degrees to form a corner. The large corner configuration provides a very wide field of peripheral vision, with stereoscopic display and head tracking for realistic perspective based on the direction the user is viewing. The RAVE is driven by an SGI Onyx 3400 parallel processor graphics super computer.

2.3 DIVERSE

The primary software controlling the RAVE is an open source API and set of utility commands called DIVERSE [9] (Device Independent Virtual Environments-Reconfigurable, Scalable, Extensible). DIVERSE handles the details necessary to implement the immersive environment. It provides a display utility *diversify* for interactive viewing within the immersive environment. In addition to immersive display systems, DIVERSE also provides support for desktop display and interaction. User applications and data can thus be utilised on a full range of systems from laptop to desktop to single wall to multi-wall immersive systems.

2.4 Inventor Utilities: ivUtilities

Open Inventor [3] is a toolkit provided by SGI for (among other things) creating and manipulating 3D polygonal objects. Inventor object files can be read by DIVERSE software and can be combined in 3D virtual environments with objects created by the Glyph ToolBox.

ivUtilities is a collection of commands created by SAVG for quickly making short snippets of Inventor ASCII files. The commands can be combined through UNIX pipes to form Inventor based visualizations. The idea is to follow the traditional UNIX philosophy of small tools that combine to form a result greater than the sum of the parts. Think of these commands as Shell Script Visualization.

ivUtilities might be useful for rapid prototyping, quick experiments, creating simple demos or any purpose where getting things displayed easily is important. The commands are intended to be easy to learn and leverage off of already known

command line practices (pipes and filters). Each command is small with a minimal number of command line options.

In this project we use the ivUtilities to create elements within the 3D immersive displays.

2.5 Weka

Here is a description of the Weka machine learning software from its web page [4]:

Weka is a collection of machine learning algorithms for solving real-world data mining problems. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes. Weka is open source software issued under the GNU General Public License.

We use Weka in this project to perform some simple data-mining tasks. These are described in subsequent sections.

Note that we convert all of the input data sets to Weka's ARFF data format. This is a straightforward ASCII format for data that can be represented as a set of instances with a fixed number of attributes for each instance.

2.6 IDL

The Interactive Data Language (IDL) [2] is software for data analysis and visualization. For this project, IDL is used to create 2D plots that become part of the immersive displays. This will be described in more detail below.

2.7 UCI Machine Learning Repository

The UCI (University of California at Irvine) Machine Learning Repository is collection of data sets "that are used by the machine learning community for the empirical analysis of machine learning algorithms" [6]. However, we use them as the raw material on which we test our ideas and techniques for data mining and immersive data visualization.

Chapter 3

Tools Created for this Project

3.1 ivMuseum

We have created a simple 3D virtual environment into which to place our visualizations. This 3D framework is based on a museum metaphor. For a given data set there will be a series of "rooms" - like rooms in a museum. Each room will display some visual representation(s) of the data set. These visualizations could be three-dimensional like a sculpture in the museum or they could be two-dimensional like a picture on the wall. There can be multiple visualizations in a single room and there can be opportunity for some of the visualizations to be interactive.

Each collection of rooms will, of course, be viewable on the workstation and in the RAVE, our immersive visualization environment. This 3D framework will enable the user to move through the rooms and interact with the data.

The museum environment can be seen in many of the figures in subsequent sections. A three room museum is shown in Figure 4.8.

3.2 Glyph ToolBox Additions and Modifications

The main focus of our project was not to expand the Glyph Toolbox but to use it for 3D data visualization. However, there were several occasions where the toolbox lacked the appropriate tools. For instance, the text function that already existed was hard to read when rendered. Also, there was no tool that could be used to label the axes of a graph. The new functions were added to address these shortcomings.

Text Functions

These functions were designed to be used in conjunction with the `gtb-text` tool. The `gtb-thicken` function creates a rectangle that is centered around each line (polygons based on two points) of a letter while the `gtb-bubtext` function creates a cylinder that is centered on each line and caps the cylinder with a sphere at each end. Both of these functions are designed to increase the readability of text in the 3D environment. These functions will also work for files that are not explicitly text. The `gtb-bubtext` and `gtb-thicken` functions are designed to filter a GTB file for any lines and extrude them in two or three dimensions respectively.

Geometric Shape Functions

These functions are additions that were added to the GTB library to allow for the creation of basic 2D figures. The `gtb-line` function draws a line and the `gtb-square` function draws a square. The user can specify the size of both line and square.

Museum Display Functions

A problem we encountered early on was that displaying the museums was problematic on the slower machines. This was due to the fact that the display of texture-mapped objects was much slower on machines without texture mapping graphics hardware. The `gtb-museum` function creates a museum structure almost identical to that of the `ivmuseum` function. The key difference is that `gtb-museum` does not use a brick texture map. Instead, it draws a red square and uses white lines to recreate the brick effect.

The `gtb-axisl` function creates a labeled `gtb-axis`, using `gtb-bubtext` to label the x, y and z axes. It also has options for creating labels for the size and color of the glyphs, two attributes which are used in nearly every graph. By automating the axis creation process, we were able to streamline the graph creation process. The labeled axis increases instant interpretability.

3.3 Other Alternatives to the Museum Environment

In addition to `ivMuseum` and `gtbMuseum`, data were displayed in other types of environments. For example, a semi-circular museum arrangement was developed

by Sean Kelly. This can be seen in Figure 4.12. This concept was used in conjunction with ivMuseum by Harry Bullen (Figure 4.18). A multi-tiered museum was created by Christian Brown (Figure 4.4), and a forest metaphor was used by Robert Xiao for the Forest Cover data set (Figure 4.5).

3.4 Visual Component Tools

These tools were developed by SAVG at NIST in preparation for this project. They aid in transforming a data file into the following visual representations:

- 3D Glyph-based display
- Parallel coordinate plot
- PDF / PMF plot
- Attribute Rank plot

The way that these tools fit together is described below in the *Dataflow* section.

getAttrOrder (order attributes)

This program reads an ARFF file and ranks the attributes based on a criterion specified on the command line. It uses the utility programs rankAttrByOutliers and extract2Attr to derive two of the rankings and it uses Weka to derive all of the other rankings.

The methods available for ranking attributes are:

- J48MAE (decision tree based on mean absolute error)
- J48Kappa (decision tree based on kappa statistic)
- InfoGain
- ChiSquared
- GainRatio
- OneR (one level decision tree)
- ReliefF

- SymmetricalUncert
- outlier (rank by number of outliers)

arffToGtb (generate 3D representation of data)

This program reads an ARFF file and an attribute ordering file and produces a 3D glyph-based representation of the data. Note that this is a completely generic program, however it can be customized to enhance the resulting display. This program uses the Glyph Tool Box to generate the 3D geometry.

arffToPcp (make parallel coordinate plot file)

This program reads an ARFF file and an attribute ordering file and produces a parallel coordinate plot file. This file is not an image, but is an intermediate file that describes the parallel coordinate plot. Note that this is a completely generic program, however it can be customized to enhance the resulting display.

pcpToImg (draw 2D parallel coordinate plot)

This program reads the parallel coordinate plot produced by arffToPcp and generates the plot in the form of an image file. It uses IDL to actually plot the data.

A parallel coordinate plot [11] is 2D display of multidimensional data. Data with any number of attributes can be represented. Each attribute dimension is represented by an axis, but unlike the standard representation in which the axes are placed at right angles to one another, in parallel coordinates the axes are all parallel to one another. Figure 3.1 is an example of a parallel coordinate plot.

arffToDistImg (draw pdf / pmf plot)

This program reads an ARFF file and an attribute ordering file (which can be produced by getAttrOrder), and produces a PDF/PMF plot in the form of an image file. It uses IDL to actually plot the data. Note that this is a completely generic program, however it can be customized to enhance the resulting display.

Figure 3.2 is an example of a PDF/PMF plot. In this figure, The four attributes petalWid, petalLen, sepalWid, and sepalLen are continuous variables and each is represented by a PDF plot. In each PDF plot, the value of the attribute for each instance is plotted with an "x". This is overlaid by a "box plot" [7] that indicates

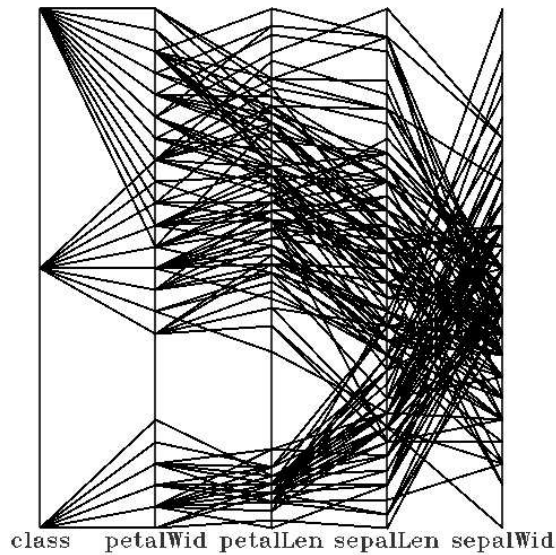


Figure 3.1: Parallel coordinate plot.

the central body of the distribution. The lower end of the box is at the 25-th percentile of the distribution, the top of the box is at the 75-th percentile of the distribution. So the box encompasses the central 50 percent of the distribution. In addition a center line is drawn at the median.

The attribute labeled "class" is a nominal attribute that has only three possible values. The PMF plot for a nominal attribute is a bar chart in which the length of each bar indicates the number of instances for a given value. So in this case there are three bars corresponding to the three possible values for the attribute class.

drawAttrRankImg (draw attribute rank plot)

This program reads an attribute ordering file (which can be produced by getAttrOrder), and produces an attribute rank plot in the form of an image file. It uses IDL to actually plot the data. Note that this is a completely generic program, however it can be customized to enhance the resulting display.

The attribute rank plot is a graphical presentation of the measure that has been used to order the attributes. Figure 3.3 is an example of an attribute rank plot. This is an attribute rank plot of iris data based on the information gain measure. It

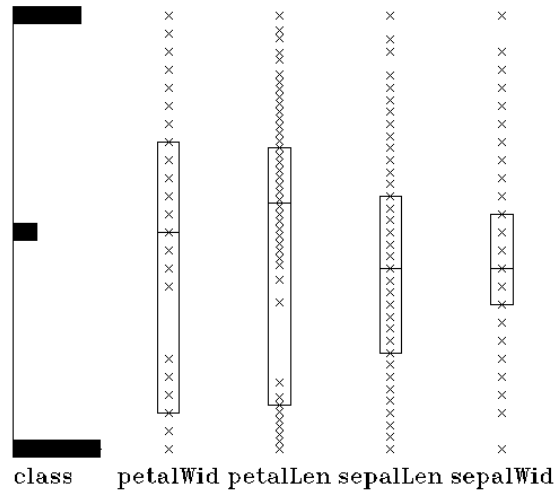


Figure 3.2: PDF/PMF plot.

indicates that according to the information gain measure, petal length is the most important attribute, followed closely by petal width. Sepal length width are both considerably less important.

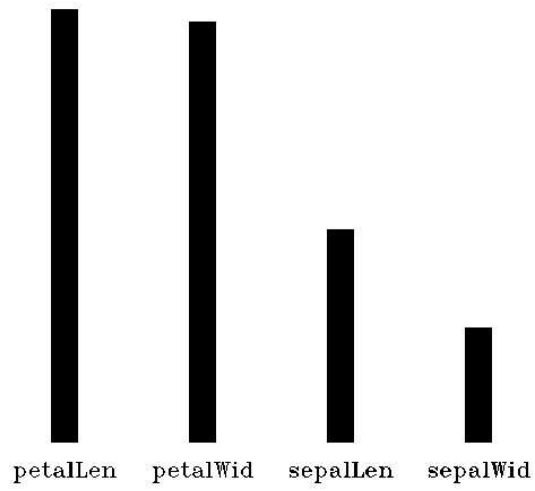


Figure 3.3: Attribute rank plot.

3.5 Dataflow

Figures 3.4, 3.5, and 3.6 give a picture of the dataflow for this project and how the the visual component tools relate to the rest of the system. Note that these figures represent only one of the possible ways that the tools could be used. For any given data set, the tools might be assembled in somewhat different ways.

Figure 3.4 is an overview diagram that divides the processing up into two main steps. The first step, "make visualization components" (I) transforms the original data file from the UCI Repository into two dimensional or three dimensional visual representations. This step broken down into component pieces in Figure 3.5. The two dimensional representations are images, the three dimensional representations are geometric files, either gtb files produced by the Glyph Tool Box, or inventor files.

The second step, "visualize" (II) takes these two and three dimensional components and melds them together into a common coordinate system within the museum-room environment. This step is broken down into its component pieces in Figure 3.6. When the components have been integrated together in this way, then the result can be displayed either on the workstation screen or in the immersive environment of the RAVE.

Overview of Dataflow Summer Project 2003

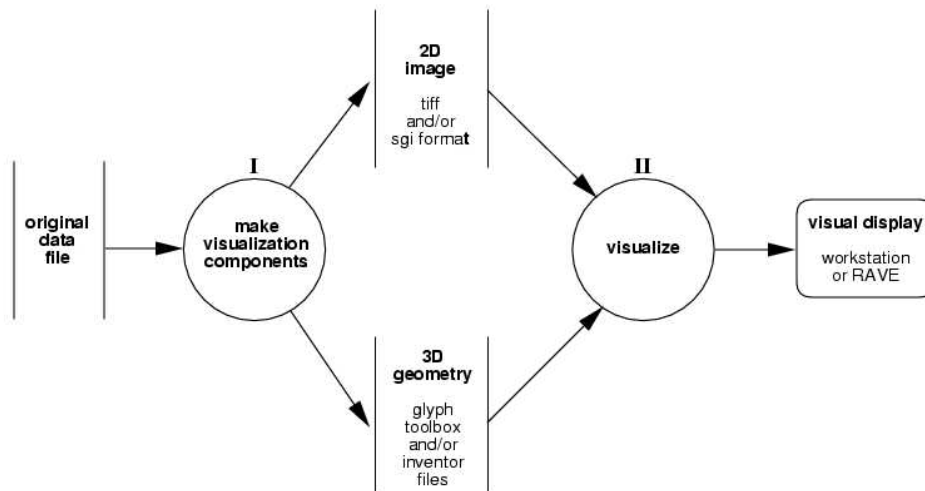


Figure 3.4: Overall dataflow of project.

