Dimensional Stacking in Three Dimensions

by

Timothy Walsh

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Computer Science

by

January 2008

APPROVED:

Professor Matthew Ward, Thesis Advisor

Professor Michael Gennert, Reader

Professor Michael Gennert, Head of Department

Abstract

Dimensional Stacking is a technique for displaying multivariate data in two dimensional screen space. This technique involves the discretization and recursive embedding of dimensions, each resulting N-dimensional bin occupying a unique position on the screen. This thesis describes the extension of this technique to a three dimensional projection. In addition to the visual enhancements, hashing was used to improve the scalability of records and dimensions. The resulting visualization was evaluated by a usability study.

Acknowledgements

I would like to thank my adviser, Matt Ward, for his guidance and patience. The hours spent walking and listening helped my thesis become a success.

I would also like to thank my reader, Mike Gennert, for reading this and providing a new perspective.

I would also like to thank my friends and family, for without their support I would not have completed my thesis. I would especially like to thank my parents for their love and support over the past 23 years.

Lastly, I thank my fiance and biggest supporter, Amanda Jamin. Her guidance, patience and gentle nagging drove me to complete this phase of my life. Thanks you for everything.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Data with many variables, or dimensions, is called multivariate data. This type of data can be found in a wide variety of fields, including government, economics, mathematical sciences and medicine. Understanding this type of data can be very difficult without help.

Information visualization techniques can be used to take multivariate data and produce a two or three dimensional picture. This picture is a visual representation of each record in the data set. Because humans can find relationships in a picture more easily than rows of numbers, data visualization helps researchers and analysts to understand and communicate their findings.

Basic data visualization techniques include scatterplots, bar charts and line graphs. These techniques work well for data with one or two variables, or a few discrete variables with one value. When data does not fit this profile, more advanced techniques that handle multivariate data are needed.

Dimensional Stacking [12, 25] is one advanced technique that can handle data with many dimensions. This works by breaking the x and y axes by a number of divisions. The number of divisions the current axis is divided into is determined by the cardinality of a dimension.

Name	Min	Max	Cardinality
Graduates	0	10	4
Professors	0	20	4
TA's	0	10	4
RA's	0	10	4

Table 1.1: Dimensional information

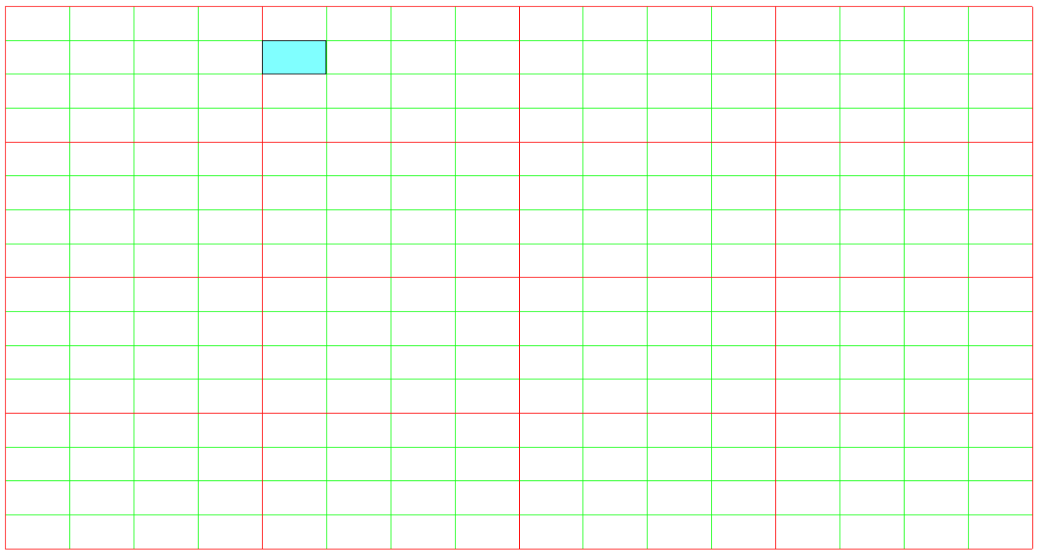


Figure 1.1: Stacking of dimensions from Table 1.1

The current axis alternates between the x-axis and the y-axis, producing a nested grid of rectangles (see Figure 1.1). These rectangles, or bins, represent a range of values for each dimension. Using the information in Table 1.1, it is possible to find the ranges. The highlighted bin in Figure 1.1 represents the ranges:

- 2.5 – 5.0 for Graduates (outer, horizontal)
- 15.0 – 20.0 for Professors (outer, vertical)
- 0.0 – 2.5 for TA's (inner, horizontal)
- 5.0 – 7.5 for RA's (inner, vertical)

For the bin to be filled there must be some record in the data set with values in the ranges of all dimensions.

However this technique is not very scalable due to the limited rendering area available. Currently the largest number of dimensions that can be understandably displayed is twenty, each containing two bins, or six dimensions each with ten bins, which would map each bin to a single pixel. Larger datasets would produce a visualization where each bin is represented by a portion of a pixel, which clearly is not likely to be useful.

Also the current implementation of dimensional stacking uses an array to internally represent the space. This is wasteful for sparse datasets since much of the array would be empty. It is also impractical for searching dense datasets since there is no intrinsic ordering of where points could lay. For data sets that have m dimensions, each with a cardinality n, the array needed to store the values for every bin would contain m^n elements. For data sets with a moderate number of dimensions (e.g. twenty) and modest number of bins (e.g. ten) this would require an array of 10^{20} elements, most of which could be empty.

Improving upon these limitations of dimensional stacking is the aim of this thesis. Although the resolution of the screen can be changed to increase the size of the rendering area, this is not a solution. Since changing the size of the rendering area is not possible, it must be used more efficiently by finding a method to allow a single pixel to represent more then an individual bin.

The rest of this paper discusses the design and implementation of the newest version of dimensional stacking called NLand. Chapter 2 gives an overview of other multivariate data visualization techniques. The design and implementation of NLand is discussed in Chapter 3. In Chapter 4 the results of the evaluation of this application are presented. Lastly, Chapter 5 presents conclusions and additional work found during the evaluation.

Chapter 2

Related Work

Several techniques exist for displaying multivariate data, including Scatterplot Matrices, Parallel Coordinates, Pixel-Oriented visualizations, and Glyphs.

2.1 Other Visualizations

2.1.1 Scatterplot Matrices

Many people are familiar with scatterplots. These simple plots are used to compare two dimensional data by plotting points on an xy-Cartesian plane. Three dimensional can be compared using a three dimensional scatterplot which would use the xyz-Cartesian space instead. For data that has more than three dimensions, these plots must be expanded to a matrix.

A scatterplot matrix is an n x n matrix that has all the rows and columns

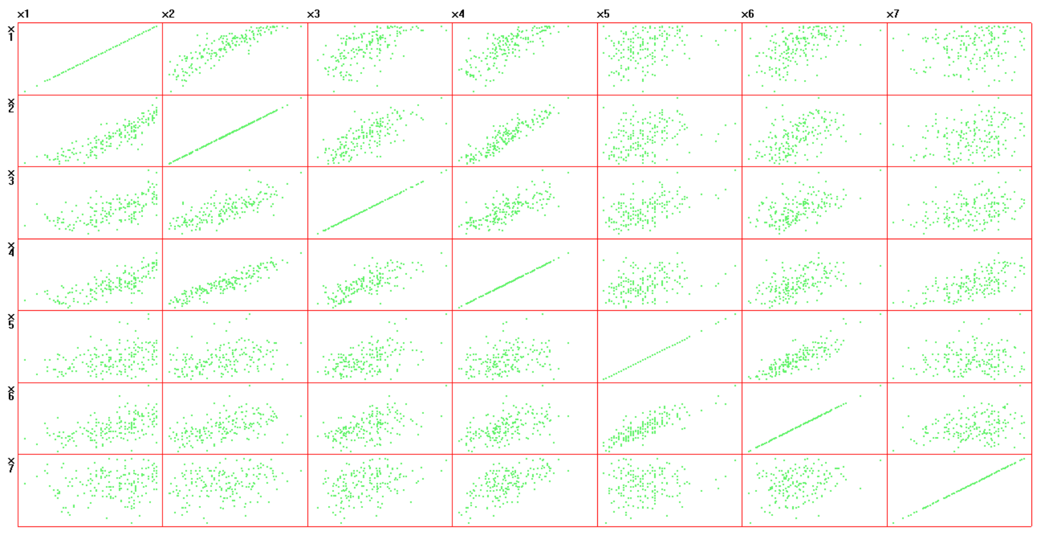


Figure 2.1: Scatterplot matrix of 7 dimensions

labeled by the n dimensions (see Figure 2.1). Each cell (i, j) in the matrix is a scatterplot with the i^{th} dimension on the y-axis and the j^{th} dimension on the x-axis. Because this matrix has all n dimensions on the rows and columns, the matrix is symmetric across the diagonal. This means that the cell (j, i) is the same scatterplot as (i, j) except which of the two axes the dimensions are on [3].

Scatterplot matrices work well for comparing a large number of records and dimensions. However, these matrices only provides information about how two dimensions relate. Comparing three dimensions requires an understanding about how all three dimensions relate to the other two dimensions.

