

A Spreadsheet Approach to Information Visualization

Ed Huai-hsin Chi, Phillip Barry, John Riedl, Joseph Konstan
Department of Computer Science, University of Minnesota
4-192 EE/CS Building, Minneapolis, MN 55455
echi,barry,riedl,konstan @cs.umn.edu

Abstract

In information visualization, as the volume and complexity of the data increases, researchers require more powerful visualization tools that enable them to more effectively explore multi-dimensional datasets. In this paper, we discuss the general utility of a novel visualization spreadsheet framework. Just as a numerical spreadsheet enables exploration of numbers, a visualization spreadsheet enables exploration of visual forms of information. We show that the spreadsheet approach facilitates certain information visualization tasks that are more difficult using other approaches. Unlike traditional spreadsheets, which store only simple data elements and formulas in each cell, a visualization spreadsheet cell can hold an entire complex data set, selection criteria, viewing specifications, and other information needed for a full-fledged information visualization. Similarly, inter-cell operations are far more complex, stretching beyond simple arithmetic and string operations to encompass a range of domain-specific operators. We have built two prototype systems that illustrate some of these research issues. The underlying approach in our work allows domain experts to define new data types and data operations, and enables visualization experts to incorporate new visualizations, viewing parameters, and view operations.

1 Introduction

Spreadsheets have proven to be highly successful tools for interacting with numerical data, such as applying algebraic operations, manipulating rows or columns, and exploring “what-if” scenarios. Spreadsheet techniques have recently been extended from numeric domains to other domains [15, 10]. One possible further application of the spreadsheet is to the domain of information visualization, which often contains datasets that are large, abstract, and multi-dimensional. In addition, datasets can frequently be visually represented in multiple ways. In this paper we present a spreadsheet approach to the display and exploration of information visualizations. We discuss how spreadsheet techniques provide a structured, intuitive, and powerful interface for investigating information visualizations of multidimensional datasets.

Information visualization systems confront such questions as how to represent abstract data visually, what types of exploratory interaction to include, and how to structure this interaction. Therefore, certain capabilities are critical, such as exploring different views of the data interactively, applying operations like rotation or data filtering to a view or group of views, and comparing two or more related datasets. These operations are natural in a spreadsheet environment. The value of a visualization spreadsheet is in enabling scientists to build multiple visual representations of one or several datasets, perform operations on the visual representations together or separately, and compare and contrast them visually.

Spreadsheets offer two key benefits to users of information visualization systems. These benefits derive from the way spreadsheets span a range of application interactive alternatives. On the one hand, spreadsheets are of direct value to end-users, because the

direct manipulation interface makes it easy to view, navigate, and interact with the data. This style of interaction is common in ad hoc spreadsheets that a user creates for a specific goal, such as to compare the lifetime cost of several alternatives for a computer purchase. Columns can be created for each component of the lifetime cost—such as hardware, software, and maintenance—and rows can be created for each type of computer. The resulting spreadsheet makes it easy to explore the alternatives, but requires little work beyond the minimum required to perform the calculations.

On the other hand, spreadsheets provide a flexible and easy-to-learn programming environment. Spreadsheet developers create templates that enable end-users to reliably repeat often-needed computations without the effort of development or coding. The success of spreadsheet-based structured interaction eliminates many of the stumbling blocks in traditional programming environments. For example, the developer does not have to worry about the data dependencies between datasets, nor do users worry about memory management. Behind the scene, these idiosyncrasies of programming are taken care of automatically.

In this paper, we show how to use the spreadsheet paradigm to ameliorate the following two issues. Within an information visualization spreadsheet, much of the user interaction is the application of operations, such as comparison, filtering, and animation. Within each of these operations, there are sub-categories. For example, the user can perform comparisons by looking at the visual representations of the data directly, or by computing the difference between datasets. The required user support for these two types of comparisons is quite different. Visual comparisons require layout strategies that enable views of the data laid out side-by-side, whereas comparisons at the data level require difference operators to be defined at the data domain level.

In information visualization, another large problem involving user-system interactions is the exploration of different methods for representing data. For a given data type, the perfect representation has not yet been discovered, or a ideal representation does not exist; instead, what is required is several different representation methods, because each method extracts different features out of the data. For a given data type there are several different representations available at the user’s disposal. Without some tools to help users to explore this representation space, users are hopelessly lost.

What is exciting about the spreadsheet approach is that it enables information visualizers to solve both problems in a single environment. On the one hand, it facilitates the easy application of operations, such as direct manipulations of the visualizations, and the entry of formulas that specify relationships between cells. On the other hand, it also supports the exploration of different visual representation techniques by emphasizing the operands rather than the operators. Unlike traditional data-flow programming environments where operands are invisible within the flow, a spreadsheet environment emphasizes operands by displaying them in cells and instead hides the operators.

The rest of the paper is structured as follows. In Section 2, we describe past research related to spreadsheets. Section 3 describes the prototypes we have built and the design constraints. Section 4

briefly describes the example domains and prototypes we have chosen. Section 5 illustrates the principles behind the utility of a visualization spreadsheet. Finally, we present some concluding remarks.

2 Related Work

People have long used tables to organize information. The spreadsheet naturally extends the tabular organization of information by allowing the user to specify and interact with the contents and the interconnections of the cells. The spreadsheet paradigm has been suggested in earlier works for domains such as images, volume visualization, and financial data. Here we review the literature related to spreadsheet-based visualization systems.

Tabular Organizations Mathematicians and statisticians have long used tables of sine, cosine, and confidence probabilities. More recently, the invention of the VisiCalc numerical spreadsheet in 1979 fueled the adoption of personal computers [3].

Statisticians have examined visualizing higher dimensional point sets by a table of projections. For example, one multivariate analysis tool is the scatter matrix, which is a table of scatter plots (see [6]). Visualization researchers have applied similar ideas, but in different ways, to produce a table of views of a single dataset [30, 2]. In the scatter matrix, a statistics researcher may mark a datum in one scatter plot, and the program would then highlight the corresponding point in all other scatter plots. These approaches represent a largely static tabular approach to the data, but some interactivity is present, such as rotations, translation, and zooming.