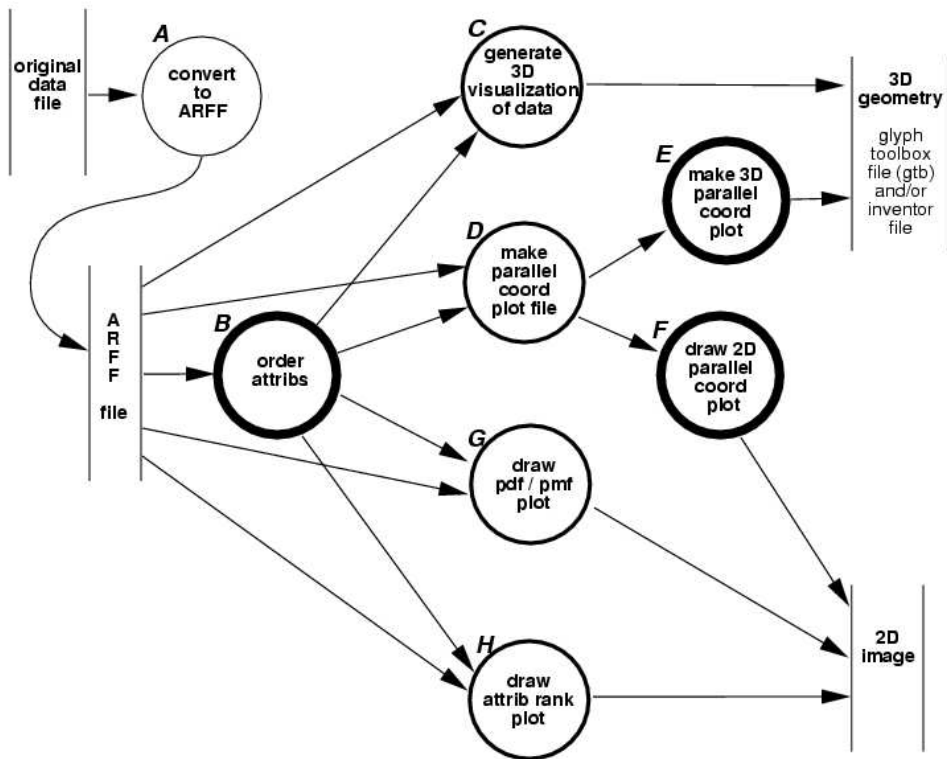


Figure 3.5: Dataflow for creation of visual components.

II. Visualize

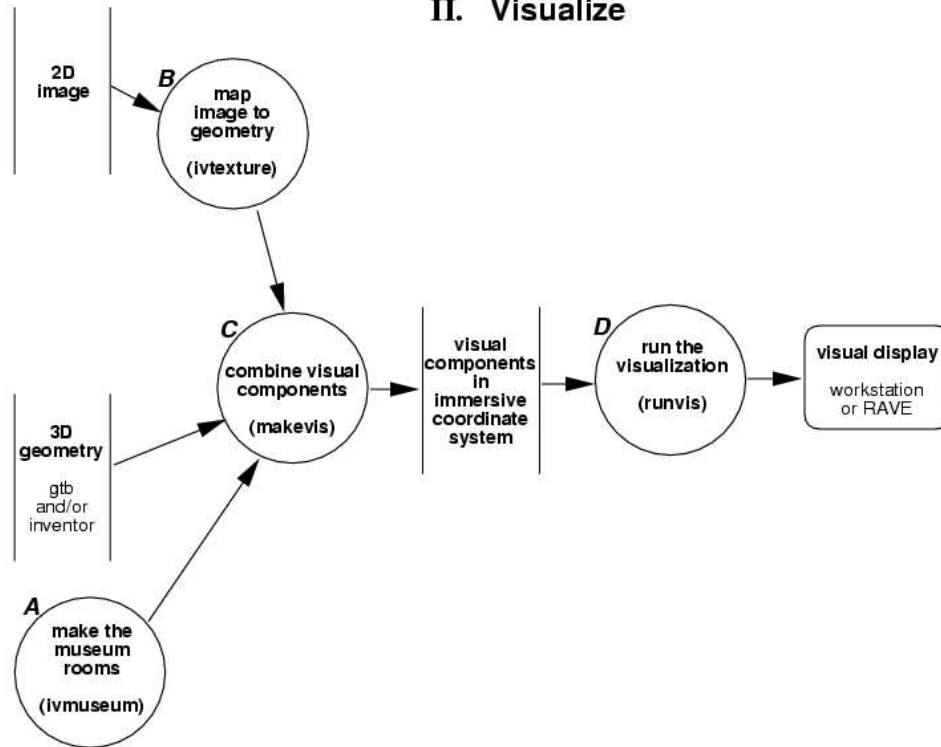


Figure 3.6: Dataflow for visualization of the components.

Chapter 4

Data Sets Analysed

4.1 Tic-Tac-Toe

This data set was analysed by Christian Brown and Robert Xiao.

Overview

The tic-tac-toe data set consisted of ten attributes, defining the end-game appearance of a tic-tac-toe board in which one player won. The first nine attributes corresponded to each of the squares on the tic-tac-toe board, and could be x, o, or b, representing an X, an O, or a blank square, respectively. The tenth attribute represented which player won, and was used as the target attribute in the statistical analysis.

Analysis

As the first data set we analyzed, the analysis was relatively simple, consisting of a ChiSquared test to determine which attributes were most important. Due to the highly discrete nature of the data, we chose to override the automatically determined x-axis, y-axis, z-axis, color, and size dependencies. Since each corner was essentially the same, as was each side, we made the x-axis correspond to the number of "X"s in the middle square, the y-axis the number of "X"s in the corner squares, and the z-axis the number of "X"s in the side squares, with color representing whether X won, and size representing how many data points overlapped at a given location on the graph.

We also analyzed the data by grouping the possible locations of the tic-tac-toe grid into three separate categories: corner, middle, and side with corner as the four corners, middle as the center square, and side as the remaining squares. The number of occurrences of Xs were summed up and displayed on a graph with the x-axis representing the number of Xs in the middle square, the y-axis representing the number of Xs in the side squares, and the z-axis. A gradient between yellow (for more X victories) and white (more X losses) was used to encode the percent of victories for X in that location. The size of the data point was determined by the number of overlapping data points at a given location in the graph.

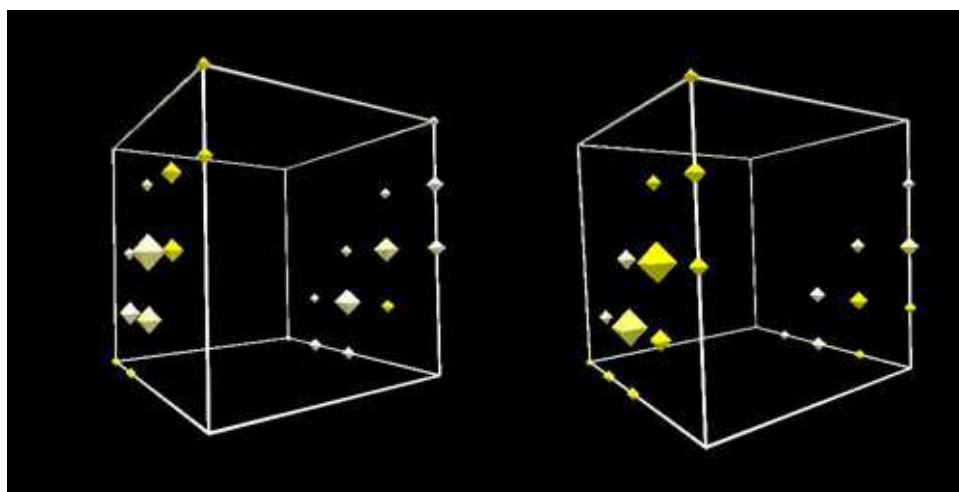


Figure 4.1: One example of our tic tac toe graphs.

Conclusion

The primary lesson learned from this data set was that discrete data are very hard to organize well. Graphs are much more meaningful if there is a range of data being displayed, not merely one of three possible values. We decided to use more continuous data sets in the future.

4.2 Miles Per Gallon

This data set was analysed by Christian Brown.

Overview

The Miles Per Gallon (MPG) data set consisted of data regarding the engines of numerous cars. Each car had 8 attributes, intended to be used to predict miles per gallon for each car. The attributes were a mix of both discrete and continuous data, with the intended target (miles per gallon) being continuous.

Analysis

Initially, we attempted to run a ChiSquared test on these data to determine which attributes were most related to the miles per gallon the car received. Unfortunately, this failed due to a previously unknown limitation of the `getAttrOrder` function: the target attribute was required to be discrete. As such, we were forced to use the number of cylinders in the engine as the target, as this was the most easily decimated attribute. Once we determined which variables were most important, they were assigned to the x-axis, y-axis, z-axis, glyph size, and glyph color variables, in order of decreasing importance. Because this failed to display whether a particular vehicle's engine had 3, 4, 5, 6 or 8 cylinders, we divided the data into 6 graphs, one for each number of cylinders and one that combined them all. We also used a `gtb-axis` and `ivtext` to create a key for the graph, explaining how attributes were displayed on the graph. All of this was placed into one large `ivmuseum`.

Conclusion

The primary lesson from this data set was that we could only have discrete target variables - a significant limitation. We also explored the use of labeling systems and layouts for the first time, attempting to create an immediately understandable set of graphs, with reasonable success. Dividing up the graphs turned out to be a very effective technique, and was used more in later data sets. We decided to explore labeling systems and layouts more in the future.



Figure 4.2: A closeup of two of the MPG graphs.

4.3 Housing

This data set was analysed by Robert Xiao.

Overview

The housing data set consisted of data regarding 506 houses in Boston, Massachusetts. Thirteen continuous attributes, including a target variable of median housing value, and one binary attribute (ignored in this analysis) were included with the data set.

Analysis

For this data set, we ran every single attribute ranking algorithm. In order to use statistical functions generally reserved for discrete data sets, we created a class containing each of the target variables possible values. Another problem that surfaced was with the attribute rankings that were produced by the ranking algorithms. The ranked attributed file that was produced with the intended target attribute contained errors. This problem was fixed by switching the target attribute to the age of the house. When the graphs were produced, we discovered that there were only three distinct rankings. The final three algorithms selected were ChiSquared, ReliefF, and Outlier. The three graphs were displayed in a three room museum structure using ivmuseum and labeled using ivtext.

Conclusion

What we learned from this data set is that the number of ranking algorithms could potentially, depending on the data set, be reduced to about three or four.

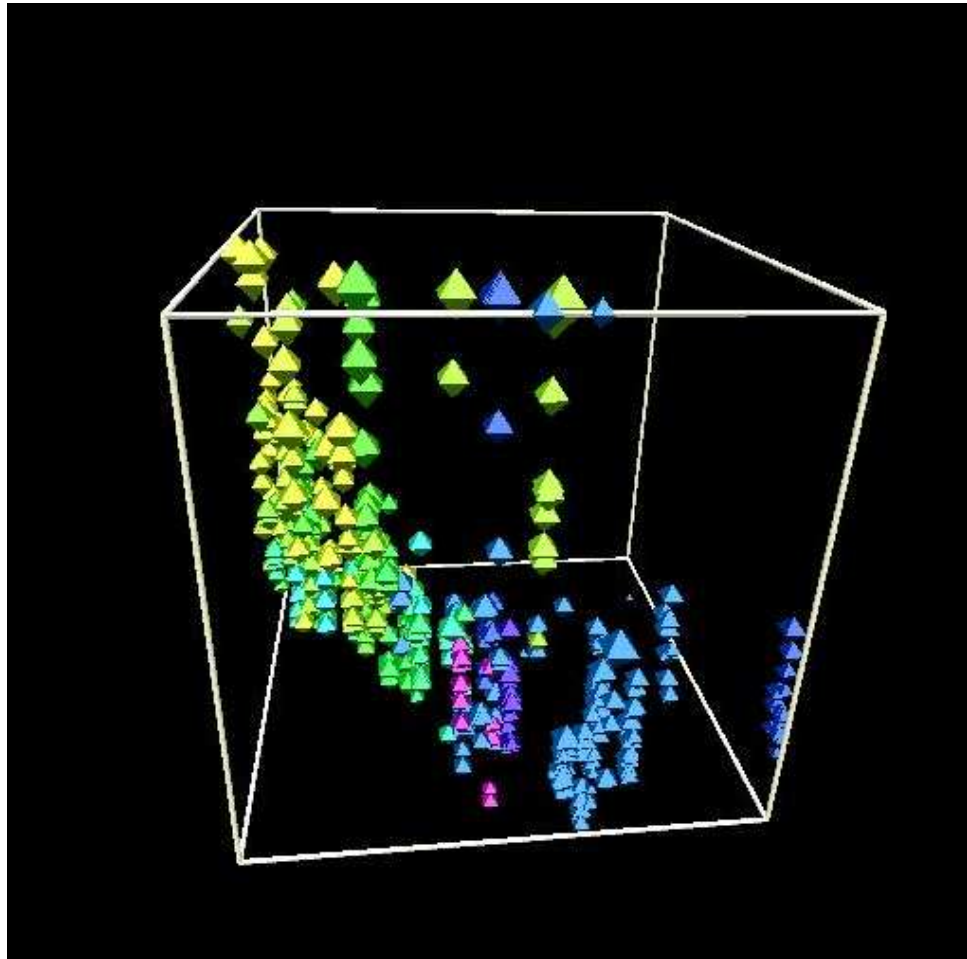


Figure 4.3: The graph of the ChiSquared analysis of the housing data set.

4.4 Image Segmentation

This data set was analysed by Christian Brown.

Overview

The Image Segmentation (Seg) data set consisted of data relating numerous analyses of the colors in subdivided images to the type of surface in the image. Each image was divided up into small subsections, each of which comprised one point of data. Each data point was composed of 18 different attributes, including one that determined what the image was of: brick face, foliage, grass, sky, window, concrete and dirt.