2.1.2 Parallel Coordinates

Parallel Coordinates creates a new coordinate system to represent n-dimensional objects [1]. This is done by placing each of the n-dimensions parallel to the y-axis, or the x-axis for horizontal axes, and evenly spacing them along the x-axis (see Figure 2.2), or the y-axis if using horizontal axes. This creates a new coordinate system in the xy-plane that has n axes perpendicular to the x-axis. Each axis x_i is a dimension in the n-dimensional space.

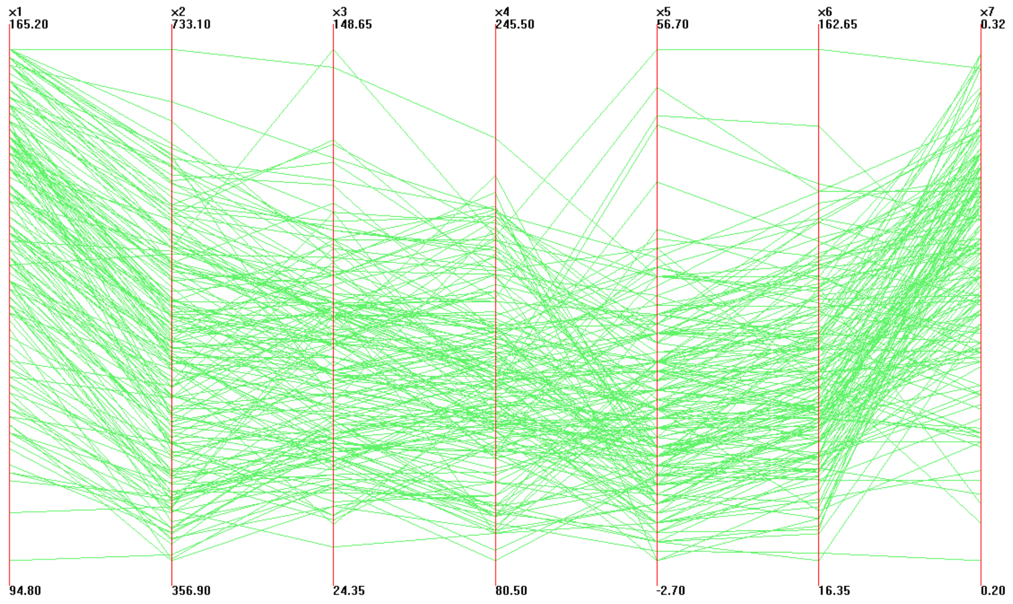


Figure 2.2: Parallel Coordinates of 7 dimensions with 200 records

In [7], Inselberg and Dimsdale look at the dimensions $x_i, x_{i+1} \in R^n$ with a distance of d between x_i and x_{i+1} (see Figure 2.3). The line $x_{i+1} = mx_i + b, m < \infty$ is a series of points, A, which are represented by the collection of lines $\overline{A}$ in parallel coordinates. When $m \neq 1$, the lines in $\overline{A}$ intersect at the *point* $(d \div (1-m), b \div (1-m))$ with respect to the xy-Cartesian coordinates. Conversely, in the $x_i x_{i+1}$-projective plane the *ideal point* with slope m is mapped into the vertical line at $x = d \div (1-m)$ of the xy-projective plane. This result gives a duality between points in n-dimensional space and polygonal lines in parallel coordinates.

This duality means that a point C, with coordinates $(c_1, c_2, c_3, ..., c_n)$, is represented by a polygonal line with n vertices at $(i-1, c_i)$ on the x_i-axis. Because each axis in parallel coordinates represents a 1D projection of the data set, clustering is easily seen [4]. Separation along an axis shows a view of isolated clusters.

Parallel Coordinates handles data sets with several dimensions and a moderate number of data records well. However, this technique becomes crowded and difficult to understand when the data set has hundreds of dimensions or a large number of data records.

2.1.3 Pixel-Oriented

Pixel-Oriented visualizations represent each record in the data set by a pixel. Due to the limited rendering area, this type of visualization has a limit on the size of the data set being visualized. Because of this limitation, these visualizations attempt to use the maximum number of pixels while still presenting an understandable picture.

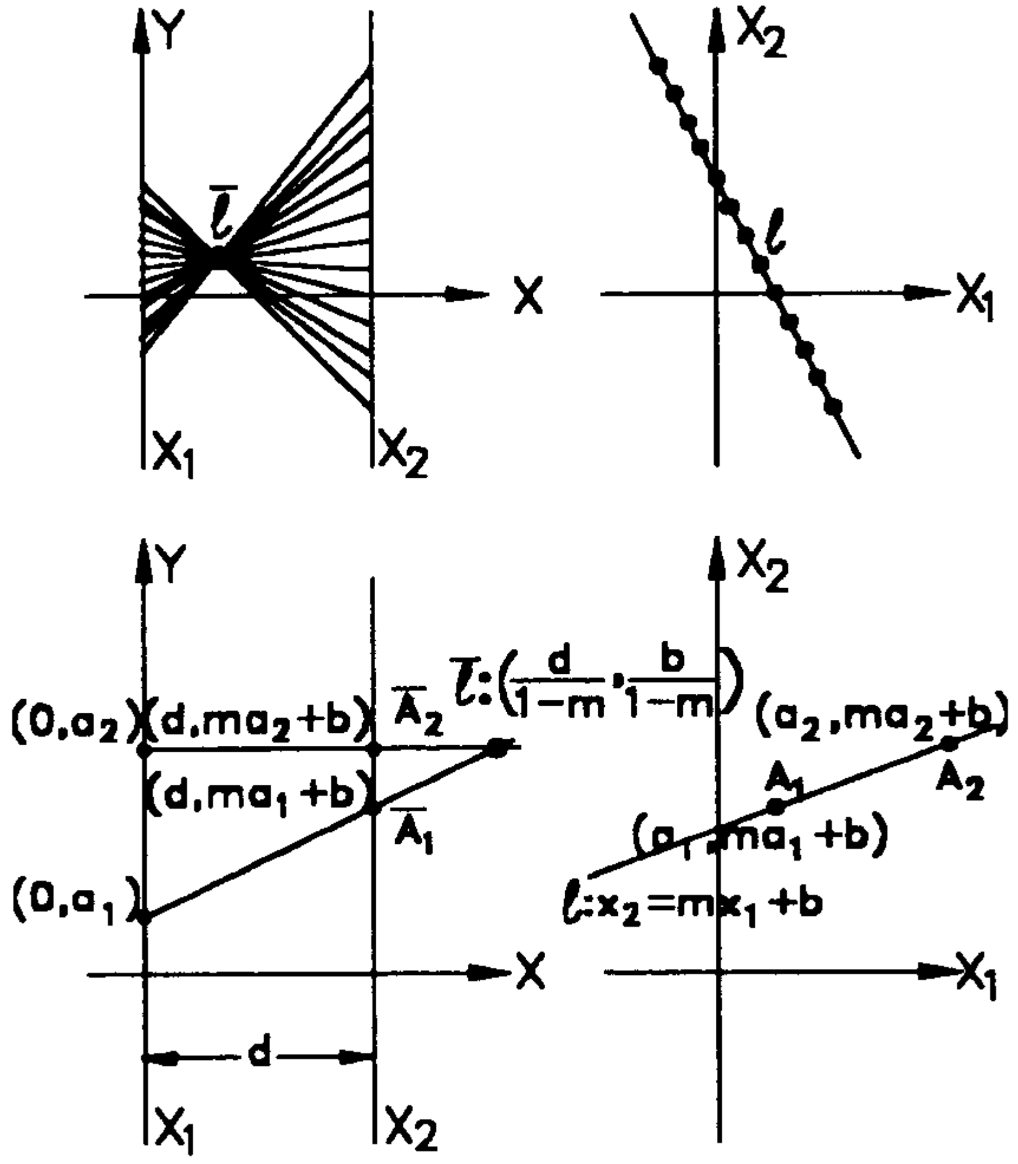


Figure 2.3: Point-line duality from [7]. Used without permission

Screen-Filling Curve Techniques

Screen-filling visualizations are based on the space-filling curves of Peano [18] and Hilbert [6]. By creating a continuous curve that passes through every point of a regular region of space they are able to maximize the number of data points that can be shown. These were primarily used to study recursion and produce pictures, although researchers have begun to use clustering properties of these curves to index spatial databases [2]. Space-filling curves map both dimensions into one dimension, which optimizes storage and the processing of two-dimensional data. This mapping preserves the spatial locality of the original two-dimensional image.

Visualizing multivariate data that has been sorted by one variable has the problem of mapping a one-dimensional distribution of data onto the two dimensions of the screen. Space-filling curves handle this issue also since data that are close together in the one-dimensional ordering are likely to be close together in the two-dimensional representation [9, 8, 11]. This may allow for discovery of patterns in the data that may not be noticeable.

The Peano and Hilbert curves are both recursive visualizations that fill squares of size $(2^i, 2^i)$ for $i = 0$ ... max-level. A pattern of size $(2^i, 2^i)$ will have four subpatterns each with size $(2^{i-1}, 2^{i-1})$, and the orientation of these subpatterns will change.