There are several distortion presentation techniques based on a tabular layout [14] such as Document Lens [21], fish-eye views [8, 23], stretching rubber sheets [24].

Spreadsheets for Images The first spreadsheet that allows the display of images in a cell is ASP [19], but it contains no advanced capabilities. Levoy's "Spreadsheets for Images" system [15] and Hasler et. al.'s IISS system [10] examine ways to profitably extend the spreadsheet paradigm to images (as well as to other datasets—Levoy briefly mentions 3D volumes). For example, Levoy shows how a spreadsheet can be used to examine an image processing pipeline, and Hasler shows how many image processing tasks can be efficiently organized in a spreadsheet system. These two systems illustrate some of the capabilities made possible by extending the spreadsheet paradigm to other domains.

Visualization Systems Interest in visualization-based user interfaces has blossomed in the past few years, with systems developed for application areas from hypertext information to geology, molecular biology, file system structure, and animal behavior patterns. Large visualization systems contain modules that users can hook together into a data-flow network to create visualizations. These systems offer many advantages for rapidly building applications. The success of these systems attests to the utility of modular, easy-to-use, extensible tools for visualization tasks. Examples of such systems include ConMan [9], AVS [29, 32], IRIS Explorer [34], IBM Data Explorer [33], and Visualization Toolkit (VTK) [25, 26].

Visual Interactive Spreadsheets Past work in the visualization community has produced interactive tables for specific applications, and include systems such as TableLens [20], FOCUS [27], a graphical financial spreadsheet called FINESSE [31]. The TableLens system [20], designed for browsing tabular numerical information, looks much like a conventional spreadsheet with bar graphs.

The FOCUS interactive table, modeled after TableLens, allows sophisticated navigation via sorting and hiding of information contained in the table, but lacks editing capabilities [27]. FOCUS is similar to TableLens, with the main difference between the two in the interaction methods. TableLens uses a fish-eye layout strategy for display, whereas FOCUS uses a dynamic querying mechanism as the primary interaction method. FINESSE is a prototype system designed for financial data, where the cells are on fixed grids and contain four representation primitives—line plots, 3D surface plots, heat maps, or 3D bar graphs.

The NoPumpG prototype [35] system abandons the fixed tabular grid of conventional spreadsheets, so all cells are free floating. It allows the specification of line plots based on sliders attached to variable values [35]. It is compared to a spreadsheet because of its data dependency capabilities.

Spreadsheet for visualization is a natural extension of the above ideas. Our work focuses on the area of information visualization, and the issues that arise prominently in that domain. We build upon the experiences of other spreadsheets mentioned above, and include a variety of different visual representations and operations useful for interacting with the data. The image spreadsheets (IISS and Levoy's SI) focused on images, and the associated image operations. We take a similar approach to Levoy's SI system in using Tcl as the command language, but we focus on the tasks and operation associated with information visualization. Our work is most like FINESSE [31], but differs from FINESSE because our prototypes allows animation, dynamic visual filtering [5, 1], and dynamic mapping of variables to representation. FINESSE has a limited number of cell primitives, whereas our prototype allows a wide variety of geometric primitives, since our prototype is built on top of the Visualization Toolkit (VTK) [25, 26]. Using a command language, our prototype also allows users to construct their own visual representations of their data. FINESSE focuses on financial data, whereas our system can be tailored to any information visualization tasks. Lastly, in contrast to the visualization spreadsheet, existing large visualization systems are designed for viewing a single visualization at a time. In a data-flow network, a large amount of screen space is devoted to the operators, rather than the operands. We believe that for many applications spreadsheets can provide better interaction.

3 Prototype-Driven Approach

In this section, we present our research approach in building a spreadsheet environment for information visualization. We discuss our prototype driven approach to understanding the problems in this integration, as well as the basic underlying constraints in our design.

3.1 Prototypes

We have taken a prototype-driven research approach in studying how spreadsheet environments can be employed for visualization. To this end, we have constructed two prototype visualization spreadsheet systems.

The first system is a domain-specific study on how spreadsheet can be structured and used in performing specific tasks in analyzing genetic sequence similarity reports, and is called "Spreadsheet for Similarity Reports" (SSR). The system is designed for biologists and their task of comparing similarity reports. SSR is built using OpenGL and Motif using C++, and is built upon the ideas in a previous system we call "AlignmentViewer" [4, 5]. It includes a computational steering environment for rapidly executing the similarity algorithm on multi-processor machines parallel using different algorithm parameters and importing the data. For analysis, it provides animation, filtering, and variable-to-axis mapping capabilities.

The second system is a general visualization spreadsheet built on top of the Visualization Toolkit (VTK) [25, 26]. We call this system “Spreadsheet for Information Visualization” (SIV, pronounced “sieve”). We chose VTK because it provides an object-oriented architecture with many pre-built objects that we can use for exploring the spreadsheet paradigm. Since VTK can be used in conjunction with the Tcl command language and Tk widget toolkit, it facilitates rapid development in an interpreted environment. The system can also run on multiple platforms since VTK and Tcl/Tk are both available under Unix and Windows 95/NT.

3.2 Design Constraints

Our basic design constraints are the elements of a spreadsheet that we consider to be “non-negotiable”. Let us discuss these below.

The tabular layout has proven useful in numerical spreadsheets, and has a number of advantages. First, it enables users to enter data into cells in various configurations. Second, because of its easy-to-comprehend structure, the cells are easily to navigate to and from. Third, because it affords easy grouping, operations can be defined on rows and columns, or portions of a spreadsheet.

Cells are adapted to handle large datasets instead of a few numbers. They handle visual representations of complex data-types with text strings, hierarchical structures, and regular and irregular shapes. These cells may contain references to other datasets in other cells. Because spreadsheets now contain groups of large datasets, users can now see much more than just a single dataset in an established context.

Since the datasets are no longer just simple numbers, the operations now consist a variety of operators for different types of datasets. Certain operators may take columns, rows, or a subgroup of cells as operands. Alternatively, operators may distribute their operation across a group of cells. Because the primary elements are visual, the vocabulary for the spreadsheet is richer. This also results in more difficulties in the design of the user interface for these operations.