Analysis

This data set was one of the most complex analyzed, and required several weeks of work. We began by using all nine statistical methods to analyze the data. Each of these nine attribute order files was used to generate a row of seven graphs, one for each kind of image, and one labeled axis. The axis was labeled with the newly created gtb-bubtext, as was each graph. The graphs and axes in each row were arrayed in a semi-circle, the method developed by Sean Kelly, as this allowed the viewer to easily rotate and view each graph, zooming in to focus on one at will, without needing to pan to the left or right. Initially, we attempted to put all nine rows into one "museum," in a vertical array, but the speed of rendering was not adequate. We eventually created a series of "buttons," gtb-octa glyphs that could be selected using the wand in the RAVE. Each button could turn one graph on or off, allowing the user to select which graphs to view, leading to an enormous boost in rendering speed. We added several point lights to light the scene. Finally, we replaced the glyphs used in the graph with gtb-tetras, and developed and used the low-detail gtb-bubtext, to effectively halve the number of polygons required.

Conclusion

First and foremost, we ran up against the slow speed of rendering these graphs. By implementing user interactivity, we were able to overcome this problem, and give the user much more control over the data being displayed. This was by far our most important achievement, as the immersive and interactive nature of the RAVE is what makes it unique in its data analysis capabilities. Our development of

low-detail models also aided us in making the graphs much more usable, without sacrificing any information.

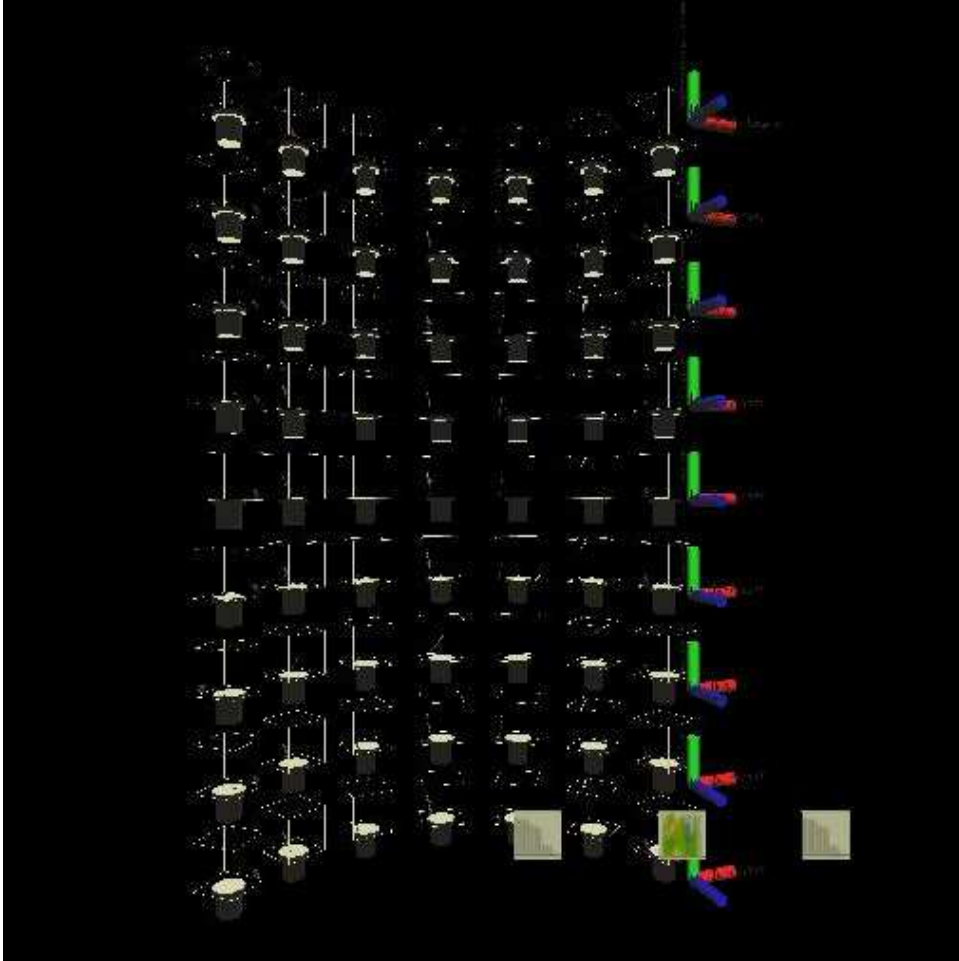


Figure 4.4: Image segmentation dataset: the original 9 by 7 array of graphs.

4.5 Forest Cover Type

This data set was analysed by Robert Xiao.

Overview

This data set was collected as part of a study to determine the effects of environmental conditions on the type of forest tree cover. There are a total of 581,102 records each with 55 columns of data. 40 of the columns were binary columns to designate the type of soil present at each site and 4 of the columns were binary columns to designate the location where the data were collected. The final attributes selected for the analysis were the other 11 columns which consisted of: elevation, aspect (in degrees), slope, horizontal distance to hydrology, vertical distance to hydrology, horizontal distance to roadways, shade at 9AM, shade at 12PM, shade at 3PM, horizontal distance to fire breakout points, and the cover type designation.

Analysis

This data set was the one of the largest data sets we encountered. Graphing all the data points was impossible because of the performance limitations of the RAVE, so only 3500 of the data points were plotted. To analyze this data set, we ran the ChiSquared ranking algorithm to identify the five most important attributes. We then created two separate graphs using variables of the same nature. For example, one graph consisted of the data for shade at 9AM, 12PM, and 3PM representing the x, y, and z axes respectively, with the size representing the slope of the terrain and the color representing the elevation. To further analyze the data, each graph was then split up by the cover type so there were eight graphs in all for each analysis method. For this data set, we decided not to use the conventional display environments which consisted of graphs being placed in a building of some sort. Instead, we created a forest using trees made of cones and cylinders and embedded the graphs in the midst of the forest. The graphs were set up in the new semi-circular arrangement to allow the user to easily view all the graphs by simply standing in the center and turning to the left or the right. Instead of using gtb-axis1 to display the label, we opted to use a sign to display the variables represented by the axes and the color and size of each data point.

Conclusion

The main problem we encountered with this data set was the enormous number of instances. In the final analysis, we ended up using less than one percent of all the data that were available to us. It is very possible that the trends found in the analysis are simply the result of the points that we selected from the entire data set. Some potential solutions to this problem include either upgrading the system to handle a larger number of polygons or creating a series of graphs using different randomly selected data points.

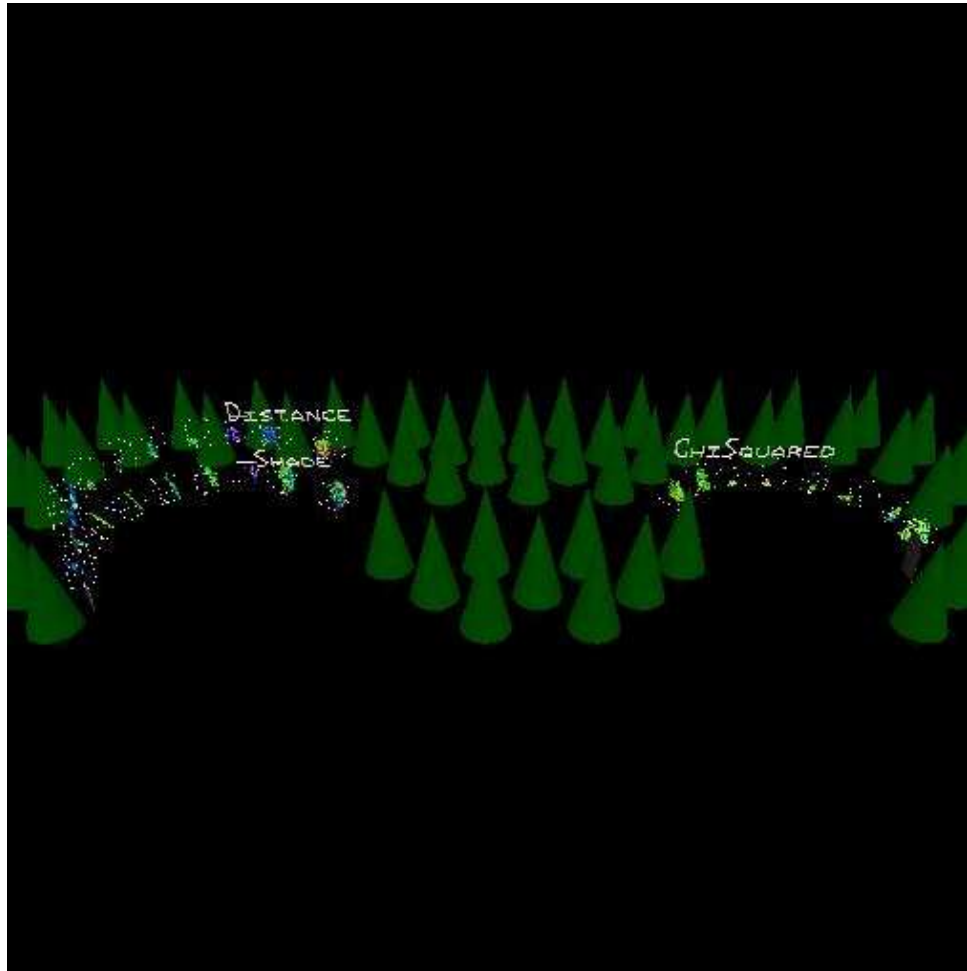


Figure 4.5: A long shot of the full tree cover data set display.

4.6 BUPA Liver Disorder

This data set was analysed by Robert Xiao.

Overview

This data set consisted of 6 attributes: the results of 5 different blood tests which were thought to be sensitive to liver disorders and the number of drinks the particular patient had before the test was administered. There is another attribute but it was disregarded in the analysis because it was only an indicator of which test group the particular subject was in.

Analysis

This data set was relatively small compared to the others. We chose to run every attribute ranking algorithm on this data set because they would not take a long time to run. However, we discovered that J48MAE, J48Kappa, and ReliefF were the only algorithms that produced valid ranked attribute files. J48MAE and J48Kappa produced the same results so we only used J48MAE. For the final museum display, we decided to only use one of the graphs because there were only five attributes. The only difference between the algorithms was the order in which the attributes were ranked. We felt this would be redundant so we arbitrarily chose to display the ReliefF graph. For this graph, we chose to stick with the traditional museum display. Since it was only one graph and the data set was so small, there was no need to produce a large number of graphs.

Conclusion

A problem that surfaced while analyzing this data set was our lack of a basic understanding of the data. We were able to produce a graph that showed trends but we did not understand the significance of these trends.

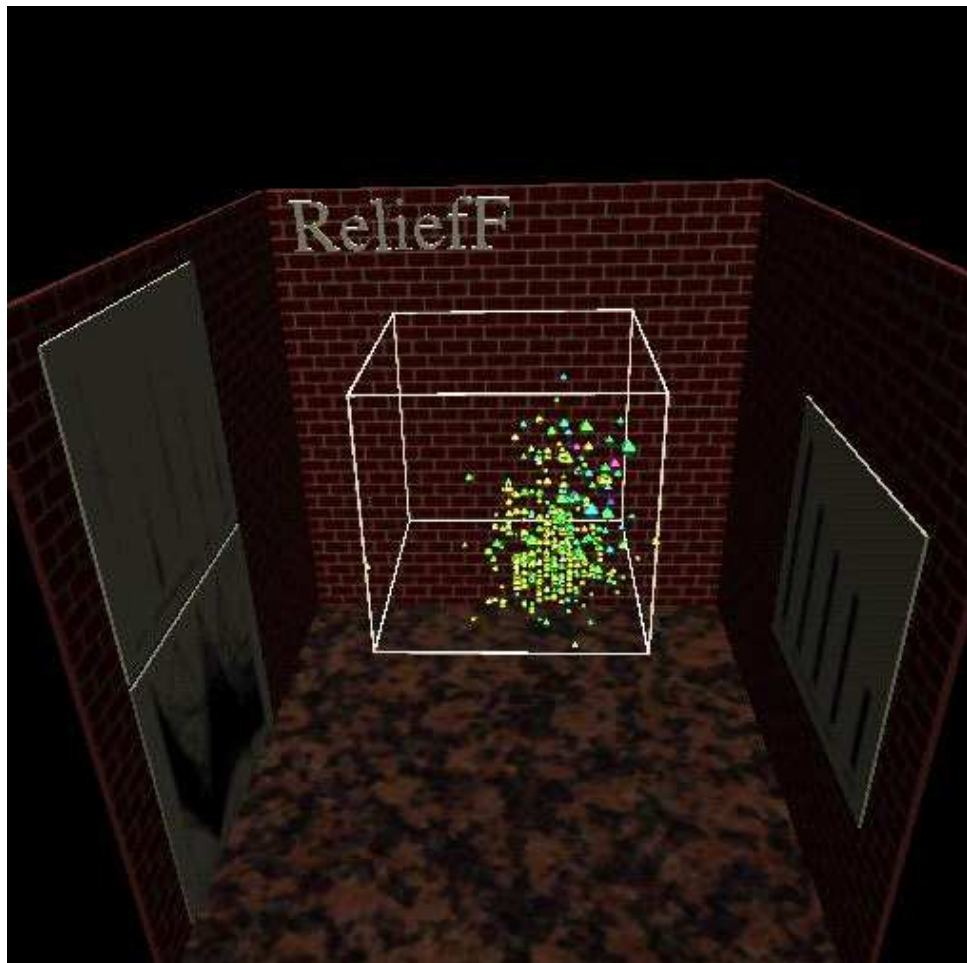


Figure 4.6: The liver data set in an ivmuseum.

4.7 Road Detection

This data set was analysed by Christian Brown and Robert Xiao.

Overview

This was the only data set analysed that did not come from the UCI Machine Learning Repository. This data set was collected by Pat Conrad under the supervision of Dr. Tsai Hong in the Metrology Department at NIST. The data consisted of 27 attributes: a binary column indicating whether the specific region of an image was a road, two columns for the x and y coordinates of the region, and 24 columns regarding the RGB values of the current region.