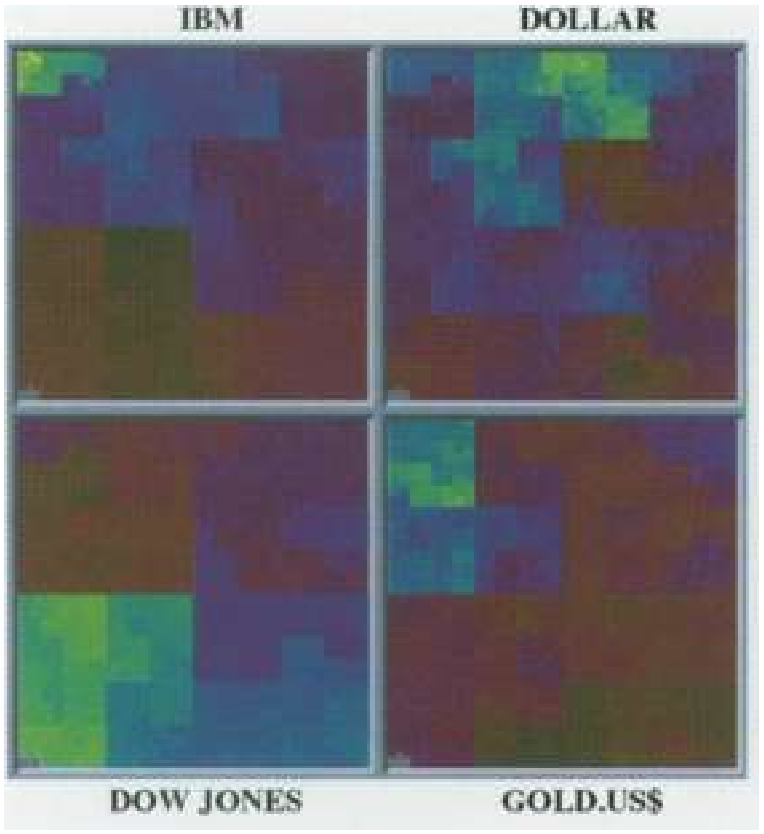


Figure 2.4: Peano-Hilbert Curve of stock information from [9]. Used without permission

Recursive Pattern

The Recursive Pattern [10, 9] technique uses a back-and-forth arrangement to recursively display data records. This technique draws a number of points on three lines. The lines are first drawn from left to right then right to left and lastly left to right. All the points in the S shaped object contains another level of the recursion.

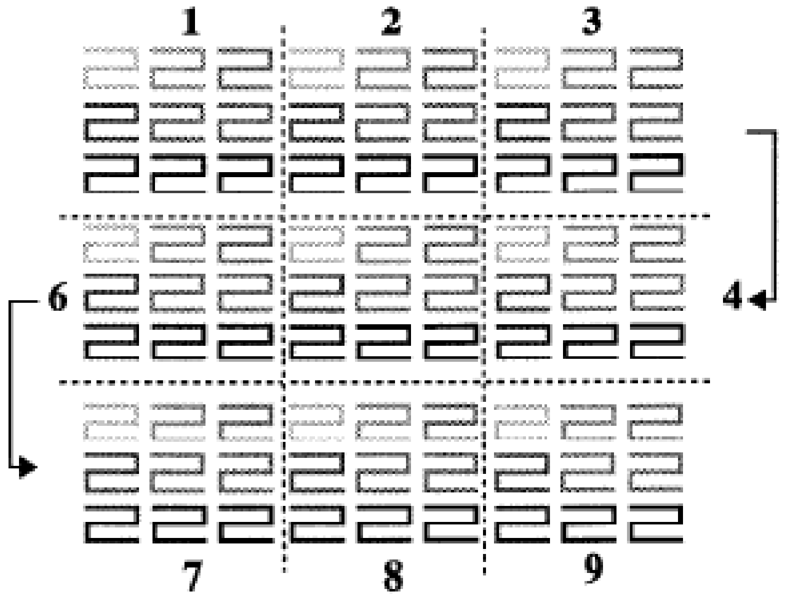


Figure 2.5: Recursive Pattern of 3-Dimensions from [10]. Used without permission

For example, Figure 2.5 shows three levels of recursion in a fully recursive arrangement. The highest level is shown by the numbered positions. Each of the numbered positions represents the next level of recursion which is made of a series of S objects that change from a light gray to black. All the S objects in this level are actually the lowest level of the recursion.

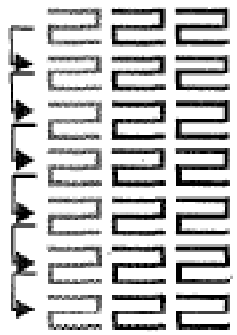


Figure 2.6: Line-by-line loop from [10]. Used without permission

This visualization allows the user to control the number of points each line contains. This is done by changing the width and height of different levels to produce a new shape. In Figure 2.5 level i has $(w_i, h_i) = (3, 3)$ for $i = 1...3$. In contrast Figure 2.6 shows the same dataset but has $(w_1, h_1) = (3, 3)$, $(w_2, h_2) = (3, 1)$ and $(w_3, h_3) = (1, 7)$.

For some types of data, such as time-series data, varying the width and height to certain numbers may give better results. If the data has been collected a few times a day over a period of a few weeks it may be beneficial to set (w_1, h_1) to $(3, 3)$ and (w_2, h_2) to $(3, 7)$. Level 1 would then represent a day and level 2 would represent a week. If the collection is continued over a few months then the user could expand the second level or add new levels to represent weeks and months.

2.1.4 Glyphs

Glyphs are icons that represent one record of the data set. Each dimension is mapped to one feature and the value determines some aspect of the feature. Two commonly used glyphs are Chernoff Faces [5] and Star Glyphs [22]. Like Pixel-Oriented visualizations, glyphs suffer from a lack of space to display all the records.

Chernoff Faces

This form of glyph uses the human's ability to recognize small differences in faces to create a powerful visualization. Each dimension is mapped to a part of the face, such as the nose or the eyes, and the shape or size is determined by the value of that dimension. Records that are similar would look the same, thus allowing for a simple and quick way to recognize clusters. It is also argued that analysis is done in parallel, which facilitates the efficient recognition of patterns [13].

Star Glyphs

Star Glyphs are based on whisker plots which have a central point and n lines, or whiskers, emanating from the central point. Each line represents one dimension of the data set whose value determines the length. The difference between whisker plots and star glyphs is the ends of adjacent lines are connected to each other [13]. Recognizing clusters is simple with star glyphs since they will all have similar shapes.

2.2 Comparison of 2D and 3D Visualizations

A few comparisons of two and three dimensional visualizations, such as [26], [27], [15] and [23], exist. Springmayer et al. [24] showed that two-dimensional visualizations are used to find precise relationships while three-dimensional visualizations are used for qualitative understanding and presenting ideas to others.

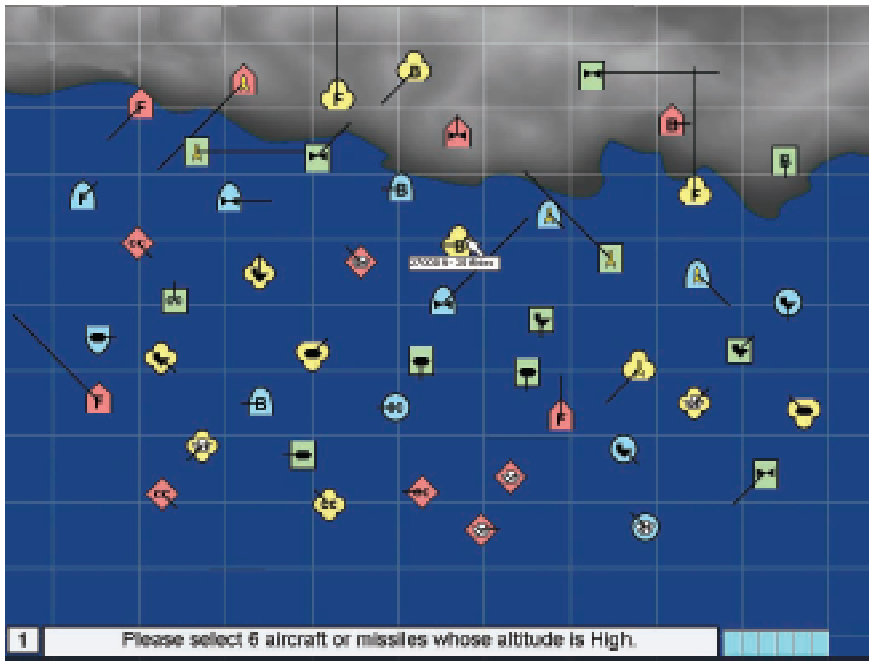

1
Please select 5 aircraft or missiles whose altitude is High.

Figure 2.7: 2D display of aircraft and ships in a space from [23]. Used without permission.

Some studies have compared the conventional two-dimensional displays used in flight control of airports and military bases to three-dimensional displays [23]. The three-dimensional version (see Figure 2.2) displayed models of aircraft and ships, while they were represented by icons in the two-dimensional version (see Figure 2.2). Altitude and pitch were provided in an analog and digital manner. The analog manner showed a white line from each icon and model to the land underneath. When an icon or model is selected, the altitude and pitch is shown to the user for the selected item.