One additional philosophical assumption is that since users are accustomed to the spreadsheet metaphor, we expect user skills in numerical spreadsheets to transfer easily to the visualization spreadsheet. The existence of a variety of operators may thwart this transfer. The challenge is to design an intuitive interface for this wide array of operators. Numerical spreadsheets map operators to textual commands to partially solve this problem. In this paper, we examine both command languages as well as direct manipulation interfaces in an attempt to understand how these two techniques can be used in a spreadsheet interface.

4 Domain Prototypes

In this section, we briefly describe the information domains on which our test studies are based and the usage of our prototype systems. In Section 5, we will further demonstrate the usage of the prototypes in these domains, and at the same time, illustrate the principles behind how the spreadsheet paradigm facilitates information visualization tasks. The three domains are molecular biology, time-series matrix visualization, and algorithm visualization. Each of these domains illustrates specific problems we encounter in information visualization analysis tasks. By using a task-centered approach, we show concretely how the visualization spreadsheet enables users to solve problems in information visualization.

4.1 Molecular Biology

Biologists exploring DNA sequences often compare a given sequence against a database of known sequences. Similarity search algorithms produce reports indicating regions of similarity, and other information useful to biologists. These reports can be tens or hundreds of pages long for one sequence.

Previously, we developed a system, called AlignmentViewer, that allows visualization of the most prominent data in such reports [4, 5]. The basic 3D visual representation of this data consists of comb-like glyphs that show the different regions of similarity, how similar they are, and where they occur along the input sequence. (For example, see cell in Figure 1.) The user can explore the data further by such means as interactively rotating, translating or scaling the representation, following a hyperlink to the textual report, mapping the data into a different geometric representation, animating the information over a variable, and filtering the data. The report data has many variables, and only a small number of them can appear in a single 3D visualization. AlignmentViewer mitigates this problem by enabling the user to selectively map the data dimensions onto 3D space, and allowing dynamic filtering of this data. In addition to dynamic query capabilities, we also support several types of animation along any of the dimensions, enhancing the display to 4D.

We chose this data domain for a number of reasons. First, it has a number of properties similar to many datasets we encounter in information visualization: (1) The similarity reports are highly textual, and (2) the similarity relationships between items are an important visualization problem. Second, we are collaborating closely with molecular biologists who can interact with us on a day-to-day basis. This allows us to directly support their information analysis tasks. For example, they have found the ability to compare visualizations for related sequences useful, and have specifically requested the ability to apply a number of different operations to the visualizations simultaneously. This close collaboration has allowed us to directly capture many of the requirements of building a visualization spreadsheet.

Using AlignmentViewer as a basis, we built our first prototype to support the analysis tasks carried out by molecular biologists. Figure 1 shows a snapshot of an example session. The cells are loaded with similarity data between genetic sequences. Each comb glyph within a cell represents an alignment, which is a region of similarity between the input sequence and a sequence from the database [4, 5]. The figure is the result of a three step operation:

- Step 1 Each column is loaded with a different dataset generated from the same input sequence by varying one parameter of the algorithm. Here, we change the parameter that is used to specify the sensitivity of the algorithm with respect to distantly-related versus closely-related sequences. We decrease the distance from far to near in columns 1, 2, and 3, respectively.
- Step 2 We select Row and then subtract cell from each cell in that row. Thus,
- Step 3 At this point, cells in Row and still contain the same datasets as the corresponding cells in Row. We change the variables that are represented on the X, Y, and Z axis, resulting in different views of the datasets.

4.2 Time-series Matrices

Besides similarity data, a time-series of matrices is another type of data that presents challenges of the type commonly encountered in information visualization. Two major difficulties arise in dealing with time-series matrices. The first difficulty is to identify differences in the matrix values between successive matrices. The second