Analysis

Our first step was to select how best to summarize the color data for each point. Since the red, green and blue values for each segment of the image was divided into 8 bins each, we decided to use a weighted average to calculate an approximate value for each color. We multiplied the value of each bin (the number of pixels falling within that bin's color range) by the average value of the bin (e.g. 7.5 for the first bin), and added all these values together, to achieve an approximation of the color of that segment. Once we had this red, green, and blue value for each point of data, we decided to use these as the x, y and z axes, respectively. We eventually created three kinds of graphs: a two-dimensional image of the scene, generated from our approximate values, with the road pixels elevated from the rest; a three-dimensional graph in which blue glyphs were roads and red were not; and a three-dimensional graph in which the color of the glyphs was determined by the red, green and blue values we calculated for each particular point.

Conclusion

This data set was the first real test of 3D analysis that we performed. We were able to determine patterns regarding the RGB values of the sample images, however the resulting graphs did not provide much use in predicting any patterns in the RGB values of the roads and neighboring terrain due to significant rounding of the RGB values.

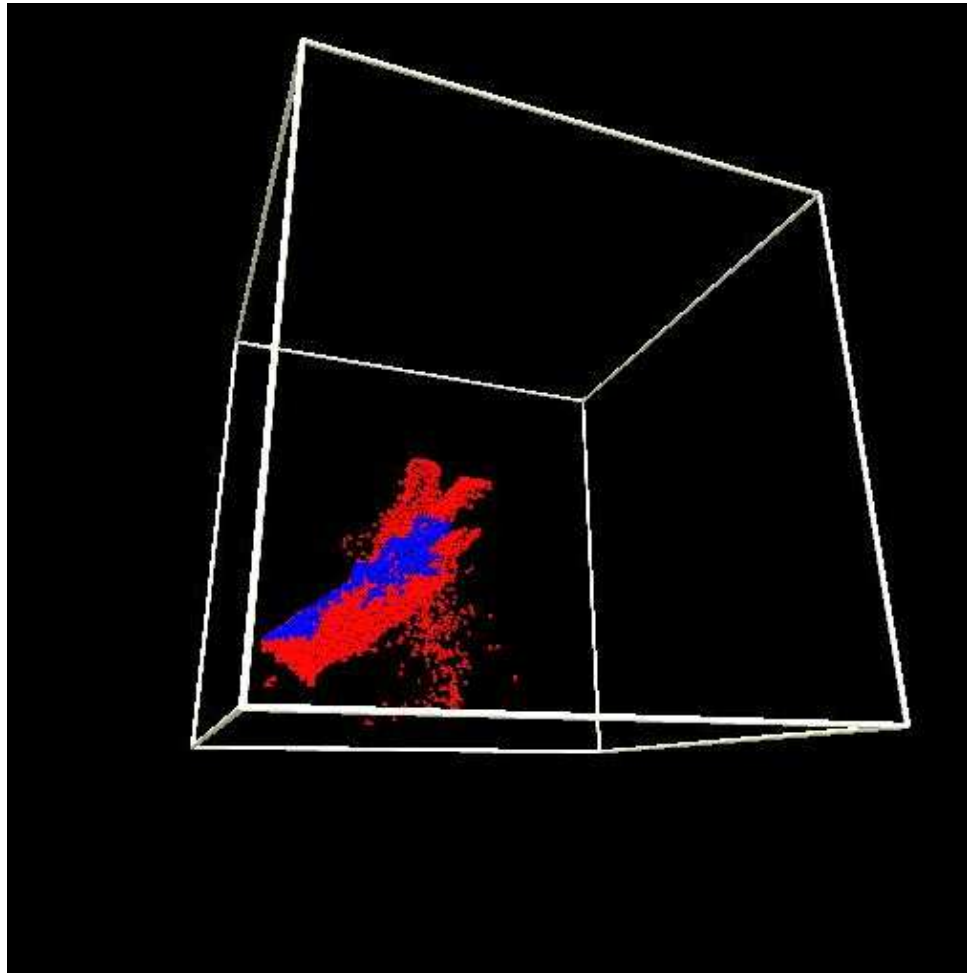


Figure 4.7: The two-color graph for all clear road images.

4.8 Wine

The Wine data set was analysed by Harry Bullen.

This data set contains the chemical analysis of wines from Italy. Wines from three different types of grapes are included. There are levels of 13 chemicals provided for each of 178 instances. The target attribute is the type of grape.

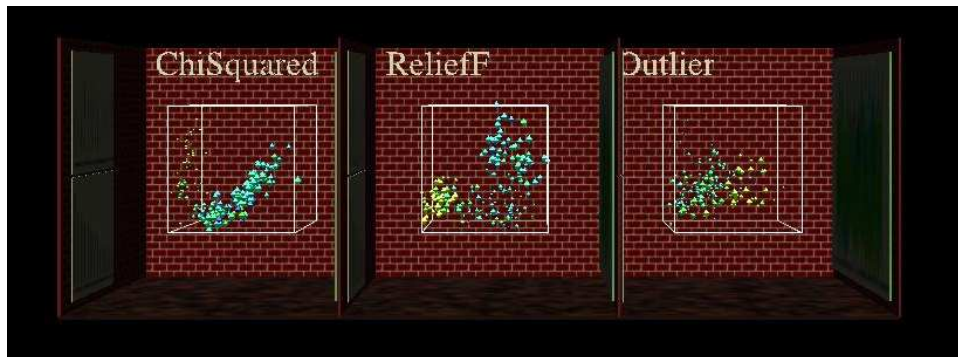


Figure 4.8: Wine data set in museum environment.

4.9 Page Block

The Block data set was processed by Harry Bullen.

This data set concerns to document analysis. Each instance describes a block of a page in a document. Each block was broken into 5 categories: text, horizontal line, graphic, vertical, line, or picture. The objective was to determine what attributes were most meaningful in determining what the block contained. The most important attributes turned out to be the size and shape of the block.

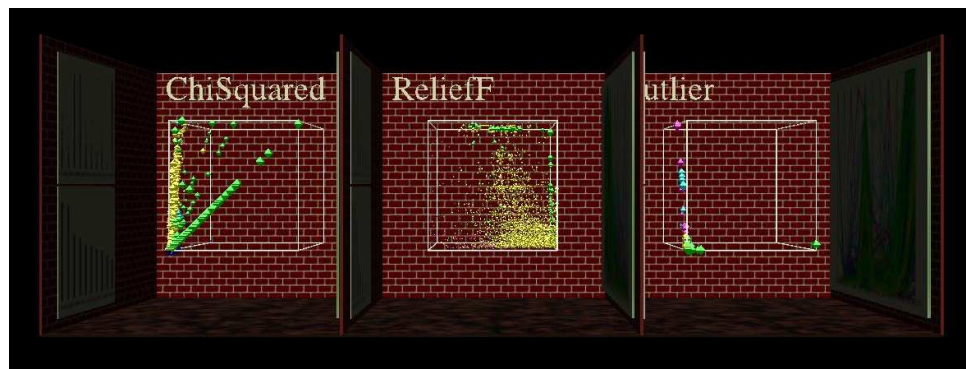


Figure 4.9: Page block data set in museum environment.

4.10 Spambase

The spambase data set was analysed by Sean Kelly.

This dataset contains roughly 4000 instances of 58 attributes each, representing e-mail messages. One attribute is binary, registering whether or not the message is spam, three are numbers describing the shortest, average, and total length of strings of capital letters in the message, and the other 54 are values describing the frequency with which certain key words are used in the message. The data were first evaluated by all ranking methods, but the volume of data caused resultant charts to be unreadable. To counter this, the dataset was broken into multiple sets of 10 attributes- spam flag, capital string attributes, and 6 word-specific attributes each. The smaller datasets were evaluated using GainRatio, ReliefF and outlier ranking, but the resultant museum was still too large to display, so outlier ranking was dropped. In each subset, the capital string attributes rank lower than at least one word-specific attribute. Many word-specific attributes, however, take the value 0 in the majority of instances in the datafile, hence box plots of the data are heavily weighted.

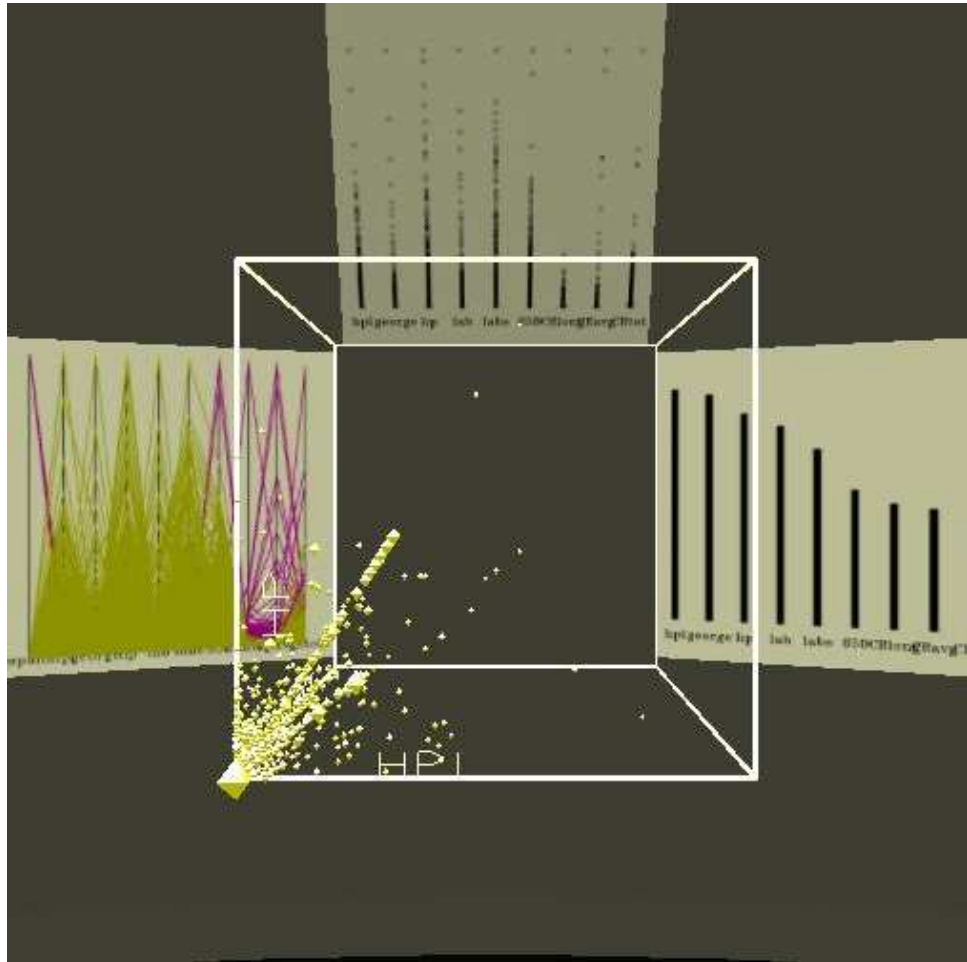


Figure 4.10: Spambase dataset: Graphs of one GainRatio ranked subset (first floor).

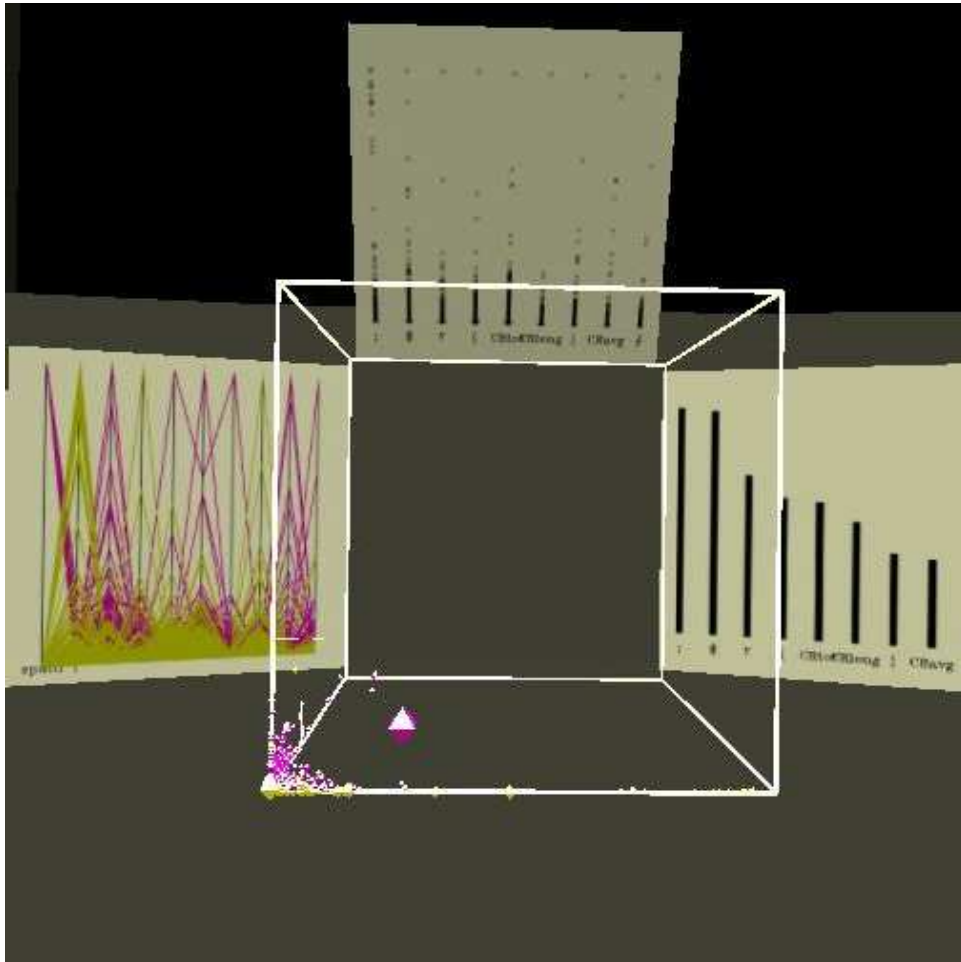


Figure 4.11: Spambase dataset: graphs of one ReliefF ranked subset (second floor).