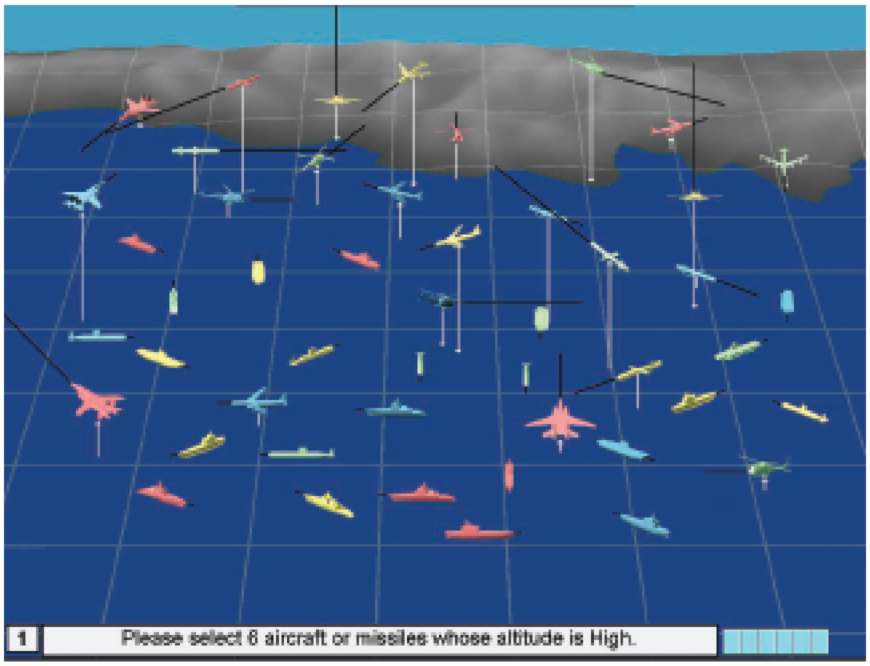

1
Please select 6 aircraft or missiles whose altitude is High.

Figure 2.8: 3D display of aircraft and ships in a space from [23]. Used without permission.

The comparisons found that two-dimensional visualizations perform better than three-dimensional visualizations in tasks such as flight management. There was some ambiguity in the analog pitch that prevented users from knowing the pitch. These studies also found that a constant display of the third dimension provided enough information for users to complete tasks quicker than without the information.

Chapter 3

Methodology

The goals of this thesis are to:

- Improve memory usage for dimensional stacking
- Experiment with extending dimensional stacking to three dimensions to improve understanding and scalability

3.1 Improving Memory

To accomplish the goal of improving memory usage, an efficient method of storing information about each bin was needed. Each bin has a unique string of numbers that determines where in the stacking it belongs. For example, Figure 3.1 shows four dimensions and a single bin. This bin is located at $< 2, 4, 1, 3 >$ which represents the second division of the x-axis (second red column), fourth division of the y-axis (fourth red row), first subdivision (first green column in the second red column), and the third subdivision (third green row in the fourth red row). Because we want to store information about each bin, this string of numbers makes an ideal identifier for a bin. Having a unique identifier for every bin in the visualization suggests an efficient method of storing information about each bin is to utilize a hashing function [28].

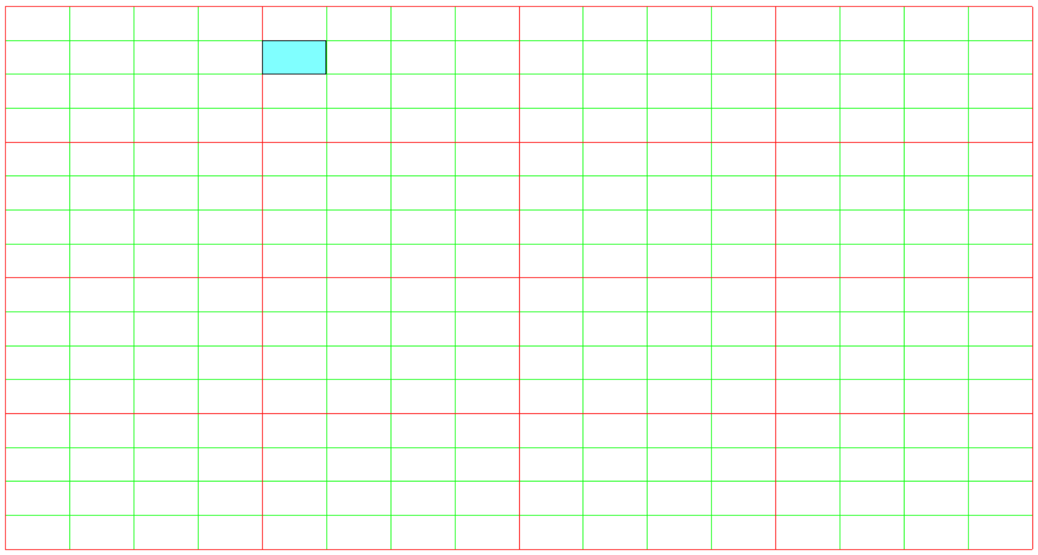


Figure 3.1: Highlighted bin at $< 2, 4, 1, 3 >$

The purpose of the hashing function $h(x)$, is to convert a key x to an index of an array. This function must be able to produce every value between 0 and $t - 1$, the size of the array. Commonly the value of t is a prime number, which practice has shown to work better [17]. For $h(x)$ to remain in the $0..t$ range the function needs to use the modulo operator, making $h(x) = x\%t$.

Keys to the hashing function must also be unique. Since the application

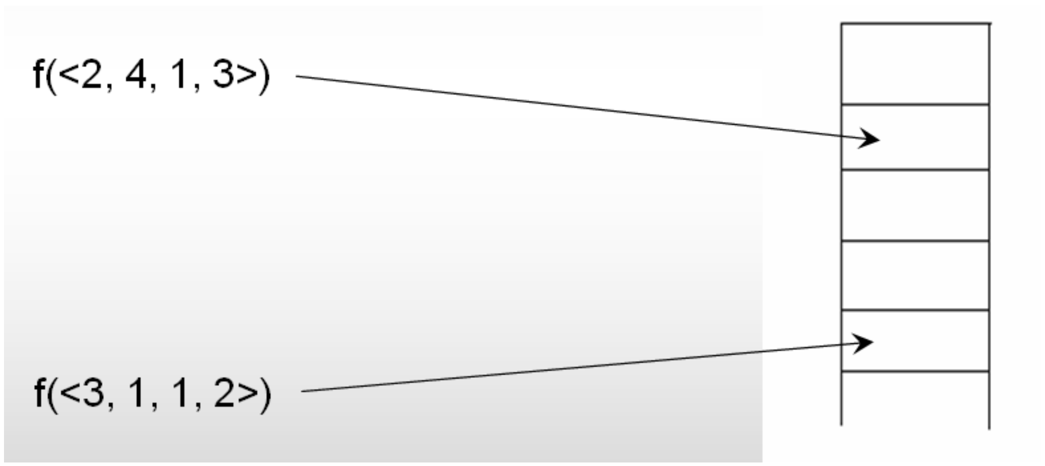


Figure 3.2: Hashing of two bins

stores information about each bin, a unique key would be the string of numbers that represents the location in the data cube. There are two methods of hashing strings: reducing the string to a string of bits and converting to a single integer number [20] [21].

Converting the string of numbers to a single integer and performing a modulus operation was the method chosen to convert each string. This allows for the use of a simple formula, $h(x) = \sum_{i=0}^{n-1} s_i * 2^i$, that takes each number s_i and converts it to an intermediate value. To reduce the number of collisions, a slightly different function can be used. This function replaces $s_i * 2^i$ with $(s_i * 2^i \% t)$ producing the hashing function $h(x) = (\sum_{i=0}^{n-1}(s_i * 2^i \% t))\%t$.

Figure 3.2 shows two bins being mapped to locations in the array. Using the string $< 2, 4, 1, 3 >$, making $t = 11$, as input to $h(x)$ produces the following results:

$$h(< 2, 4, 1, 3 >) = ((2 * 1)\%11 + (4 * 2)\%11 + (1 * 4)\%11 + (3 * 8)\%11)\%11$$

$$h(< 2, 4, 1, 3 >) = (2 + 8 + 4 + 2)\%11)$$

$$h(< 2, 4, 1, 3 >) = 5$$

Now performing the same computation on the string $< 3, 1, 1, 2 >$ gives the following index:

$$h(< 3, 1, 1, 2 >) = ((3 * 1)\%11 + (1 * 2)\%11 + (1 * 4)\%11 + (2 * 8)\%11)\%11$$

$$h(< 3, 1, 1, 2 >) = (3 + 2 + 4 + 5)\%11)$$

$$h(< 3, 1, 1, 2 >) = 3$$

From this example it is obvious that the size of the array used must be large to avoid collisions. The hash table size, t, must be at least as large as the number of items being stored. When the load factor, the ratio of the number of occupied locations to the size, increases above a certain threshold the size of the table is increased to improve performance. By reducing the load factor, the chance of a collision is reduced also.

Increasing the total size of the hash table takes is a costly event, since every record must be updated to a potentially new location. To avoid performing this operation often, a simple method is used to find the new size, doubling the previous size. This however now produces a a non-prime size which is not as efficient as a prime number. If the properties of a prime number are desired, using the next prime could be used but at the cost of potentially using more space than needed.

Though a hash table has the potential to use more memory than an array, the average data set will not require the use of every bin. In fact, dense data sets has a higher potential for records to belong to the same bin. Because the hash table does not store every record this allows for a smaller table. In this case the hash table will require less memory than an array.

Another advantage is the improvement over finding information about a record. Searching for a match in an array requires O($logn$) operations while finding a match in a hash table requires O(1) operations. This computational savings offsets the unused space in the hash table.