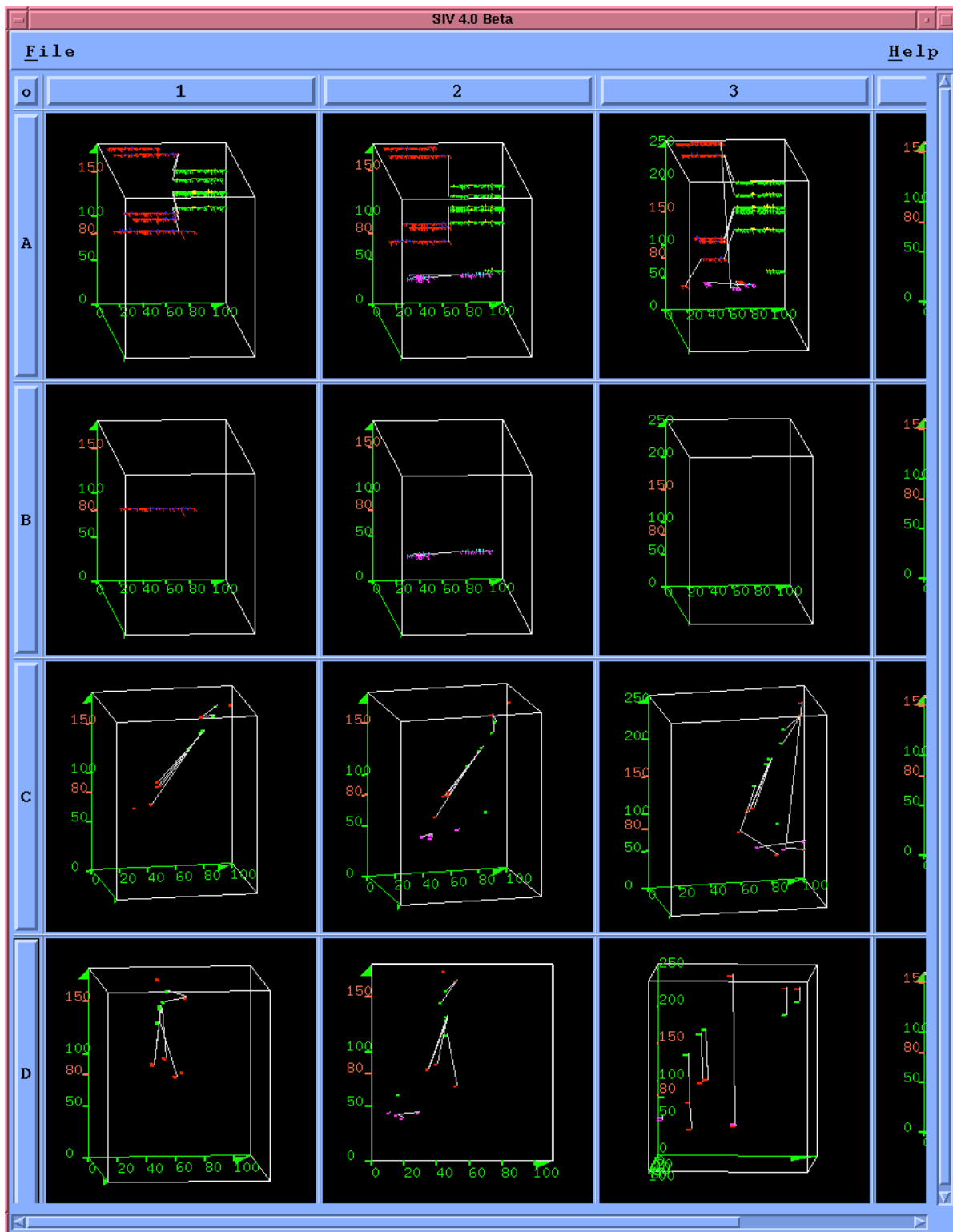


Figure 1: A screen snapshot of the first system (SSR) after performing three operations. (Step 1) Initially, we loaded each column with a slightly different, but related, dataset (, ,). (Step 2) We selected Row , and then subtracted cell from it (). Cell contains the empty set as expected. (Step 3) We changed Row and to show different views of Row . The views show different sets of variables using a different representation, thus increasing our ability to see other dimensions of the multivariate datasets simultaneously.

difficulty is that there are many visual representations that can be applied. For example, the "cityscape" representation shows the matrix values as 3D bars, whereas the "heatmap" representation shows the values as colored tiles [28]. Different representations extract different features, so an easy way to view and explore these several representations simultaneously is needed. Fortunately, the spreadsheet environment is excellent for dealing with these difficulties.

We encountered two matrix series in trying to solve problems with molecular biologists, who are interested in studying the effect of mutation and natural selections on genetic sequences. Natural selection accepts certain mutations, which result in the substitutions of one protein residue by another residue. For a mutation to be accepted, the protein usually must function in a similar way to the old one, presumably due to chemical and physical similarities. PAM and BLOSUM are two series of matrices with each matrix representing substitution probabilities at a given evolutionary distance [7, 11]. The two matrix series were calculated from different sets of information sources. An element of a matrix specifies the relative probability that the amino acids and will be substituted after a given evolutionary interval. A positive entry specifies an accepted mutation that is more likely than random, whereas a negative entry specifies less likely than random.

The detailed nature of this series of matrices results in a large amount of information [7]. For example, these matrices are used in the calculation of similarity between sequences. Unfortunately, the computational molecular biology community have not applied visualization techniques to these matrices. To be sure, biologists are very interested in understanding the nature of these series of matrices due to their mathematical and biological complexity. The computational molecular biology community seeks to understand these matrices, because the choice of which matrix to employ is dependent on the situation.

We have used the SIV system (the second prototype) to try to gain a better understanding of these matrices. We used our system to compare the two matrix series (PAM and BLOSUM), and found that the ability to quickly bring in data and lay them out in different ways to be extremely useful. For example, after 7 lines of commands, the last row shows the BLOSUM62 matrix. To understand the differences between the matrices, it is important to be able to visually compare a number of different matrices simultaneously. In Figure 2, the first, second, third, and fourth rows of cells visualize the PAM40, PAM120, PAM250, and BLOSUM62 matrix, respectively. The first column uses a cube representation that maps positive matrix values to the volume, height, and color attributes of the cubes. The second column uses a carpet plot that maps values to the height and color of a 3D surface (using a rainbow colormap with negative entry mapped to red). The third column uses a bar representation that maps values to the length, height, and color attributes of the bars. The fourth column shows various representations in different rotational configurations.

In Figure 2, by vertically scanning the spreadsheet, the user can detect differences between matrices quickly. As we can see from all the columns, the diagonals of these matrices have strong values, which makes sense since the identity substitution (no mutation) is favored by evolution. From the second column we see that the matrices are quite different because the colors get brighter and brighter from top to bottom. The last row shows the BLOSUM62 matrix, and we see its values are clearly different from any of the PAM matrices shown.

4.3 Algorithm Visualization

A third domain we examined is algorithm visualization. In the past, algorithm visualizations have used animation techniques and sequential layouts to show successive steps. In Section 5, we show how a spreadsheet can be used to easily construct both animations

and tabular layouts of steps for 3D Delaunay triangulation. We also show how we can utilize multiple visual representations to enhance the comprehensibility of the visualization. We use this algorithm as an example of how algorithm visualization can be supported in our visualization spreadsheet.

The algorithm generates 3D random points using random number generators, and then forms tetrahedra from the points using Delaunay triangulation. Delaunay triangulation has been used in scientific and information visualization domains to generate structures around points. 2D Delaunay triangulation is an optimal triangulation and has a number of interesting properties, such as maximizing the minimum angles. However, 3D Delaunay triangulation is much more complicated than 2D, and is a more complex algorithm. Even though the problem of 3D triangulation is well studied, it is still non-intuitive for many people. So visualization techniques can help in gaining better insights into the algorithm.

Figure 3 shows the SIV spreadsheet system loaded with this data. The columns show the results of the algorithm after 5, 6, 25, and 50 steps, from left to right respectively. Row 1 shows the point set using 3D scatter-plots. Row 2 shows the same data using transparent tetrahedra after 3D Delaunay triangulation has been performed on the point sets. Row 3 represents the tetrahedra using edges between vertices. The last row aggregates several cells together to form new visualizations. The combination of geometries from several cells results in visualizations that show differences between successive steps of the algorithm.