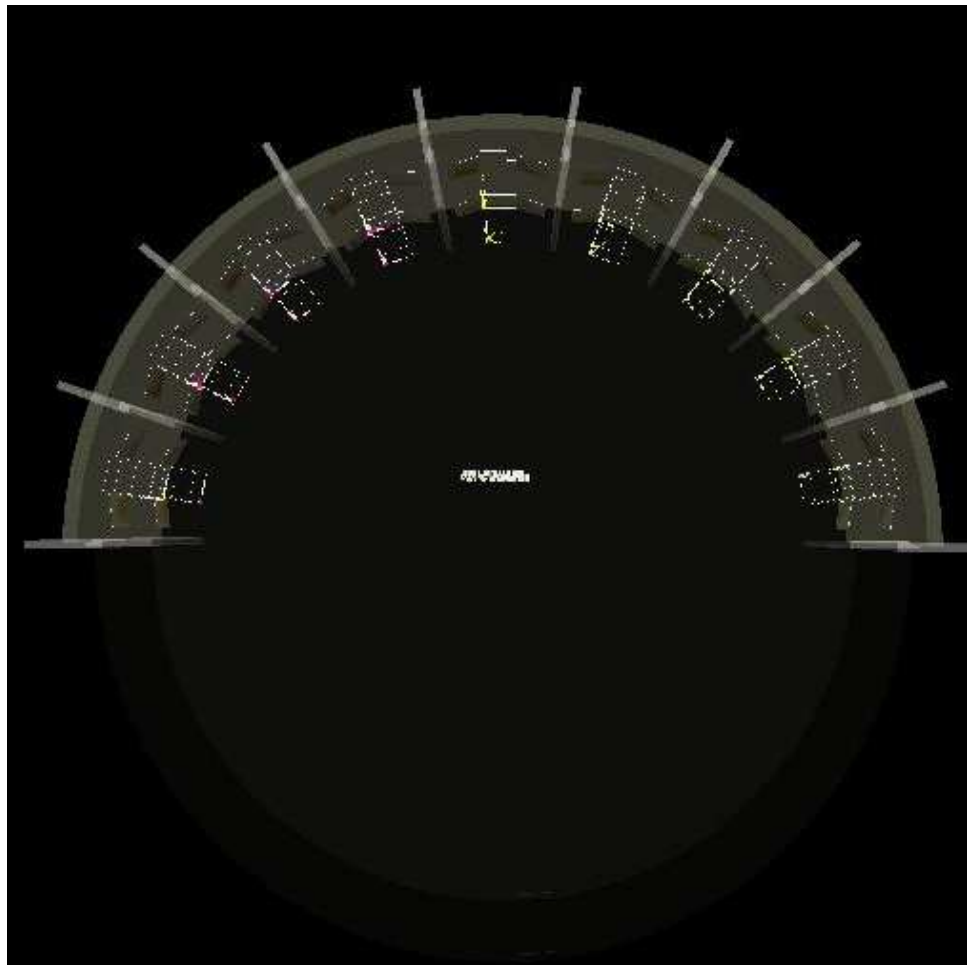


Figure 4.12: Spambase dataset: top view of museum layout.

4.11 CPU

The CPU data set was analysed by Sean Kelly.

This dataset contains 209 instances of computer CPU stats. Prior to processing, instance-unique nominal values had to be removed. The remaining attributes were then ranked by relevance to the chips' manufacturers. All 9 available ranking methods were used on the data. For the final museum, individual manufacturers of more than 8 chips were grouped by room, and a floor of rooms was made for each ranking method. Unfortunately, each floor was so large that no more than 3 of the 9 floors of the museum can be rendered at once, and on low-end machines the screenshot utility used for taking web page pictures has trouble working when even 3 floors are being rendered. To solve this problem, the build script only constructs a single floor, but takes a parameter to determine ranking method. Viewing of the full dataset shows how cpu power tends to vary by manufacturer. While each attribute apart from manufacturer is numeric, all attributes but overall performance tend to be standardized values, predominantly powers of 2 x1000. Given the nature of computers, such values are to be expected, although they cause overlap in many displays.

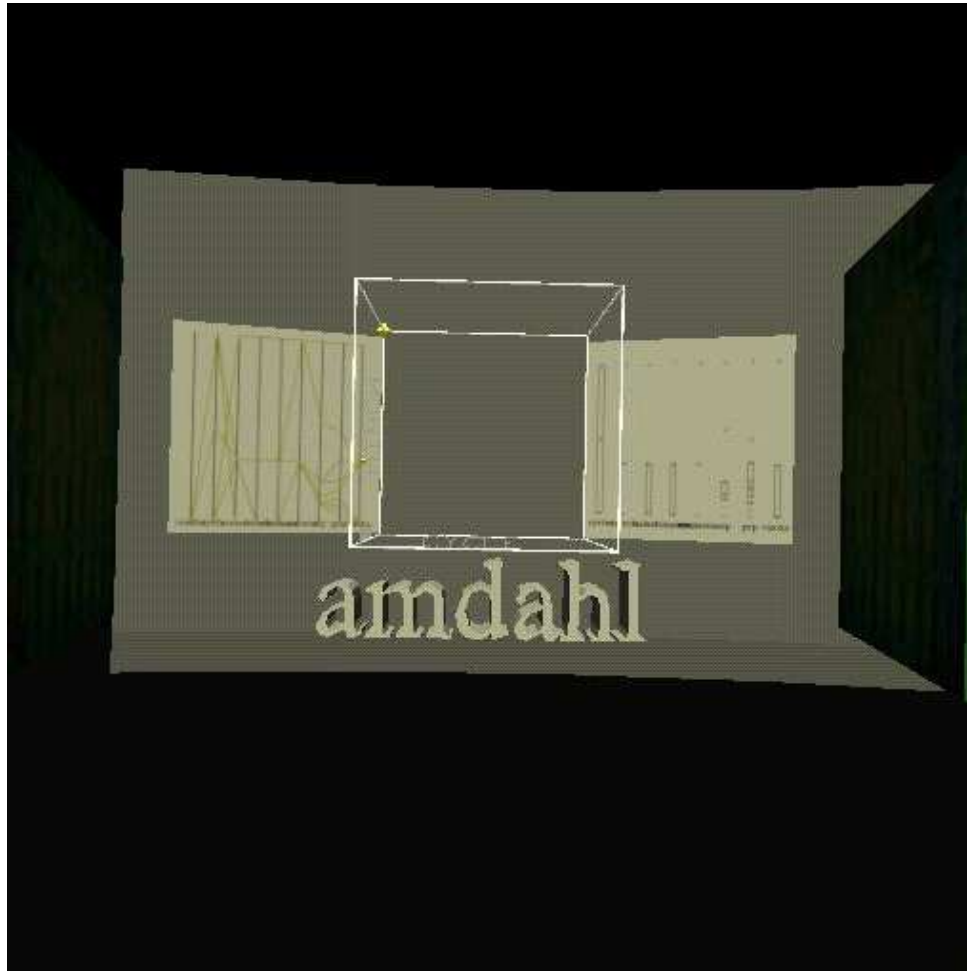


Figure 4.13: CPU dataset: GainRatio floor, data from Amdahl CPUs only.

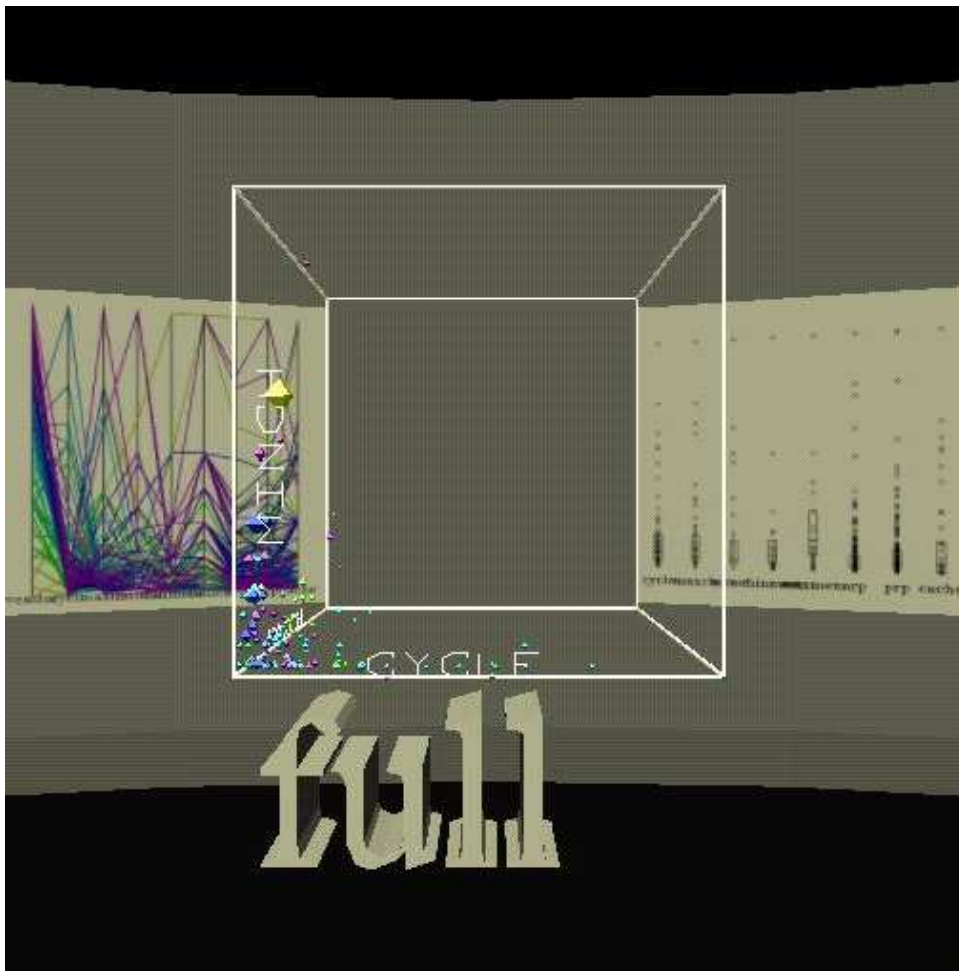


Figure 4.14: CPU dataset: GainRatio floor, full data, 3D plot colored by manufacturer.

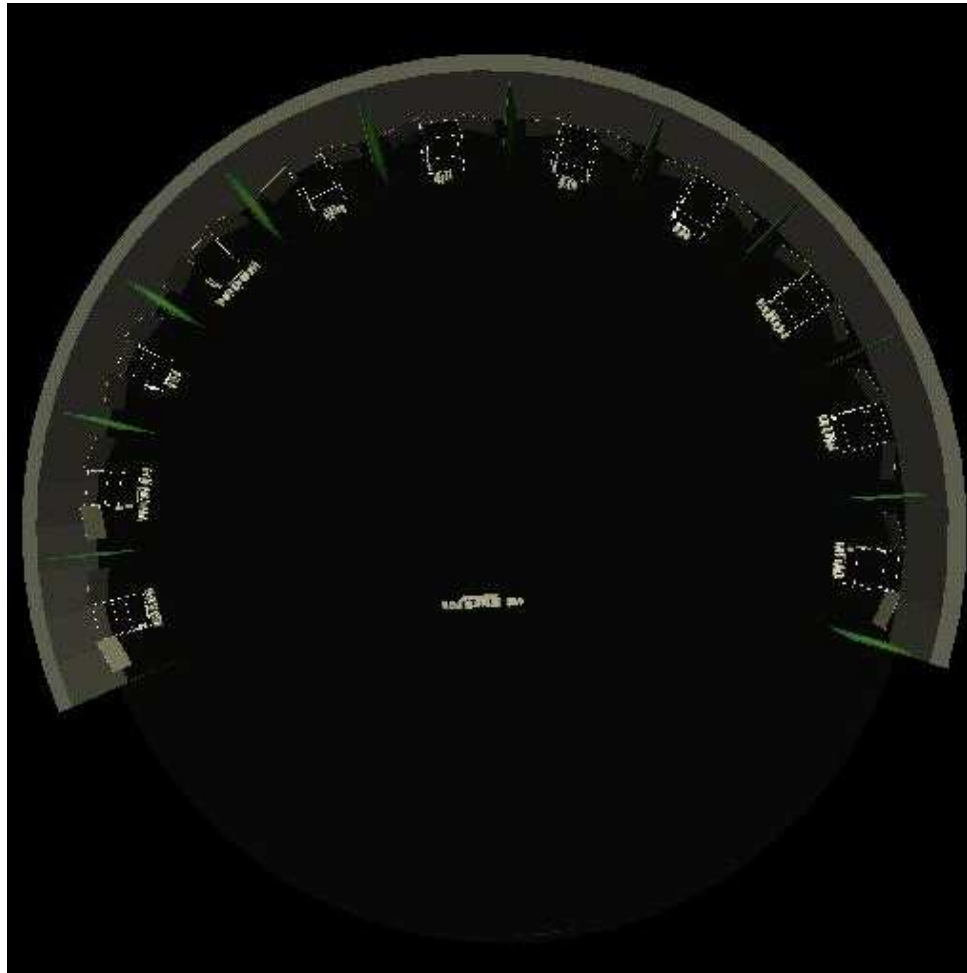


Figure 4.15: CPU dataset: whole floor from above; extended semicircular layout.

4.12 Solar Flare

The Solar Flare data set was processed by Harry Bullen.

This data set has been made up of only discrete variables. Unfortunately the 3D visualization did not produce interesting results. This is because most of the data points overlapped one another and reduced the amount of information shown in the graphs. The most interesting data came from the attribute order files. These described the apparent importance of different attributes in predicting a solar flare. Seven of the tests produced apparently meaningful data. The data representations were displayed in a circular arrangement. In the RAVE the user would start in the center of the circle and would be able to move to any of the different data sets.



Figure 4.16: Solar flare dataset: one room of the museum.