3.2 Extending to 3D

The traditional view of dimensional stacking is a two dimensional mapping of data values to rectangles. Extending this technique to three dimensions creates a mapping of data values to rectangular volumes that may occlude each other. To handle this problem, volume rendering techniques were applied.

Several techniques exist for performing volume rendering [14] [16] [19]. One class of techniques, called ray-casting [16], produces a two dimensional representation of the volume. The general idea of these techniques is to cast a series of rays from the virtual camera through every pixel in the view. When a ray hits an object, it sets the opacity based on the time the hit happened. Sampling each pixel is necessary for determining the color of a pixel because it may be not filled, partially filled or completely filled. This sampling of pixels increases the number of rays needed to draw the representation.

These techniques produce a true representation and allows for a fine degree of control over rotation and zooming. However, it takes several seconds to produce this representation. This fact prevents its use in an interactive application. Some techniques for speeding up the rendering exist, such as fast voxel approximation [14], but produce a blocky representation. Because an accurate representation is more important than an interactive program, this increase in speed is insufficient.

The technique used to render the rectangular volumes was to create a transparency value based upon the distance from the camera to the center of the volume. These transparency values ranged between zero, or completely transparent, to one, completely opaque. This means the closer to the camera, the more transparent the rectangular volume will be, and the further from the camera, the more opaque it will be. This prevents volumes close to the camera from occluding volumes behind it.

To properly analyze the data a user needs to be able to move about and zoom into a portion of the data cube. Techniques for interacting with object are broken into two categories: direct and indirect. Direct interaction is accomplished by manipulating the cube with the mouse, while indirect would manipulate the cube through a separate control. Neither direct or indirect manipulation performs better than the other, but many people prefer one over the other.

This application uses indirect controls for allowing the user to interact with the application. Figure 3.3 shows the control panel from NLand. The four controls in the roll-out labeled Translations allow the user to rotate the data cube, move the camera down each axis, and reset the view to the standard one. Zooming is accomplished by moving the camera closer to the origin.

On the side of the main visualization are three standard views of the data cube. Figure 3.4 shows the xy face (top), yz face (middle), and the zx face (bottom). These cube faces provide information about relationships between the dimensions on the two axes in that face. Allowing the user three points of reference helps in locating areas of interest. These side views also allow the user to recenter the camera when clicked on.

In an effort to support datasets where values map to the same bin, each volume is colored corresponding to the number of data values in that bin. This allows the user to know the density of a volume and provides more information to the user. Since users will not know what the different colors mean, a legend (see Figure 3.5) is provided. This legend also shows which dimensions are mapped to an axis. The dimension names are color-coded to the axis it is dividing and indented to show what it is subdividing.