5 Illustrated Principles

Via examples in this section, we illustrate how the spreadsheet paradigm facilitates data exploration by enabling researchers to visually compare different values in the cells, perform simple algebraic operations between the values, construct different views and animate the visualizations. We first show how the spreadsheet layout enables comparisons between cells. We examine two techniques for performing this layout operation—direct manipulation and loading a command script. We then show how users can use the spreadsheet for prototyping visualization representations. Finally, we show how users can take advantage of the properties of the spreadsheet to perform operations between cells. By equipping the user with a set of operations, the user can explore datasets in their unique situations by combining the operations in various ways.

5.1 Custom Tabular Layouts Enable Comparisons

The advantages of the tabular layout are that it is familiar, flexible, easily configurable, and excellent for interactive comparison tasks. These advantages are evident in numerical spreadsheets, and translate easily into visualization spreadsheets. Users can construct their own configurations in situations that programmers cannot foresee. Because users are familiar with tables, they can immediately start organizing their data in this spreadsheet metaphor. This flexibility is what contributed to the success of the numerical spreadsheets. It can be tailored to multiple situations in a single tool that is both easy to understand, as well as easy to configure.

For example, comparison tasks are commonplace in numerical spreadsheets, therefore these tasks are easily supported by the visualization spreadsheet. For easy comparison in numerical spreadsheets, users often put two numbers next to each other or load two sets of numbers into adjacent columns. Similarly, in the visualization spreadsheet, users layout two datasets next to each other, or compare two groups of data using adjacent columns. By allowing users to enter data into cells in various configurations, the spreadsheet supports a variety of different tasks. In the following examples, we show how this technique can be adapted in a visualization

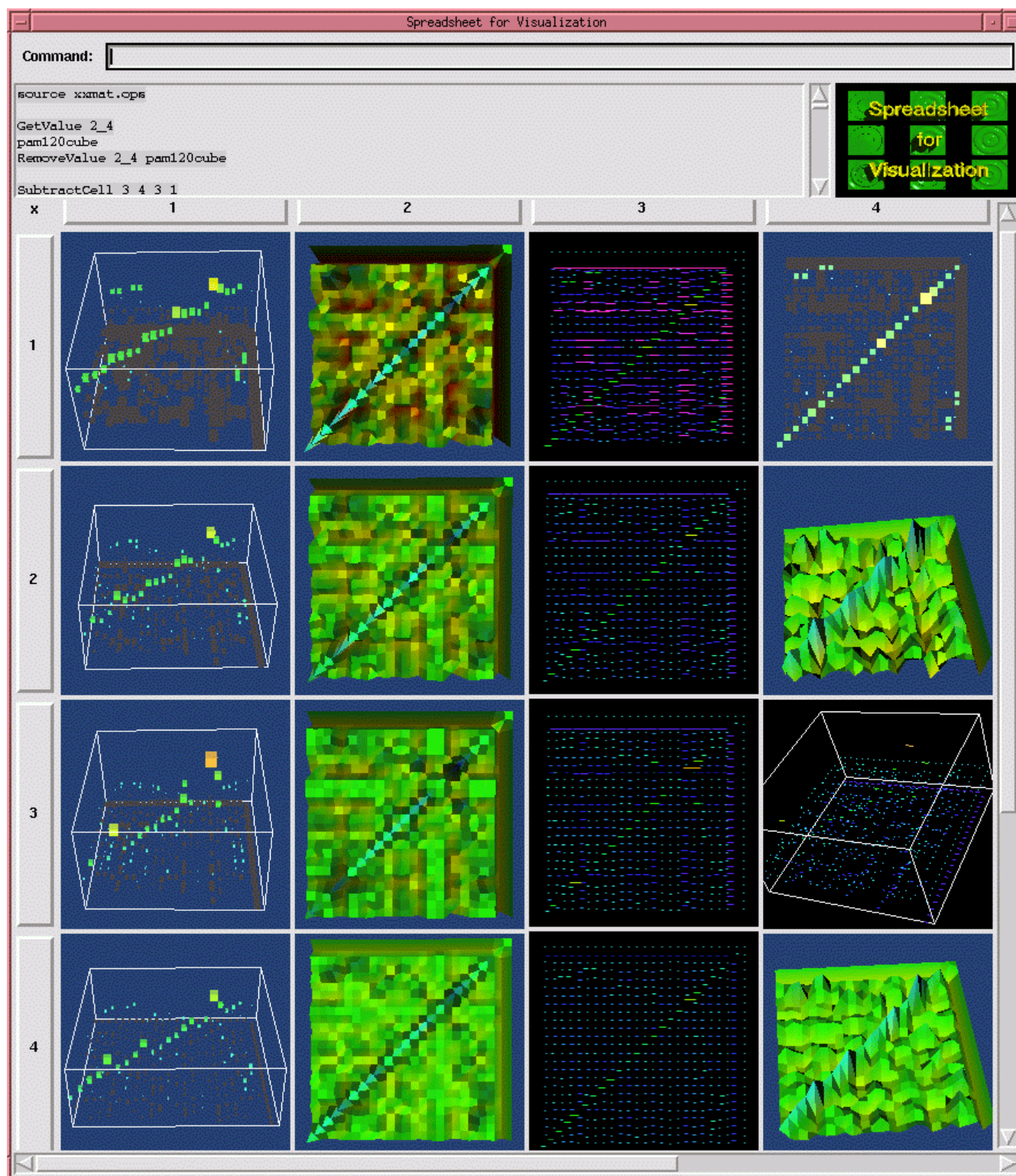


Figure 2: Visualization of time-series matrices. The visualization is built using the second system (SIV). The screen snapshot shows visualizations of protein residue substitution probability matrices of various evolutionary distances. The first, second, and third rows visualize matrix 40, 120, and 250 from the PAM matrix series. The fourth row visualizes matrix 62 from the BLOSUM matrix series. The first column uses a cube representation that maps positive matrix values to the volume, height, and color attributes of the cubes. The second column uses a carpet plot that maps values to the height and color of a 3D surface. The third column uses a bar representation that maps values to the length, height, and color attributes of the bars. The fourth column shows various representations in different rotational configurations.