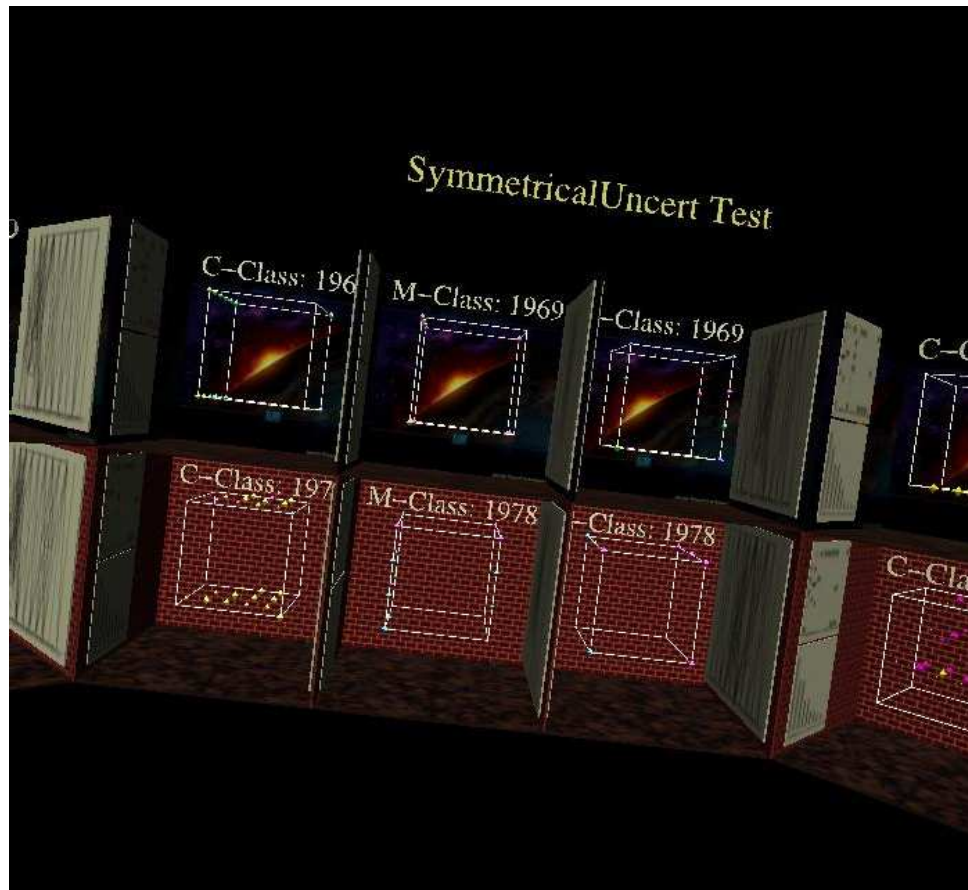


Figure 4.17: Solar flare dataset: one segment of the circular museum.

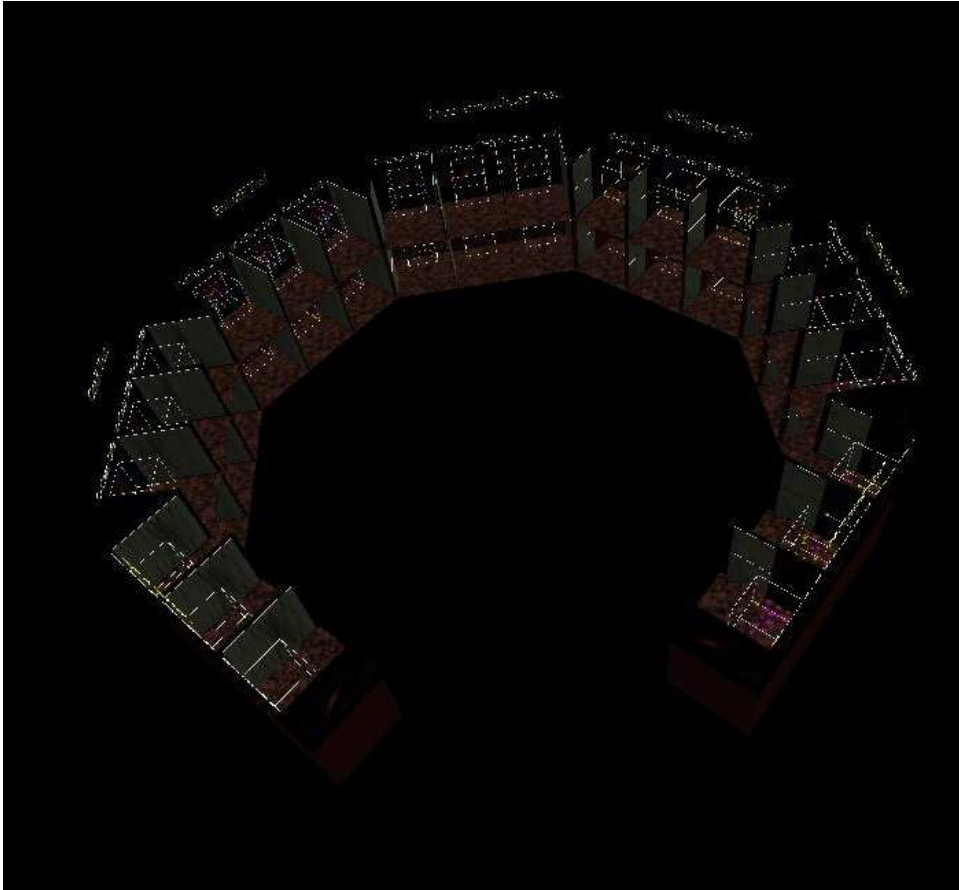


Figure 4.18: Solar flare dataset: an overview of the entire museum layout.

4.13 Abalone

The abalone data set was analysed by Sean Kelly.

This dataset contains roughly 2000 instances of measurements of abalone shellfish. The biggest issue in visualizing it was the lack of distinct target attribute for attribute ranking functions. Only one attribute in the set is nominal, and many functions don't like numeric targets. Thus, for museum display, I evaluated the dataset using 2 or 3 functions targeting each of 3 attributes (2 numeric, 1 nominal). Unsurprisingly, weight showed greatest correlation with size and number of rings (age). The attributes with greatest correlation to number of rings were all weights, indicating abalones get heavier fairly reliably as they age. Gender-based plots were harder to read, but generally showed infants were smaller and lighter than adults of either gender.

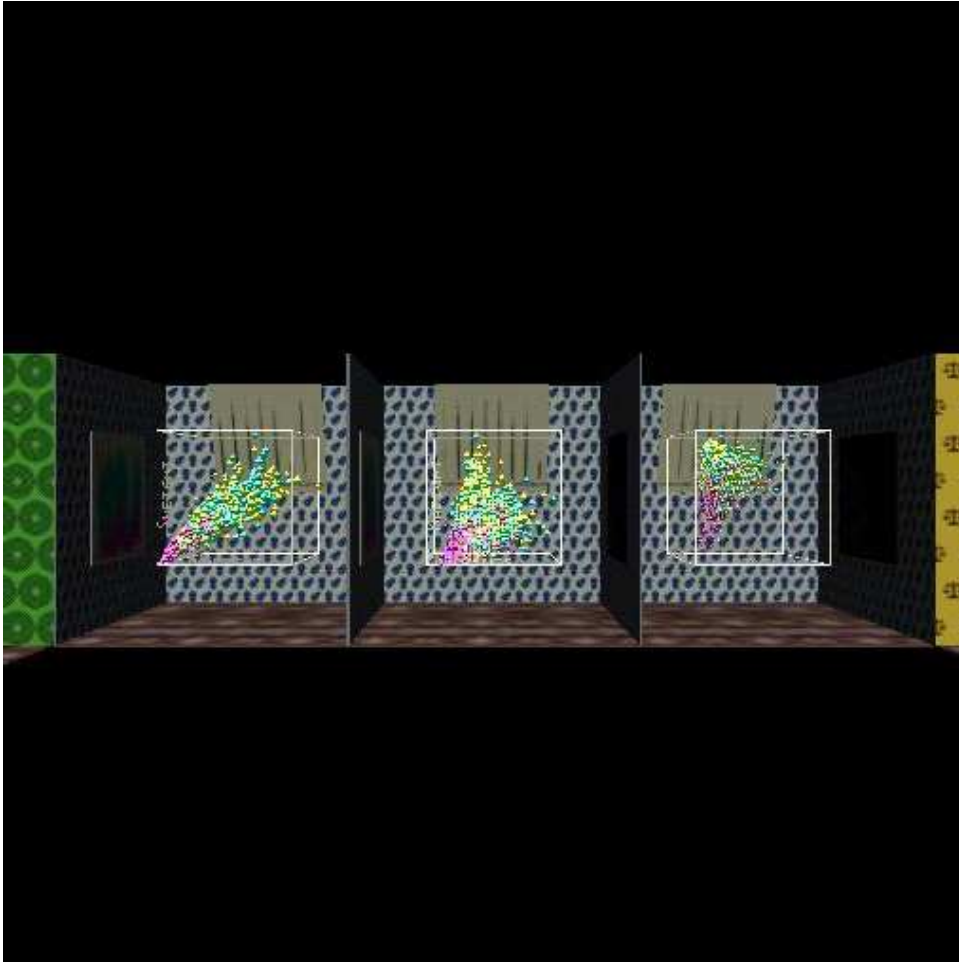


Figure 4.19: Abalone dataset: back 3 rooms of museum, containing gender-targeted ranking by InfoGain, ReliefF and outliers.

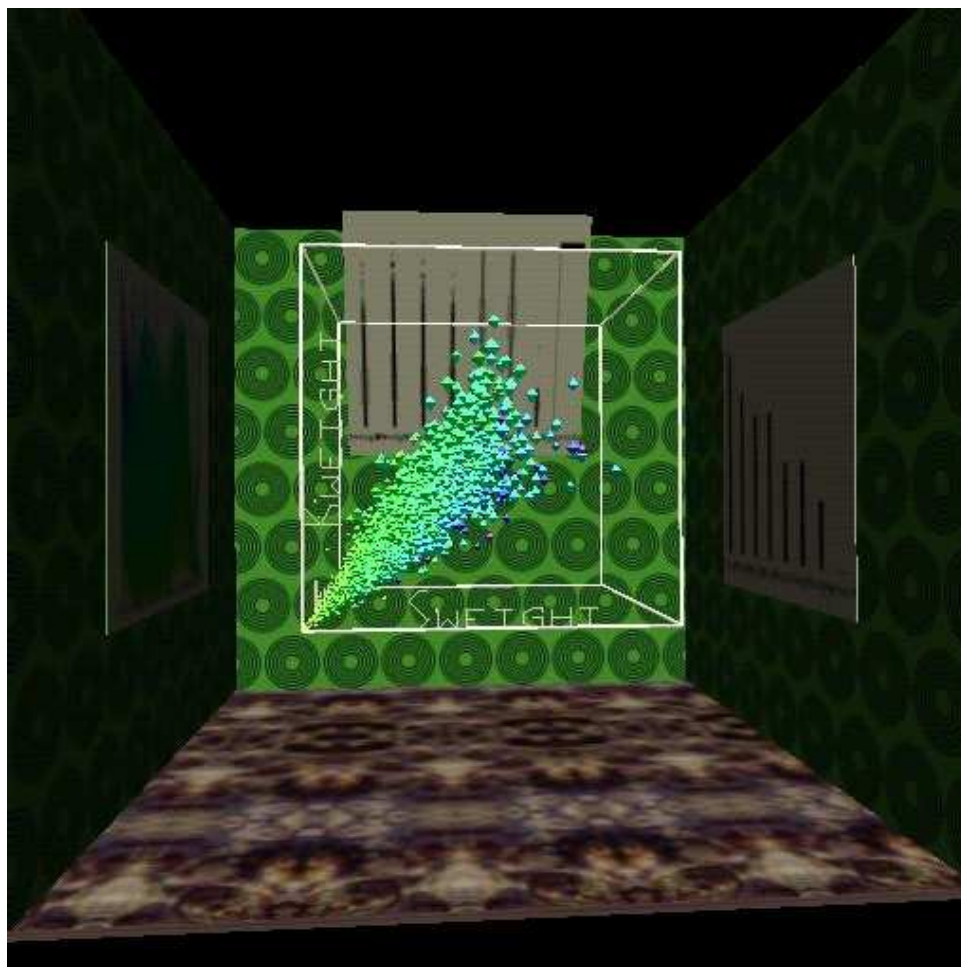


Figure 4.20: Abalone dataset: left side, 2nd room containing rings-targeted ranking by ReliefF.

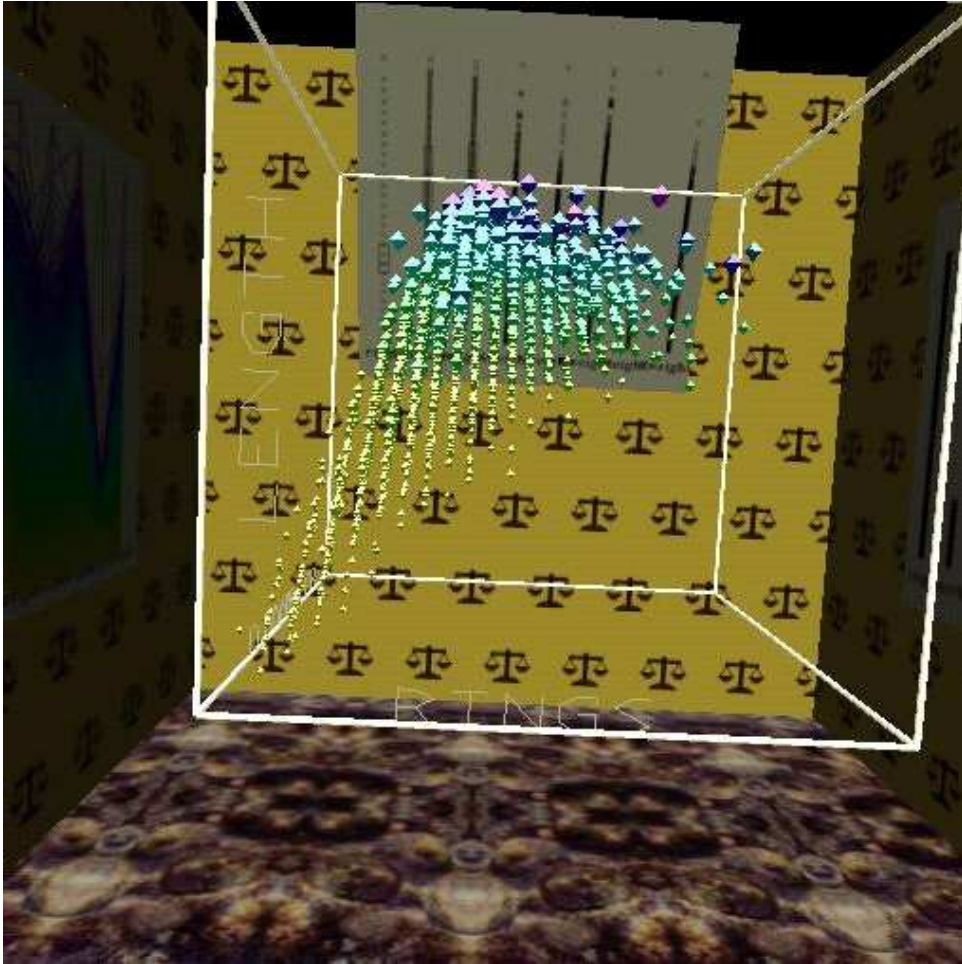


Figure 4.21: Abalone dataset: right side, 3rd room containing whole-weight-targeted ranking by outliers.