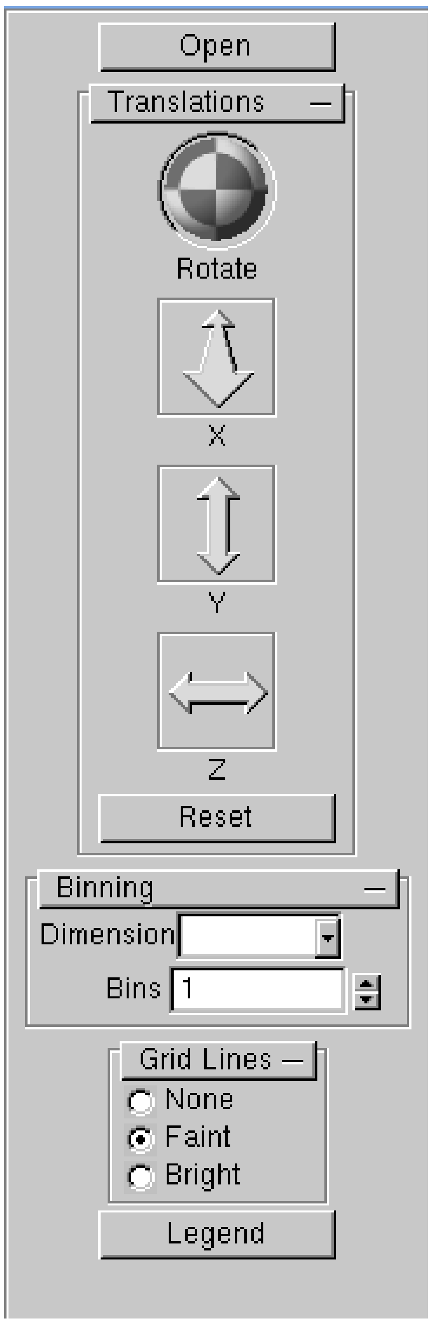


Figure 3.3: Control panel from NLand

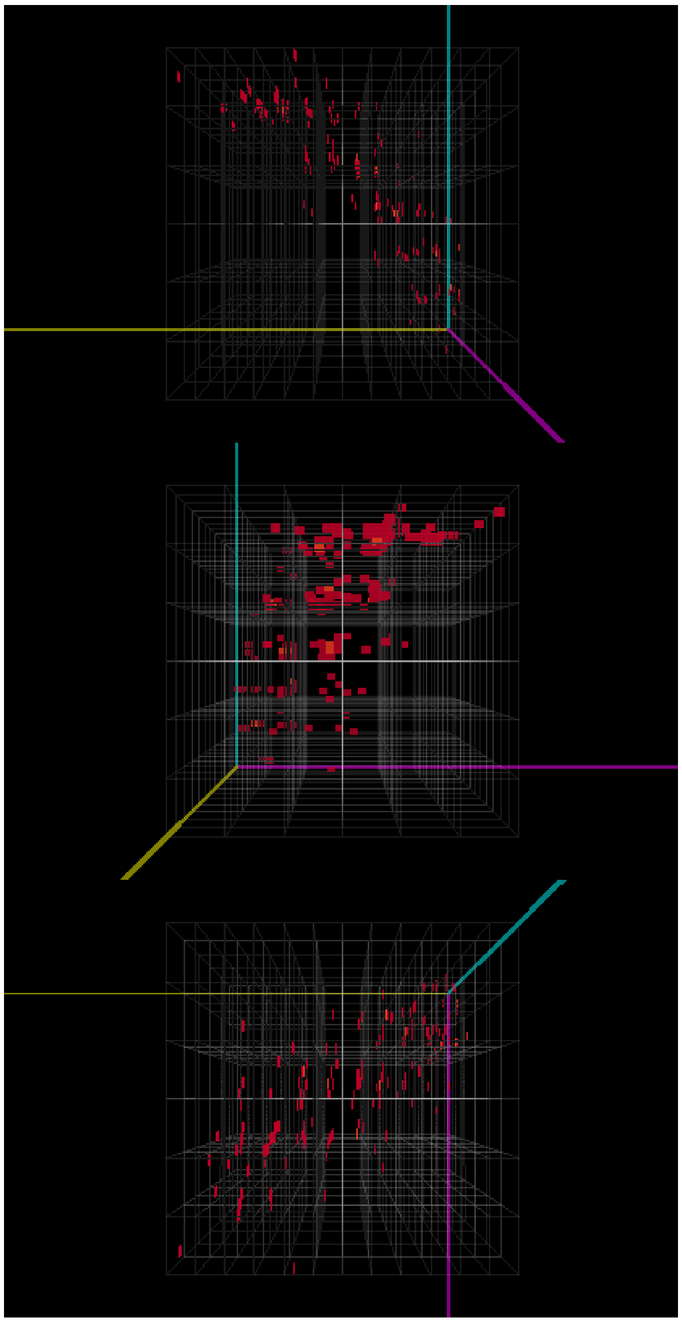


Figure 3.4: Three static views along the x, y and z axes

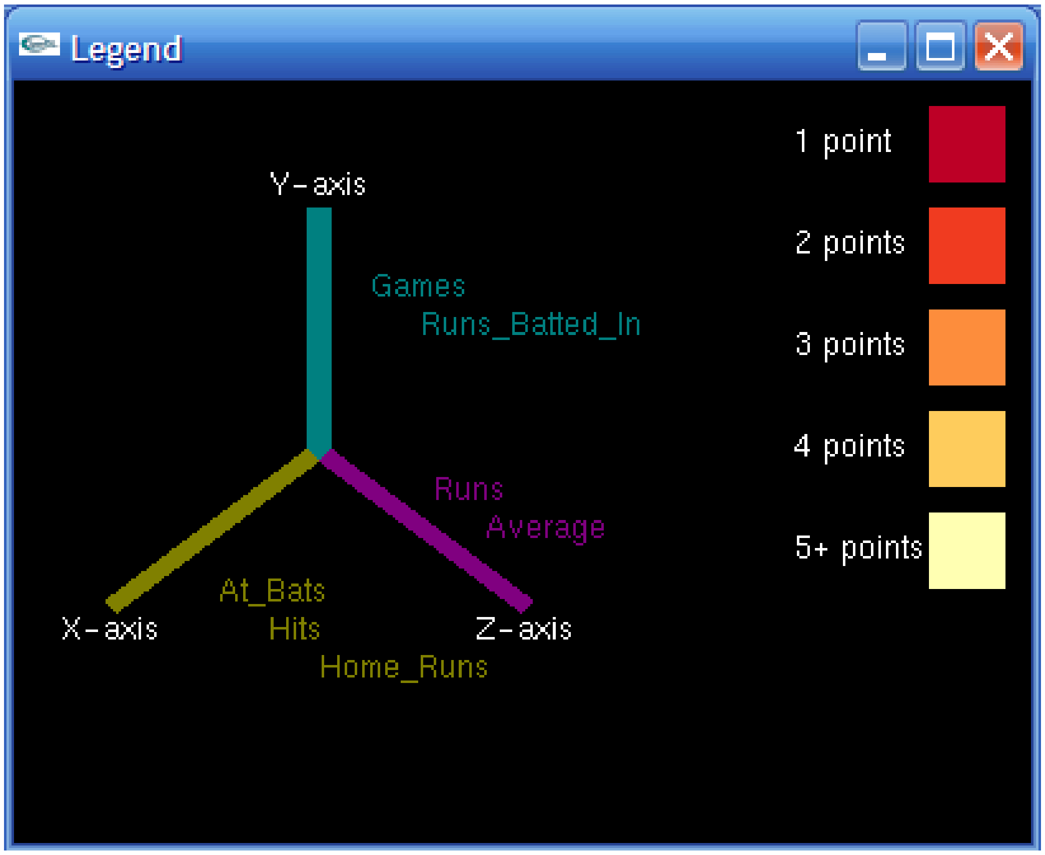
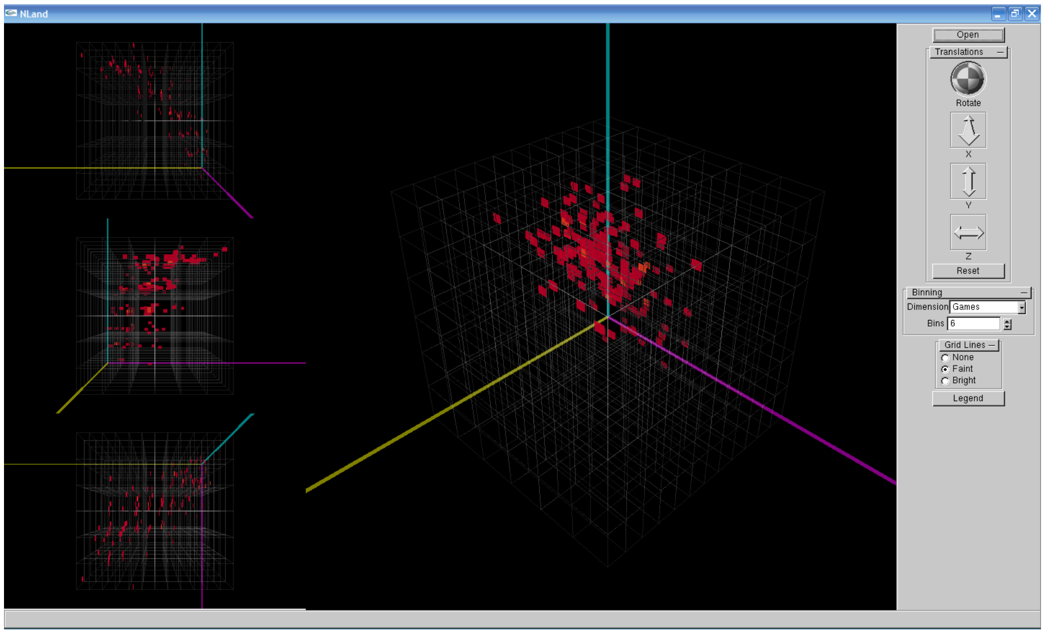


Figure 3.5: Legend provided explaining colors and mappings



Dimension: Games

Bins: 6

Grid Lines: None [✓ Faint] Bright

Figure 3.6: NLand with example dataset

Chapter 4

Results

4.1 Usability Study

In order to determine if the three dimensional version of dimensional stacking has any benefits over the current two dimensional version, a comparative study was performed. This was done by giving users a set of tasks to complete in both versions. Comparing the accuracy and speed of each user between the tasks provides information about any benefits of the visualization.

Prior to being given the tasks, users were given a tutorial with both XmdvTool and NLand (see Appendix A.1). This tutorial covered how each axis is divided, what a bin is, and how data is mapped to a bin. Users were also provided an opportunity to use XmdvTool and NLand and to ask questions.

To limit the bias introduced by a learning curve, users saw different versions first. However the tasks that users were asked to perform were kept in a fixed ordering. The three tasks (see Appendices A.2 and A.3) were ordered based on difficulty. This works with the learning curve to reduce frustration and helps the user learn to interpret the visualization.

The data set used was the top two hundred baseball players ranked by the number of at bats during the 2007 baseball season. There was seven dimensions to the data:

- Number of games played in
- Number of at bats
- Number of runs scored
- Number of hits
- Number of home runs
- Number of runs batted in
- Batting average

The range for each dimension was determined by taking the minimum and maximum of each dimension. This spreads each record more evenly across the visualization instead of bunching them together in one corner.

The first question looked for a relationship between dimensions on the major axes of both visualizations. The other two tasks asked the user to find a player: one of the best players and one of the worst. Because the best player had similar statistics as one other player, this question came second.

To help users who were unfamiliar with baseball, some clues were provided in each task's question. The second task asked the user to find the player that *leads* the league in home runs and runs batted in and also has the *most* runs. Though these clues are subtle, they help users gain a sense of where to look in the visualization.

After each user finished the tasks, they were given a questionnaire (see Appendix **??**) about the three dimensional version. This questionnaire has ten questions that ask the user to circle their ability on a one (easy) to seven (hard) ordinal scale. Following these questions are some preference and suggestion questions to find areas of improvement.

Thirteen users participated in the usability study. Of this thirteen, five was female and eight were male. One user was an expert in data visualization, eight users had some experience and four had no experience. Six users had received a bachelor degree, six other users had a master degree, and the last user had completed high school. The median age of the users was twenty six with a mean of twenty nine. One user was color challenged.

4.2 Study Results

Looking at the times to complete each task gives evidence about places where NLand has made an improvement over XmdvTool. Figure 4.1 shows a scatterplot of the times to complete the first task, shown in Table B.1 (see Appendix B.1). This plot shows no correlation between a user's ability to see a relationship in either program. However, in general, it took less time to see the positive linear relationship between games and at bats in the three dimensional version.

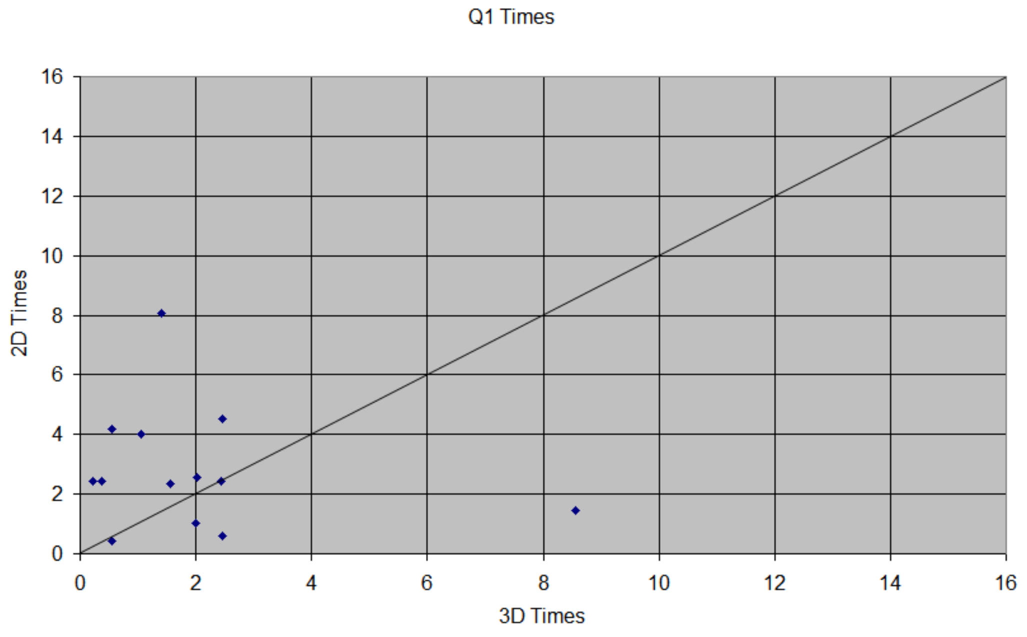


Figure 4.1: Scatterplot of Task 1 times (in minutes)

Except for one user, all users were able to see a positive relationship between the number of games played and the number of at bats over the season. The fastest users used their previous knowledge of baseball to answer this task, which may have skewed the data.

Table B.2 (see Appendix B.1) shows the times to complete the second task. The plot of these times is found in Figure 4.2, which shows a few outliers in both the two and three dimensional times. Ignoring these it is easy to see two small clusters of times at $(2.00, 2.00)$ and $(4.00, 5.00)$. These clusters are close to the line $y = x$ which indicates that these clusters have users with similar times for completing each task. The cluster at $(4.00, 5.00)$ is above the line which shows that the three dimensional task took slightly less time to complete.