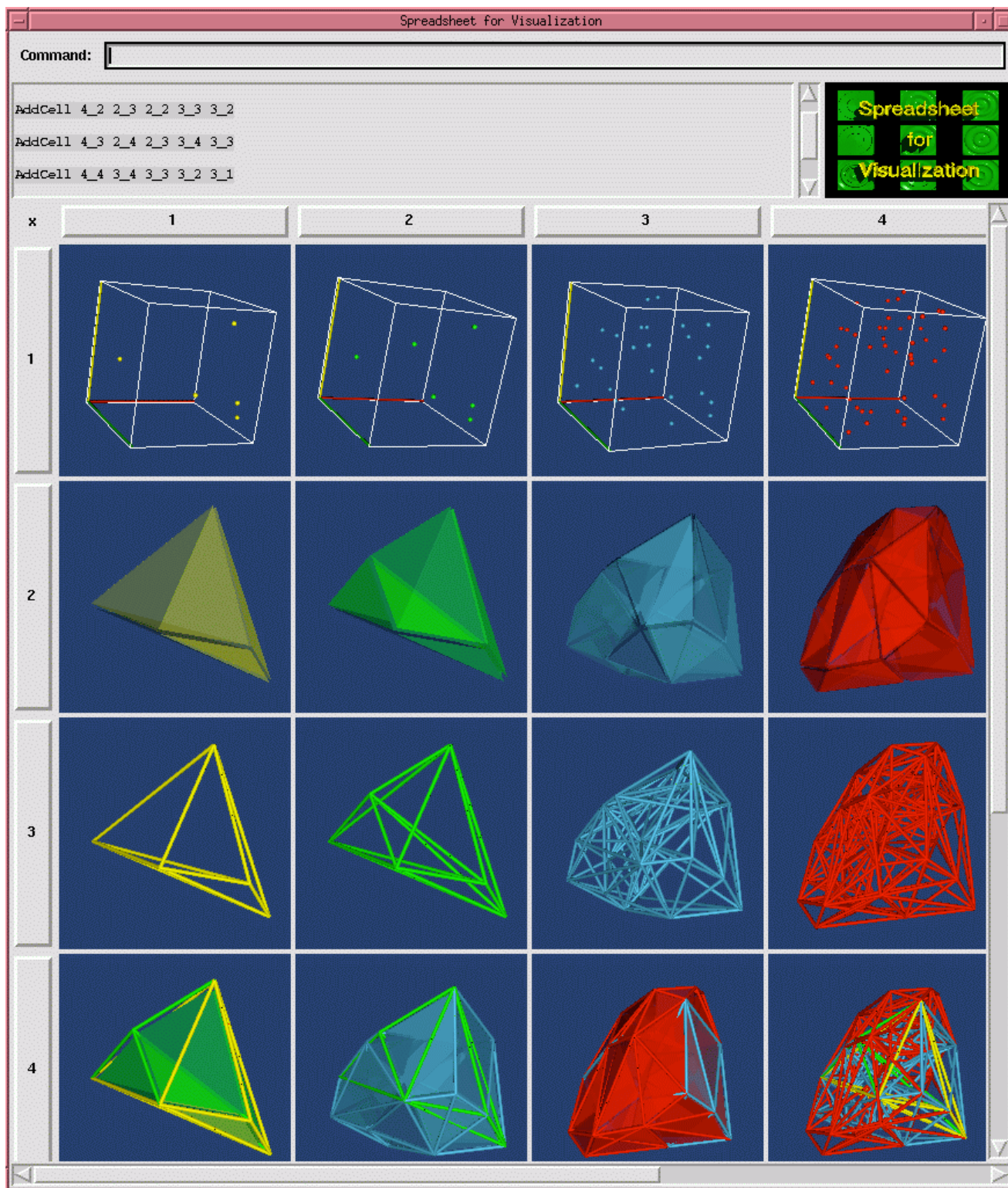


Figure 3: Visualization of 3D random point generation and Delaunay triangulation of the resulting point set. The columns visualize the outcome of the algorithm after 5, 6, 25, and 50 steps, respectively. The last row shows the result of several addition operations (the formula syntax is “command **result** operands”):

AddCell 4_1 3_2 3_1 2_2 2_1;
 AddCell 4_2 3_3 3_2 2_3 2_2;
 AddCell 4_3 3_4 3_3 2_4 2_3;
 AddCell 4_4 3_4 3_3 3_2 3_1;

spreadsheet to show different datasets, different visual representations, and several time steps in the time dimension.

The tabular organization of the spreadsheet enables the user to immediately detect differences between the visualizations of different datasets. For example, even viewers without experience with molecular biology can see that the general structure of the datasets in Figure 1 are similar, but that some alignments that are present in cells A2 and A3 do not appear in A1. Users can now take advantage of their visual comparison abilities to detect differences between datasets.

Users of the spreadsheet can also use it to compare different visual representations. In Figure 2, the tabular layout is used to show different visual representations in different columns. Across each row, the values in the cells are the same. The visual representation in each cell of a row has been changed to bring out different features of the dataset.

The columns and rows of the table increase the number of dimensions we can see simultaneously. In Figure 3, the columns show several snapshots of the steps of the 3D Delaunay algorithm. So in this case, the columns are used to represent the time dimension.

As the above examples show, the tabular layout is one of the reasons why spreadsheet-based environments are so powerful. The organization is familiar to users, and simple direct manipulation operations can be used to rotate contents in the cells. It can be custom tailored to individual situations on-the-fly.

5.2 Direct Manipulation and Command Languages

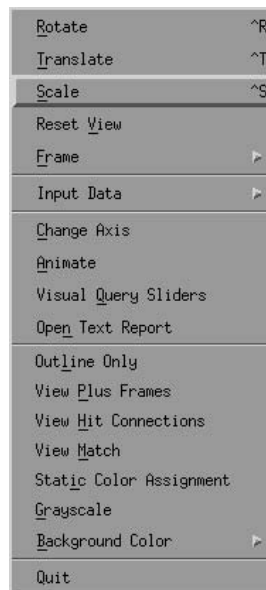
How to access and apply operations is an important aspect of the spreadsheet. We examined two different methods for performing spreadsheet operations.