Figure 4.22: Abalone dataset: whole museum from above- empty rooms due to lack of support for numeric targets by InfoGain.

4.14 Glass

The glass data set was analysed by Sean Kelly.

This is a purely numeric dataset containing roughly 200 glass samples with information on amounts of various chemical elements in the samples and what purpose the glass serves (window, headlight, tableware, etc.). For processing, the numeric class assigned to each glass sample was treated as an enumerated type. As all other data were numeric, it was easy to run all 9 available attribute ranking methods on the data, and a custom semicircular museum was generated to display all results.

Silicon, on the whole, seems to be the least important element related to a glass' function. Apart from that, each ranking method chose a decidedly different list of important elements.

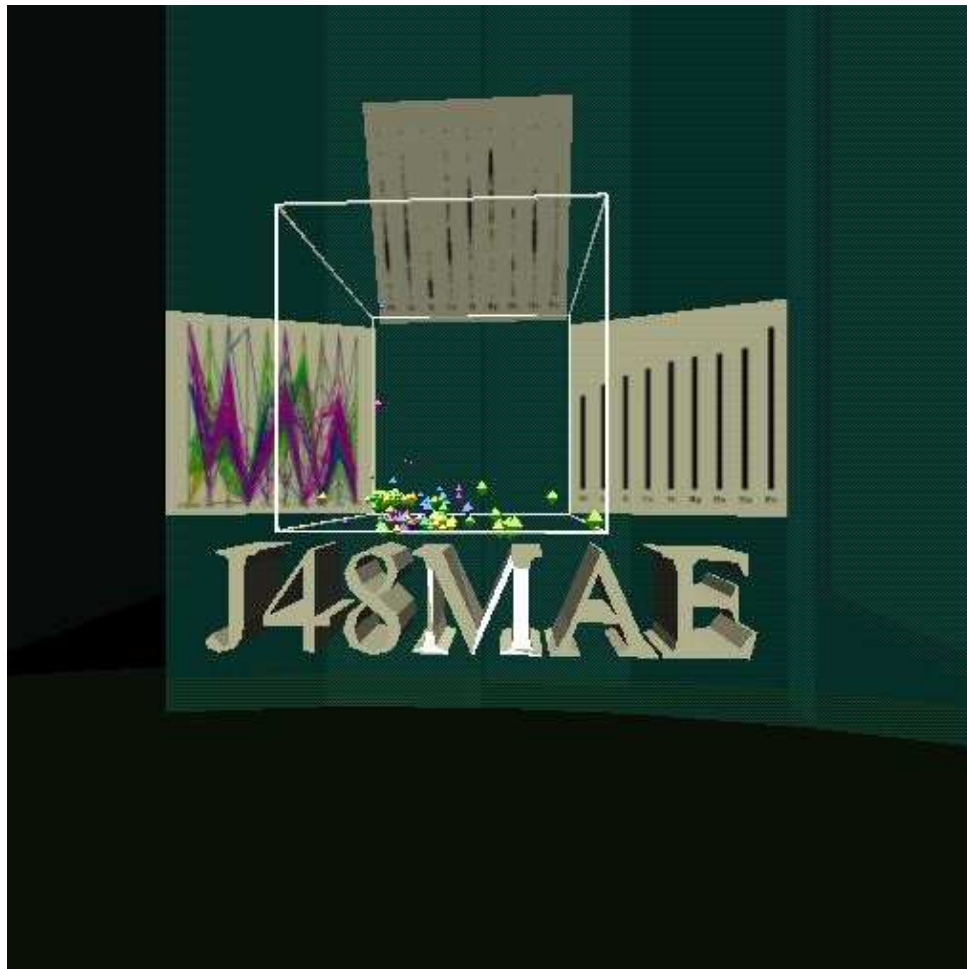


Figure 4.23: Glass dataset: first room, containing data ranked with J48MAE algorithm.

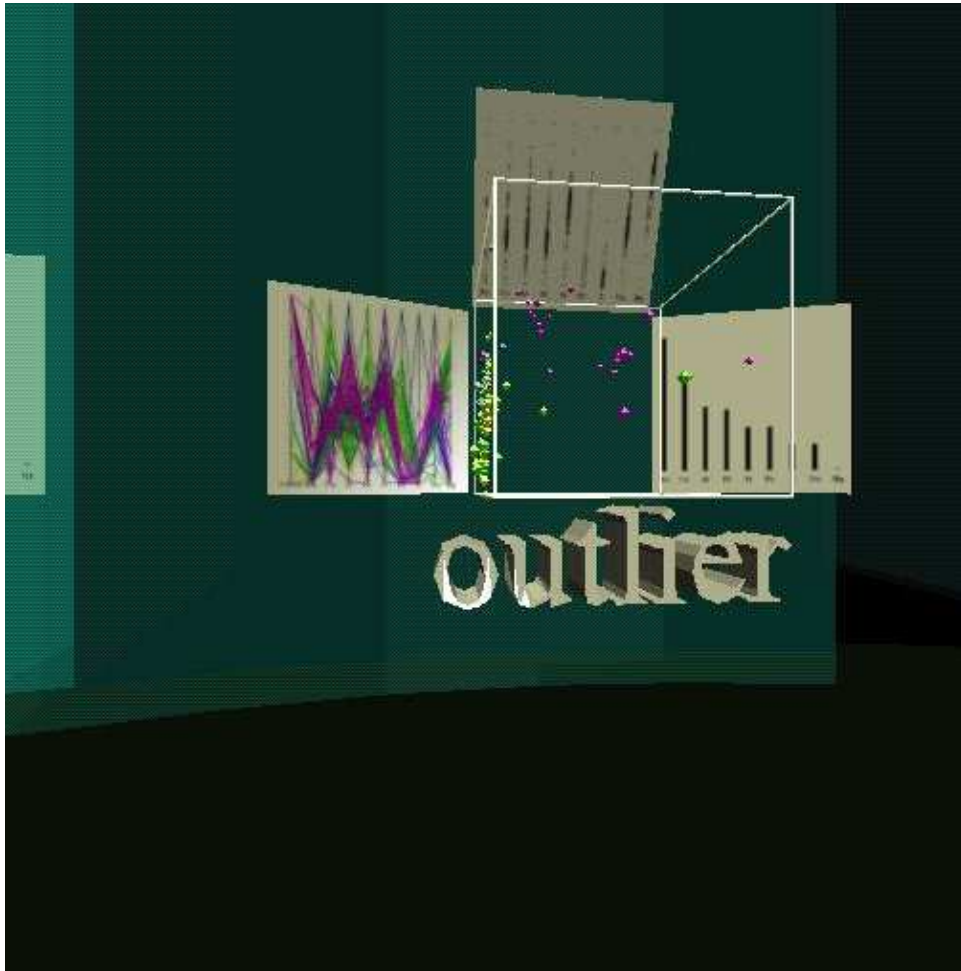


Figure 4.24: Glass dataset: last room, containing data ranked by outliers.

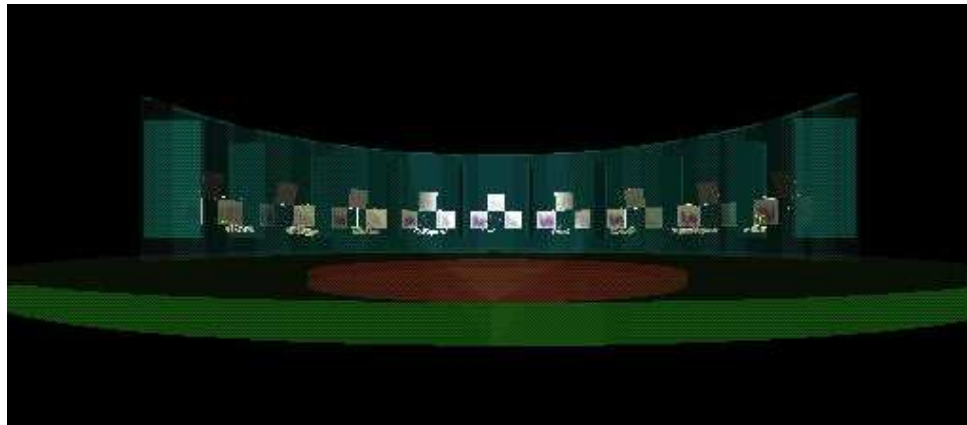


Figure 4.25: Glass dataset: whole museum at distance.

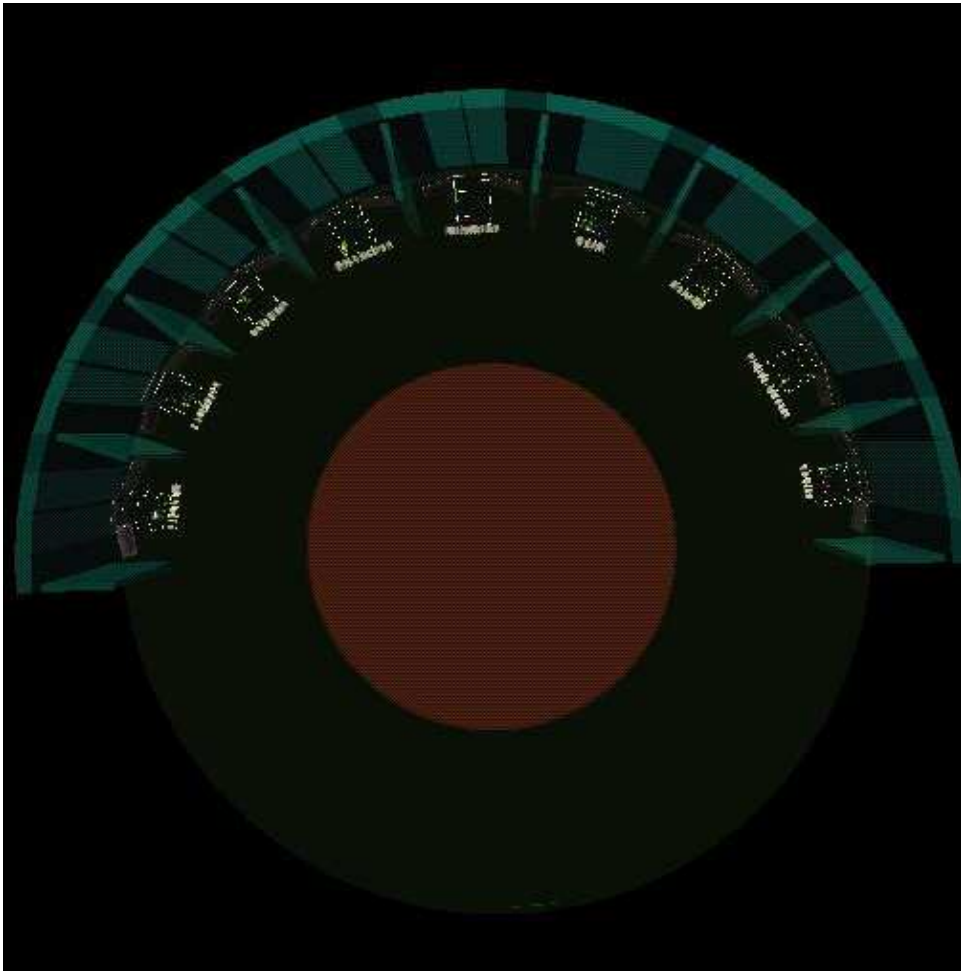


Figure 4.26: Glass dataset: whole museum from above.

4.15 Mushroom

The mushroom data set was analysed by Sean Kelly.

The mushroom dataset reflects all the problems found in the tic-tac-toe data, but on a larger scale. Once again, all attributes are nominal/enumerated, and there are 22 of them compared to tic-tac-toe's 10. There are also 8100 instances of data, making processing, and especially model-building, particularly tedious. For display purposes, as well as modeling the whole dataset, functions were written to filter out and process smaller subsets of related attributes. The final model is a museum containing seven rooms, one holding charts and models corresponding to the whole dataset, the others holding similarly formatted data on each of six attribute subsets. The most important attributes turned out to be ones related to a mushroom's spores, odor, and to some extent gills. This suggests a mushroom's toxicity may result from its spores, whether airborne or within the mushroom itself.

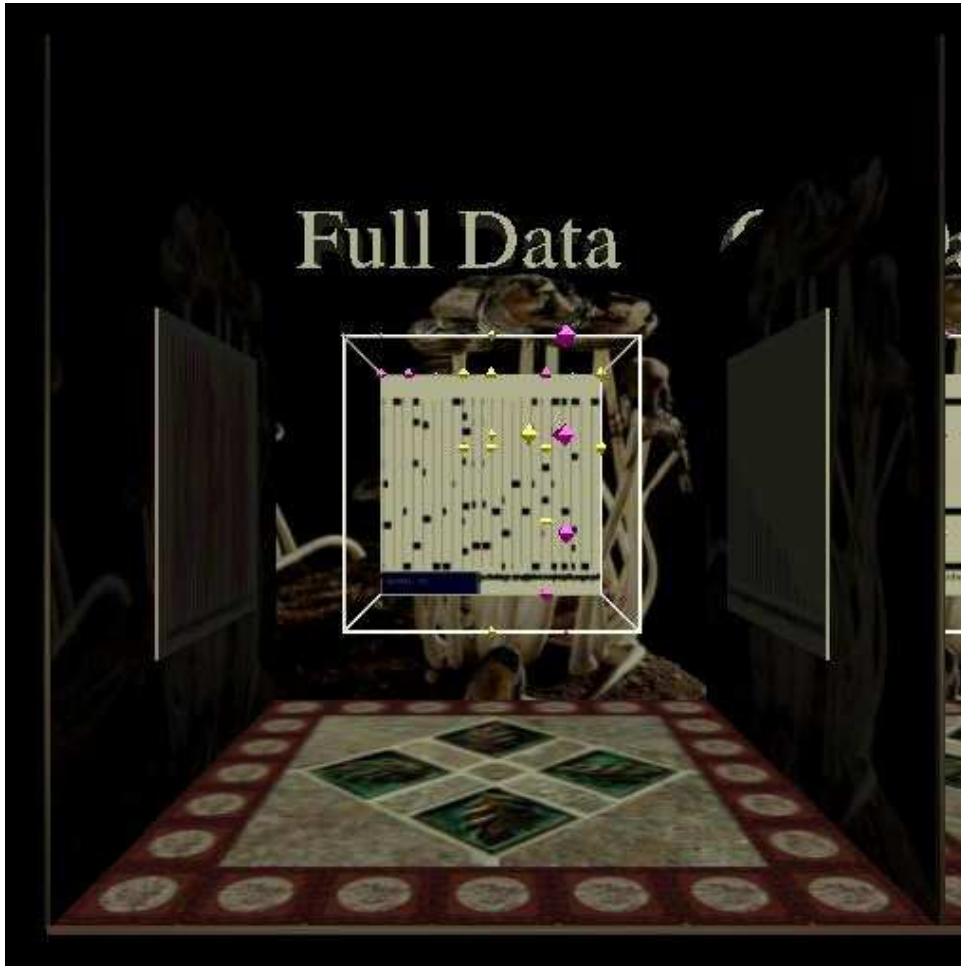


Figure 4.27: Mushroom dataset: first room, containing models reflecting the whole dataset.