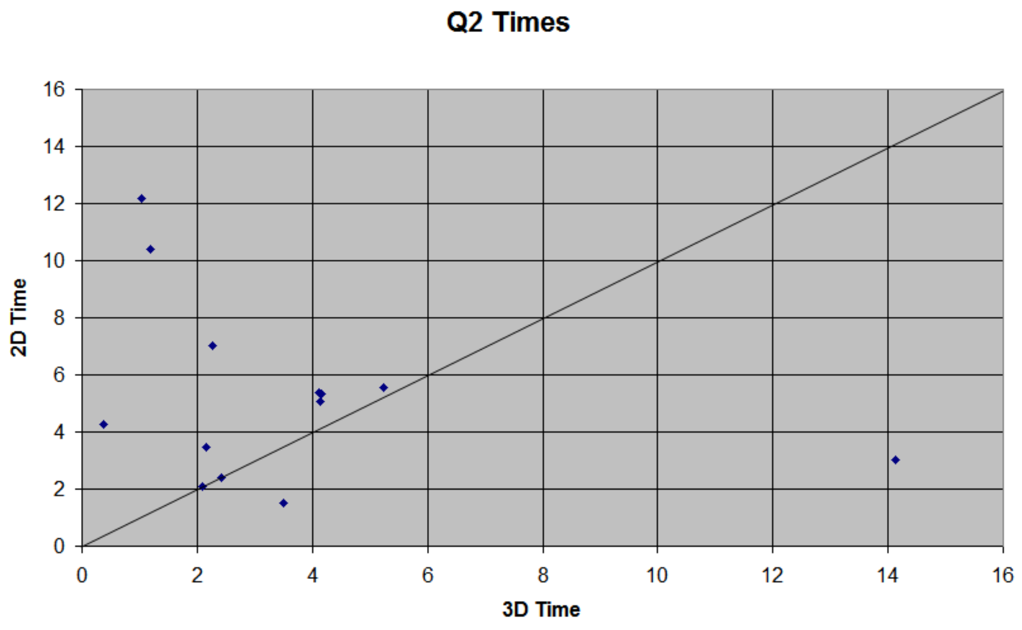


Figure 4.2: Scatterplot of Task 2 times (in minutes)

Three users were able to correctly locate the bin containing the described player in the three dimensional version. However, eight users were able to limit the potential bins to two bins but chose the wrong one. The other two users randomly guessed that the player was in a bin located in the center of the cube. A similar thing happened in the two dimensional version; however the ability to see the median value of a bin (shown below the visualization) in XmdvTool allowed a user to choose correctly between the bins.

Task 3 times (see Table B.3 in Appendix B.1) have a similar linear appearance. However, this plot (see Figure 4.3) shows that almost all users completed the task faster in the two dimensional version. This shows that there was no improvement in the user's ability to perform this in the three dimensional version.

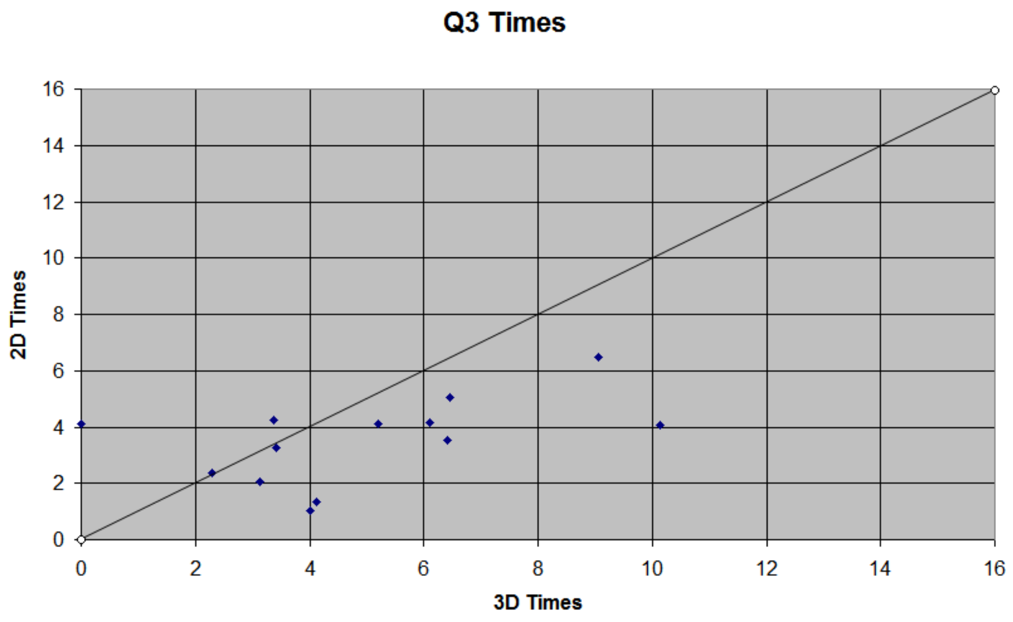


Figure 4.3: Scatterplot of Task 3 times (in minutes)

Only two users were able to locate the player in the three dimensional version. Most users did not have any insight to begin their search and randomly guessed as to the location of the player. This was not the case for the two dimensional version, again because users were able to check their answers by looking at the median value of the bin.

Chapter 5

Conclusions

Looking at the comparisons between the two versions of dimensional stacking, it is clear that there is no statistically significant difference. This may be due to a general lack of understanding about the visualization. Without fully understanding how the visualization works it is difficult to understand where to begin looking for trends in data that is two or three levels deep.

Another reason for this may have been the tasks the users were performing. Many users had difficulty understanding what the numbers represented and where to look. Because XmdvTool displays the median value for the bin under the cursor, all users used this to find the bin that was closest to the data provided. This was a frequent comment during the usage of NLand.

Every user would use the three static views to recenter the data cube based upon the data provided. A hybrid version, with two static views, may be interesting to consider. Each static view would use the dimensions from either the x-axis or the y-axis in a two dimensional stacking. The additional information from this dimensional partitioning may prove useful for seeing trends in two subsets of the dimensions.

Adding additional grid lines would help users to know the division of each axis better. This aids in comprehension and allows the user to properly see the size of a bin. However, in the three dimensional version, these grid lines are part of the cube representation. Drawing all the possible grid lines for a small number of dimensions produces a cube of lines that adds significant clutter. However, drawing only the first division of each axis does not provide enough information for analysis.

Allowing the user to control how many levels of grid lines to draw would provide a method for balancing the amount of information shown and the clutter of the lines. In addition to this control, the grid lines for each level should be colored according to its depth. This gives the user a tool for knowing exactly where in the stacking a bin is located. It also provides visual information about the cardinality of the dimensions whose grid lines are shown.

Additional control over the color and transparency of bins is also needed. Currently the user has no control over which bins are displayed. Adding a control that filters out bins outside some range would allow for better analysis.

In addition to filtering out bins, the user should be able to see some of the structure of the n-dimensional space. When a user clicks on a bin, the selected bin and any bins within a certain distance in n-space could be highlighted. Because there may be one or more bins below the mouse, a system for disambiguating which bin is currently "under" the mouse is needed.

This disambiguation helps also in another improvement. While the cursor is above an occupied bin, information about that bin should be presented to the user. Information such as the actual data records, ranges for the bin and the median value is the type of data that should be displayed. This addition will aid comprehension and give feedback to the users.

The mouse also should control rotation, movement along an axis, and zooming while over the display. This became apparent when several users attempted to rotate and zoom using the mouse and not the controls provided. Providing interactions that are more natural to a user helps to reduce the learning curve and increase comprehension.

Appendix A

Usability Study

A.1 Tutorial

```
Speech for beginning the usability study:

Welcome! During the next 45 minutes you are going to use a
3-dimensional visualization called Dimensional Stacking.
First we are going to introduce you to a simpler version of
this visualization in order for you to become familiar
with this type of visualization.

This is what Dimensional Stacking looks like in 2-dimensions.
Each of the X and Y axes are repeatedly subdivided in a
manner to produce a grid of bins, or small rectangles, that
either contain data or do not. An axis is divided by the
dimension with the fastest speed, largest number of divisions,
and then each of those divisions are subdivided by the next
fastest dimension.

For example in this dataset the fastest dimension is
ft_police which breaks the X axis into for 4 pieces. The
next fastest dimension is unemp which divides the Y axis
into four parts. Now we divide the X-axis again but this
time we divide each of the four divisions of ft_police are
broken into four pieces of manu_workers. This repeated
```

partitioning of both the X and Y axes is continued until there are no more dimensions.

Do you understand how the axes are divided?

Ok, now I will explain what a bin is and how it becomes filled.

A bin is the smallest rectangle formed by the subdivision of the axes. Each bin has one particular set of numbers that defines where in the stacking it belongs. For example the set of numbers 2, 4, 1, 3, 2, 1, 3, represents the bin at the second division of X and the fourth division of Y. The next two numbers, 1 and 3, represent the first subdivision of X, and the third subdivision of Y. What do you think the 2 and 1 represent? Can you find this bin in the graph in front of you?

The data is also broken into partitions based upon the speed of the dimension. For example if a dimension had a minimum value of 0 and a maximum value of 10 and a speed of 5 then a value of 4 would be in the third partition of this dimension. Remember that each bin has a particular set of numbers that represents it. Well a data point can be made into this set of numbers by finding which partition of its dimension it belongs to. This means a bin is filled only if there are one or more data values that produce its set of numbers.

Now that you have been given a brief tutorial on how to read and understand Dimensional Stacking, do you have any questions? If not you can use this version until you feel comfortable.

Now we can introduce you to the 3-dimensional version of Dimensional Stacking.

This version is just like the 2-dimensional one, but instead the dimensions divide three axes instead of two and each bin still has a unique set of numbers that indexes it. Do you have any questions on this version before we go over he basic interactions?