The first method is a direct manipulation interface corresponding to a “noun-verb” model, where the user first selects a group of cells (the noun), and then applies an operation (the verb) to those cells. The operation is specified using a combination of menus and dialog boxes. For example, to set up the similarity data in Figure 1, the user first selects a column of cells, then performs a single import operation of a large dataset into those cells. Some example menus and dialog boxes used in SSR system is shown in Figure 4.

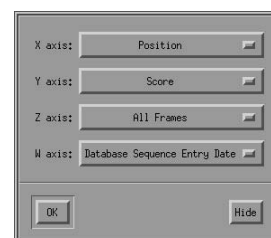
The second method is a command and script language based interface. The user can interactively enter commands in an entry box, similar to a traditional numerical spreadsheet. Alternatively, she can write a script file and load in the script. For example, she can define a layout by writing a script that specifies the datasets and the representation method used for each cell. The script file can contain other non-layout commands such as animation, or even define new commands. In Figure 2, we can see in the history window, the user has just loaded a script with a pre-defined layout.

The command language can also be used to define modules to extend the spreadsheet, such as file input modules or modules that define a visual representation for a given data type. To use the module, the user simply loads the module, and the new commands in that module become available to her. For example, in Figure 2 the user programmed new modules that implement new representations for matrices using the command language. The command language we defined for the visualization spreadsheet includes operators such as `AddCell`, `SubtractCell`, `Scatterplot`, `ReadBioMatrix`, and `Carpetplot`. The operators follow the convention of “command result arguments”, where command operates on the arguments and puts the outcome in cell result.

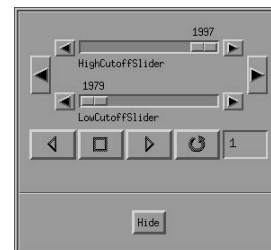
From our experience of the two systems, we believe a combination of the two approaches is appropriate for the visualization spreadsheet. The advantages of a menu-based interface are that it is relatively intuitive to use for first-time users, and training time for new users is short. However, because there are many functionalities



(a) The main popup menu is accessed by holding down the right mouse button.



(b) The mapping dialog box.



(c) The animation dialog box.

Figure 4: Our spreadsheet visualization system for molecular biology uses a direct manipulation interface with menus and dialog boxes, which makes the system easy to use.

in a visualization spreadsheet system, there is the danger of creating a large number of menus with no structure to them. Menu systems also tend to slow down frequent knowledgeable users. The advantages of a command language based interface are its flexibility and its appeals to power users. Command languages can also be used to construct macros so complex tasks can be performed rapidly. The disadvantages are that command languages are difficult to master and require substantial training and memorization.

5.3 Exploring Visual Representations

For a given data type, there are many visual representation techniques to choose from. Often, a technique contributes to the finding of one visual feature, while another visually extracts a different visual feature. Fortunately, the spreadsheet environment assists in the organization and display of various visual representations. Because our system can be easily extended to handle new techniques, it allows us to quickly prototype and compare several representation techniques. Here we show this flexibility in all three data domains.

By constructing several modules for different visual representations of matrices, we used the SIV spreadsheet to answer specific scientific questions on protein residue substitution time-series matrices. For example, we used it to discover several novel patterns in these matrices. In Figure 2, the cube representation used in the first column shows the interesting variation of the diagonal entries more clearly than the other representation methods. The entry represented by the orange cube varies more than any other entry. The carpet plot technique used in the second column shows that the matrices have different ranges of values (i.e. the colors get brighter and brighter from top to bottom). In the third column, the bar-plot tech-

nique makes comparing a specific entry from matrix to matrix easy, and shows the overall trend of most off-diagonal entries to decrease.

The algorithm visualization of Figure 3 shows several different visual representations of a random 3D point generator. Row 1 shows the results of the random point generator as 3D scatter plots. We can see the spread of the points quite well in this representation. Row 2 shows the same data using transparent tetrahedra after 3D Delaunay triangulation has been performed on the point sets. Through interactive rotation, this representation gives a better view of the relative placement of the points. It also shows the convex hulls of the point sets, and how the hulls change between steps of the algorithm. Row 3 represents the Delaunay triangulation as edges rather than tetrahedra, thus giving a better view of the interior structure of the triangulation.

Like SIV, SSR also allows changing of visual representation for similarity reports. A mapping tool enables the user to choose the geometric representation used by the cell(s). In Figure 1, the cells in Row 4 and 5 contain the same datasets as the corresponding cells in Row 1, but we changed the mapping in Row 4 and 5 to show different variables of the similarity report. In this organization, the cells in a given column represent the same value; however, each row offers a different view of the data. The ability to map different variables to different axes in different cells results in an improved ability to see more variables simultaneously. (In SSR, all these operations are accomplished via a click-and-point interface. The user loads the columns with data one column at a time, and changes the mapping of the data of each row using the mapping tool dialog box. Shown in Figure 4, the mapping tool is implemented as a pull-down menu for each axis.)

Our experience shows the elegant organization of the spreadsheet allows interesting ways of combining different visual representations of the underlying data. Users can compare and visually extract different features from the different representations. The spreadsheet environment equips users with the necessary tools to explore the representation space.

5.4 Application of Operators

In the process of data exploration, a large amount of user interaction is the application of operators to datasets. The visualization spreadsheet facilitates these interactions by enabling users to explore 'what-if' scenarios in a structured environment. For example, users can copy and then modify the contents of a cell, or perform an operation on two cells and put the result in a third cell. Whereas the application of operators has largely been viewed as a sequential process in other environments, the spreadsheet environment is capable of supporting non-sequential spontaneous explorations.