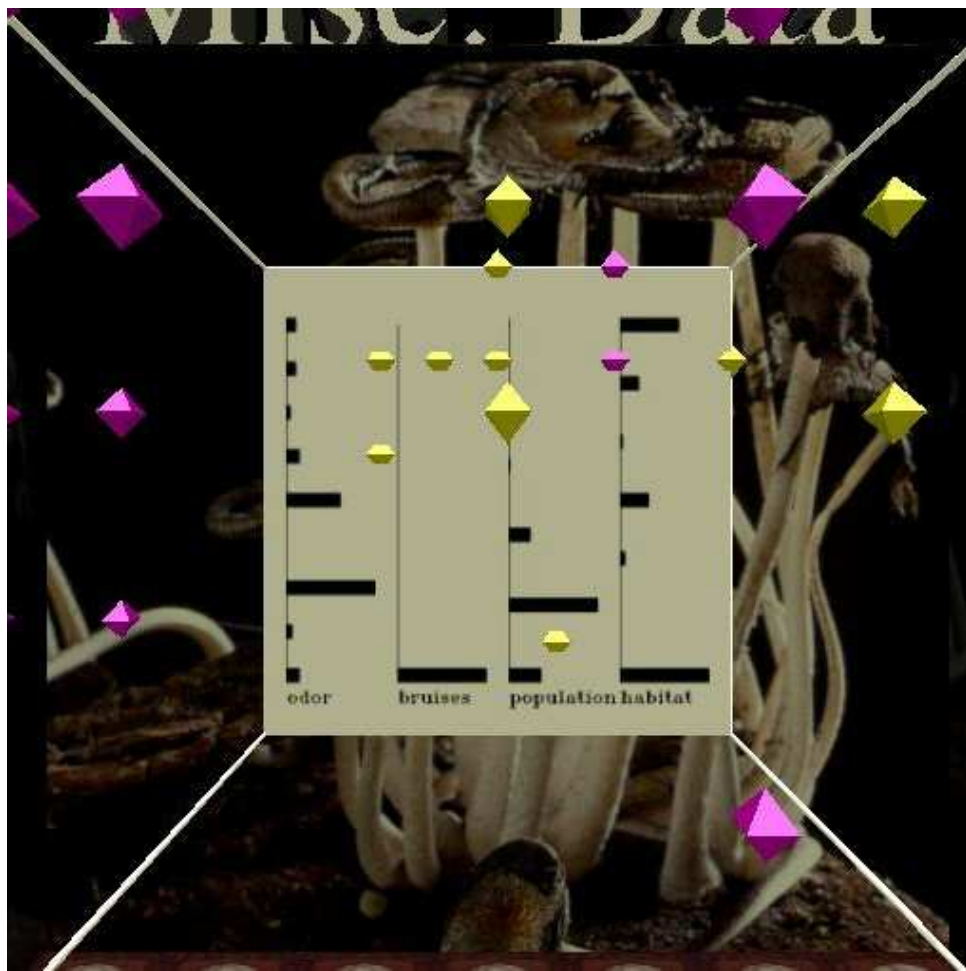


Figure 4.28: Mushroom dataset: room containing rankings and models comparing misc. attributes: odor, bruises, population and habitat; close-up on distribution image.

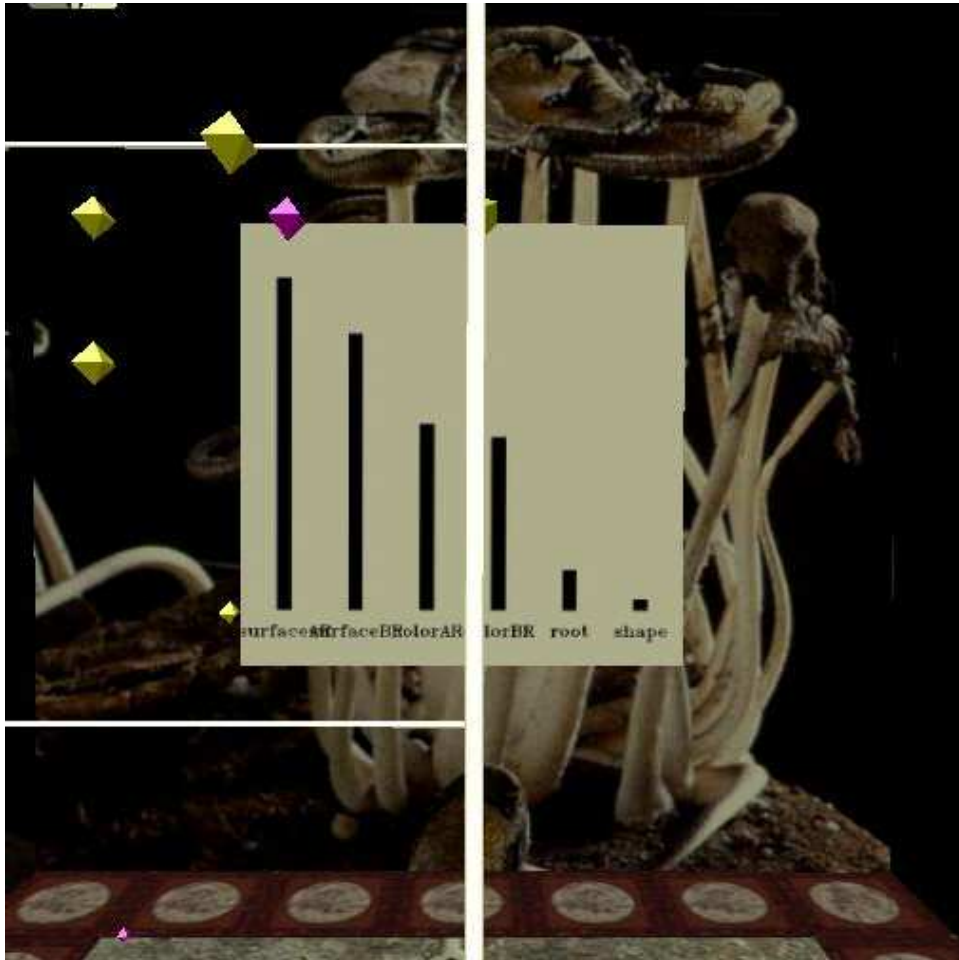


Figure 4.29: Mushroom dataset: stalk data room, attribute rank image.

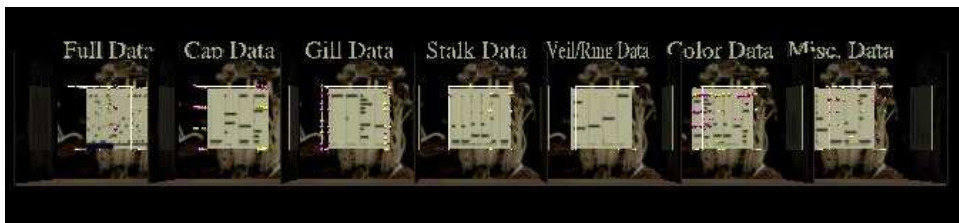


Figure 4.30: Mushroom dataset: whole museum at distance.

Chapter 5

Conclusions

5.1 Problems

A key issue is the limitations of the computer systems used to render the graphs. We managed to display up to 150,000 polygons at once without major slow-down. 150,000 polygons corresponds to roughly 15,000 octahedrons, plus labels and axes. With each octahedron representing a single point, only 15,000 points could be displayed at once, which is a serious limit in big data sets. By using tetrahedrons instead of octahedrons, we doubled this number, but this does very little to lessen the system requirements, especially when analyzing a single data set in numerous ways. While using a limited amount of the data was somewhat effective, it also limits the validity of the results. The lack of robust animation features is another limitation in dynamic data sets. Were animation features adopted, time-based data could be displayed much more easily, and user interactivity could become much more powerful and meaningful, allowing the user to move graphs on the fly within the virtual world. The inherent separation of the data visualizer and the experimenter in our research was very limiting in terms of the results we could develop. An excellent example of this was the liver data set, where we didn't understand much of our findings except in the most abstract sense. Ideally, in practical scientific application, the researcher and the visualizer would work together, with the visualizer developing less traditional theories and results based on the data and the researcher grounding him/her in previous research.

5.2 Effective Techniques

Several techniques were found to be especially effective means of conveying the maximum amount of information with the most clarity possible. Most simplistically, using the semi-circular museum layout lends itself very well to user exploration. The user is able to easily and quickly navigate the layout without getting lost, and is able to compare different graphs with each other. More significantly, the automated labeling techniques developed (gtb-axis1, and the additions to gtb-text) made the data far clearer, avoiding the issue of needing the visualizer to understand the data. They also enabled more complex graphs without sacrificing readability. Interactivity, via the wand in the RAVE, was by far one of the most useful developments. Using the ABSwitch function, it was possible to create "buttons" which turned graphs on and off. This allowed us to display far more graphs than we could have otherwise, greatly increasing the performance of the rendering. While this did not alleviate issues with running out of memory, it did make large series of graphs far more useable. Also, the development of a "radio button" feature in which buttons could be used to select only one graph to turn on made this even more effective, although we were unable to test it thoroughly.

5.3 Conclusions

One of the goals of this project was to investigate the effectiveness of 2D and 3D information visualization in an immersive environment. Though we have had success with selecting data sets and interpreting the graphs, it is still unclear how useful these types of data representation are. The extra dimensions of these graphs provide scientists with an easy way to represent data sets that have numerous attributes. Also, it is relatively clear to see trends due to the overall location of the data points and their size and color. However, due to our lack of a full understanding of the data that we worked with, it is not certain whether or not we gained any insight on these data sets that could not already be perceived by looking at conventional methods of data visualization (i.e. scatter plots, histograms, bar graphs, etc.).

Bibliography

- [1] C5, 2002. [online] <http://www.rulequest.com>.
- [2] Idl software, 2003. <http://www.rsinc.com/idl/>.
- [3] Open inventor, 2003. [online] <http://oss.sgi.com/projects/inventor/>.
- [4] Weka 3: Machine learning software in java, 2003. <http://www.cs.waikato.ac.nz/ml/weka/index.html>.
- [5] A glyph toolbox for immersive scientific visualization [online]. http://math.nist.gov/mcsd/savg/papers/NIST_SAVG_2002103000.pdf.
- [6] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998. <http://www.ics.uci.edu/~mlearn/MLRepository.html>.
- [7] J. Chambers, W. Cleveland, B. Kleiner, and P. Tukey. *Graphical Methods for Data Analysis*. Wadsworth, 1983.
- [8] R. Cheeseman, J. Kelley, M. Self, W. Taylor, and D. Freeman. Autoclass: A bayesian classification system. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 65–74, San Francisco, CA, 1988. Morgan Kaufman.
- [9] DIVERSE: Device independent virtual environments- reconfigurable, scalable, extensible [online]. <http://diverse.sourceforge.net/>.
- [10] John Hagedorn and Judith Devaney. A genetic programming system with a procedural program representation. In *2001 Genetic and Evolutionary*

Computation Conference Late Breaking Papers, pages 152–159, July 2001.
<http://math.nist.gov/mcsd/savg/papers>.

- [11] A. Inselberg. The plane with parallel coordinates. *The Visual Computer*, 1:69–91, 1985.
- [12] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kauffann, San Mateo, 1993.
- [13] James S. Sims, William L. George, Steven G. Satterfield, Howard K. Hung, John G. Hagedorn, Peter M. Ketcham, Terence J. Griffin, Stanley A. Hagstrom, Julien C. Franiatte, Garnett W. Bryant, W. Jaskolski, Nicos S. Martys, Charles E. Bouldin, Vernon Simmons, Olivier P. Nicolas, James A. Warren, Barbara A. am Ende, John E. Koontz, B. James Filla, Vital G. Pourprix, Stefanie R. Copley, Robert B. Bohn, Adele P. Peskin, Yolanda M. Parker, and Judith E. Devaney. Accelerating scientific discovery through computation and visualization ii. *Journal of Research of the National Institute of Standards and Technology*, 107(3):223–245, 2002. May-June issue, <http://math.nist.gov/mcsd/savg/papers>.
- [14] James S. Sims, John G. Hagedorn, Peter M. Ketcham, Steven G. Satterfield, Terence J. Griffin, William L. George, Howland A. Fowler, Barbara A. am Ende, Howard K. Hung, Robert B. Bohn, John E. Koontz, Nicos S. Martys, Charles E. Boulden, James A. Warren, David L. Feder, Charles W. Clark, B. James Filla, and Judith E. Devaney. Accelerating scientific discovery through computation and visualization. *Journal of Research of the National Institute of Standards and Technology*, 105(6):875–894, 2000. Nov.-Dec. issue, <http://math.nist.gov/mcsd/savg/papers>.
- [15] Fakespace Systems, 2003. <http://www.fakespace.com>.
- [16] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kauffann, San Mateo, 1999.