The panel on the right is your control panel, it allows you to move the camera along each of the axes, rotate the cube, reset the view and change the speed of a dimension. You can also move the camera along each of the axes using WS, AD, QE. The left side of the visualization contains a static view of three faces of the cube, the XY, YZ, and ZX. If you click on one of these faces the cube will be repositioned to the view.

Changing the speed of a dimension has the potential to change the order of the dimensions so a legend describing the various colors and which axis a dimension is subdividing has been provided.

Do you have any questions on any of the controls? If not then feel free to use this version until you are comfortable with this version.

Now that you are ready to begin I will explain the procedure. You are going to be asked to perform several tasks and your goal is to do this quickly and accurately.

A.2 Two-Dimensional Tasks

In the 2-dimensional version please do the following task and answer the questions.

Describe any relationship that exists, if any, between games and at_bats.

I lead the league in home runs with 54 and runs batted in with 156. I also have the most runs with 143. Find me. How many other players have similar stats?

I had 378 at bats this season. I also had 16 home runs and a average of .259. In my defense though, I only played in 98 games. Find me. How many players have similar stats?

A.3 Three-Dimensional Tasks

In the 3-dimensional version please do the following task and answer the questions.

Describe any relationship that exists, if any, between games and at_bats.

I lead the league in home runs with 54 and runs batted in with 156. I also have the most runs with 143. Find me. How many other players have similar stats?

I had 378 at bats this season. I also had 16 home runs and a average of .259. In my defense though, I only played in 98 games. Find me. How many players have similar stats?

A.4 Questionnaire

On a scale of 1 to 7 where 1 is easy and 7 is difficult, rate your ability to perform the following tasks.

	Easy						Difficult
Find the bin that a particular data point maps to?	1	2	3	4	5	6	7
Find a cluster of data points?	1	2	3	4	5	6	7
Read the visualization?	1	2	3	4	5	6	7
Move cube around to see various angles?	1	2	3	4	5	6	7
Zoom in and out of the cube?	1	2	3	4	5	6	7
Reposition the cube to a standard view?	1	2	3	4	5	6	7
Distinguish the number of data points in a single in?	1	2	3	4	5	6	7
Change the ordering of how axes are divided?	1	2	3	4	5	6	7
Understand relationships in the data?	1	2	3	4	5	6	7
Use the application?	1	2	3	4	5	6	7

Please explain any difficulties you had with
this visualization.

Please explain any suggestions on how to improve
the visualization.

What did you like about the application? What did
you dislike?

Did you enjoy using the application? Were the interactions with the application intuitive and understandable?

Please explain any other comments you may have.

Appendix B

Study Data

B.1 Task Completion Times

2D Time	3D Time
4.19	0.54
2.44	2.43
8.07	1.40
4.52	2.45
2.35	1.55
4.00	1.05
0.42	0.55
2.42	0.21
0.58	2.45
2.44	0.37
1.46	8.55
2.55	2.02
1.04	2.00

Table B.1: Time to complete Task 1

3D Time	2D Time
1.51	3.49
2.42	2.42
3.01	14.12
5.57	5.24
5.06	4.14
5.39	4.12
3.46	2.16
2.09	2.10
5.34	4.16
10.42	1.18
7.03	2.26
12.19	1.04
4.25	0.37

Table B.2: Time to complete Task 2

2D Time	3D Time
2.36	2.30
4.23	3.38
4.10	0.00
6.47	9.05
4.16	6.11
1.03	4.01
4.12	5.20
5.05	6.45
3.26	3.42
3.54	6.42
4.07	10.14
2.06	3.13
1.36	4.12

Table B.3: Time to complete Task 3

Bibliography

[1] Panel: Visualizing multidimensional (multivariate) data and relations -perception vs. geometry. In *VIS '95: Proceedings of the 6th conference on Visualization '95*, page 405, Washington, DC, USA, 1995. IEEE Computer Society.

[2] V. V. Alexandrov and N. D. Grosky. Recursive approach to associative storage and search of information in data bases. In *Proceedings of the Finnish-Soviet Symposium on Design and Application of Data Base Systems*, pages 271–284, 1980.

[3] D. F. Andrews. Plots of high-dimensional data. *Biometrics*, 28(1):125–136, mar 1972.

[4] J. X. Chen and S. Wang. Data visualization: parallel coordinates and dimension reduction. *Comput. Sci. Eng.*, 3(5):110–112, 2001.

[5] H. Chernoff. The use of faces to represent points in k-dimensional space graphically. *American Statistical Association*, 68(342):361–368, jun 1973.

[6] D. Hilbert. Ueber die stetige abbildung einer line auf ein flachensta 1/4 ck. *Mathematische Annalen*, 38(3):459, 1891.

[7] A. Inselberg and B. Dimsdale. Parallel coordinates: a tool for visualizing multi-dimensional geometry. In *VIS '90: Proceedings of the 1st conference on Visualization '90*, pages 361–378, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press.

[8] D. A. Keim. Pixel-oriented database visualizations. *SIGMOD Rec.*, 25(4):35–39, 1996.

[9] D. A. Keim. Pixel-oriented visualization techniques for exploring very large data bases. *Journal of Computational and Graphical Statistics*, 5(1):58–77, mar 1996.

[10] D. A. Keim, M. Ankerst, and H.-P. Kriegel. Recursive pattern: A technique for visualizing very large amounts of data. In *VIS '95: Proceedings of the 6th conference on Visualization '95*, page 279, Washington, DC, USA, 1995. IEEE Computer Society.

[11] H.-P. Keim, D.A.; Kriegel. Visualization techniques for mining large databases: a comparison. *Transactions on Knowledge and Data Engineering*, 8(6):923–938, Dec 1996.

[12] J. LeBlanc, M. O. Ward, and N. Wittels. Exploring n-dimensional databases. In *VIS '90: Proceedings of the 1st conference on Visualization '90*, pages 230–237, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press.

[13] M. D. Lee, R. E. Reilly, and M. E. Butavicius. An empirical evaluation of chernoff faces, star glyphs, and spatial visualizations for binary data. In *APVis '03: Proceedings of the Asia-Pacific symposium on Information visualisation*, pages 1–10, Darlinghurst, Australia, Australia, 2003. Australian Computer Society, Inc.

[14] B. Mora, J.-P. Jessel, and R. Caubet. Accelerating volume rendering with quantized voxels. In *VVS '00: Proceedings of the 2000 IEEE symposium on Volume visualization*, pages 63–70, New York, NY, USA, 2000. ACM.

[15] K. F. V. Orden and J. W. Broyles. Visuospatial task performance as a function of two- and three-dimensional display presentation techniques. *Displays*, 21(1):17–24, 3 2000.

[16] S. Parker, W. Martin, P.-P. J. Sloan, P. Shirley, B. Smits, and C. Hansen. Interactive ray tracing. In *I3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 119–126, New York, NY, USA, 1999. ACM.

[17] A. Partow. General purpose hash function algorithms. http://www.partow.net/programming/hashfunctions/.

[18] G. Peano. Sur une courbe, qui remplit toute une aire plane. *Mathematische Annalen*, 36(1):157, Jan-1 1890.

[19] H. Pfister, A. Kaufman, and T. Chiueh. Cube-3: A real-time architecture for high-resolution volume visualization. In A. Kaufman and W. Krueger, editors, *1994 Symposium on Volume Visualization*, pages 75–82, 1994.

[20] M. V. Ramakrishna. Hashing practice: analysis of hashing and universal hashing. In *SIGMOD '88: Proceedings of the 1988 ACM SIGMOD international conference on Management of data*, pages 191–199, New York, NY, USA, 1988. ACM.

[21] M. V. Ramakrishna and J. Zobel. Performance in practice of string hashing functions. In *Proceedings of the Fifth International Conference on Database Systems for Advanced Applications (DASFAA)*, pages 215–224. World Scientific Press, 1997.

[22] J. H. Siegel, E. J. Farrell, R. M. Goldwyn, and H. P. Friedman. The surgical implications of physiologic patterns in myocardial infarction shock. *Surgery*, 72:126–141, 1972.

[23] H. S. Smallman, M. S. John, H. M. Oonk, and M. B. Cowen. Information availability in 2d and 3d displays. *IEEE Comput. Graph. Appl.*, 21(5):51–57, 2001.

[24] R. R. Springmeyer, M. M. Blattner, and N. L. Max. A characterization of the scientific data analysis process. In *VIS '92: Proceedings of the 3rd conference on Visualization '92*, pages 235–242, Los Alamitos, CA, USA, 1992. IEEE Computer Society Press.

[25] R. Tipnis, M. O. Ward, and J. LeBlanc. N-land: a graphical tool for exploring n-dimensional data. In *CGI '94: Proceedings of the Computer Graphics International*, 1994.

[26] M. Tory. Mental registration of 2d and 3d visualizations (an empirical study). In *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, page 49, Washington, DC, USA, 2003. IEEE Computer Society.

[27] M. Tory, T. Moller, M. S. Atkins, and A. E. Kirkpatrick. Combining 2d and 3d views for orientation and relative position tasks. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 73–80, New York, NY, USA, 2004. ACM.

[28] J. Zobel, S. Heinz, and H. E. Williams. In-memory hash tables for accumulating text vocabularies. *Inf. Process. Lett.*, 80(6):271–277, 2001.