In the algorithm visualization of Figure 3, we show how the visualization spreadsheet can be used to quickly perform operations between successive steps of the 3D Delaunay triangulation. For example, by adding the geometric contents of cells together, the user can aggregate representations together to create new representations that show differences between the steps of the algorithm. Row 4 is the result of performing the following operations:

```
AddCell 4_1 3_2 3_1 2_2 2_1;
AddCell 4_2 3_3 3_2 2_3 2_2;
AddCell 4_3 3_4 3_3 2_4 2_3;
AddCell 4_4 3_4 3_3 3_2 3_1;
```

Cells 4_1 through 4_4 shows differences between steps of the algorithm. Cell 4_1 shows the difference between step 5 and 6, whereas 4_2 shows the difference between step 6 and 25. We can see where new points were added into the point set, as well as the structural changes in the convex hulls between steps. In cell 4_4, we see the convex hull after 25 steps is almost completely embedded inside the convex hull obtained after 50 steps. We see the blue surfaces and

vertices where the convex hull has not changed. Cell 4_3 shows the aggregate of adding all of the stick models in Row 3 together. These representations are discovered after many iterations of trying different combinations of the points, sticks, and surface representations of the data in Row 1, 2 and 3.

The spreadsheet paradigm also provides a simple interface for performing operations such as set subtraction or addition. Using the similarity report example in Figure 1, let us demonstrate by first constructing Row 4 as a duplicate of Row 1, then subtract cell 4_1 from each cell. Thus, Cell 4_1 contains the empty set as expected. Cells 4_2 and 4_3 show alignments found by using far evolutionary distance parameters, but not by the near evolutionary distance parameter used in Row 1. The subtraction operation is a typical case of comparing two similar, but not identical datasets, something of interest to researchers in many fields. The spreadsheet approach makes such algebraic manipulations straightforward.

These algebraic operations can take on multiple semantics at different levels. At the low level, we can capture the cell images and perform image subtractions, which is done by subtracting corresponding pixels. At the mid level, we can perform geometric unit algebraic operations. In VTK, these geometric units are called "actors" in the scene. We can define actor objects and algebraically add them to or subtracting them from the scene. At the high level, we can perform subtractions based on domain semantics. For example, in our similarity report example above, the values are the sets of alignments, and we define two alignments to be equal if they share a region. Likewise, high-level algebraic operations in other information visualization domains should be based on domain-specific semantics.

Within the domain-specific semantic level, sometimes there are several possible definitions for the operator. For example, the set difference operator above is only one of the three possible interpretations. We actually define three different types of equality between alignments, resulting in three difference operators. The three equalities are name, overlap, and exact. In name equality, two alignments are considered equal if they occur in the same database sequence. In overlap equality, the alignments must also share an overlapping region. In exact equality, the alignments must share the same exact region.

Other than algebraic operators, our systems also provide other operations, such as animation and filtering. The algorithm animation example can be visualized as an animation of steps across the rows. The SSR system also provides animation, as well as dynamic query filtering capabilities. The animation tool provides accumulative, or sliced animation over any variable [5], and adds an extra dimension to the spreadsheet. A synchronized animation can be performed on a group of cells simultaneously. In Figure 1 for example, suppose we are interested in the distributions of the lengths of the alignments, so we animate the cells over the length variable. Animation shows the extra alignments in Cell 4_1 are short alignments when compared to the alignments in Cell 4_2. A filtering tool enables the user to explore subsets of the data. When the user interactively adjusts sliders controlling each variable, the view is updated in real-time. Using the filtering tool, closer inspection reveals that the short alignments in Cell 4_1 are between 11 and 29 residues long.

One common, but just as important, interaction is the application of direct manipulation operations such as rotation, translation, and zooming. Performing these operations in a spreadsheet environment does have an interesting twist. Often we want to be able to apply the same operation to multiple cells simultaneously. We have implemented this feature in both systems, and have found it to be extremely useful for comparison tasks. For instance, the user can click on the first row button in Figure 3 to select the first row. Then she can perform rotations simultaneously on all of the cells in that row, giving a rotationally-coordinated view of the data. This feature

is useful in this situation because we want the scatter plot to be in similar orientations to provide correspondence between the points in different cells. Similarly, we found the ability to propagate these view changes to be highly valuable in the matrix visualization example of Figure 2. By selecting a row, we can compare the various visual representations in the same orientation. Or alternatively, we can select a column and compare different matrices using the same visual representation.

6 Conclusion

Visualization research spans a remarkable range of scientific disciplines and corresponding visualization techniques. Visualization researchers have discovered that certain operations are needed across this entire range. These operations include comparing visualizations of two different datasets, as well as performing algebraic operations on two or more visualizations, such as visualizing the difference between two datasets. Furthermore, the need to explore multiple visual representations simultaneously arises especially in information visualization, because different techniques often extract different visual features and the complexity of the data. A visualization spreadsheet is an excellent way to address these issues that involve multiple visualizations.

Over the past year we have learned that the spreadsheet approach is a powerful and intuitive technique for interacting with 3D information visualizations. In this paper, we showed that a visualization spreadsheet supports information visualizers as they are confronted by the challenge of visualizing a wide variety of different types of data. Two types of interaction tasks are important to the users. One is being able to quickly prototype an application for interacting with data. The other is being able to apply operations such as compare/contrast, rotation/translation/zooming, filtering/sorting/searching and other domain-specific operations. The visualization spreadsheets described in this paper have proven useful in the above two tasks in the domains we examined—visualizing sequence similarity data in molecular biology, two sets of time-series matrices, and visualizing the steps in the 3D Delaunay triangulation algorithm.

For each domain, we showed how our prototype spreadsheets enabled users to compare visualizations in cells using the tabular layout. Using these domain examples, we also showed how users use the spreadsheet to display, manipulate, and explore multiple visual representation techniques for their data. By applying different operations to the cells, we showed how visualization spreadsheets afford the construction of 'what-if' scenarios. The possible set of operations that users can apply is now a rich set of domain-dependent as well as domain-independent operators, such as animation, filtering, and algebraic operators between cells. Subtraction between cells, for example, can be applied both at the pixel level as well as at the geometric-object level. Other operations may now be coordinated, such as applying the same rotation manipulation across a group of cells.

We also examined the differences between two interaction styles—command language and direct manipulation. Our first prototype for visualizing sequence similarity data uses the noun-verb direct manipulation model. We find that this interaction style is somewhat less flexible than a command language, but supports most of the needed flexibility in an easy-to-use framework for a specialized domain. The command language used in the second prototype is based on Tcl, and is considerably more flexible for users to define their own macros and modules. It also allowed expert users to quickly perform a number of complex operations.

The spreadsheet approach is a powerful and intuitive technique for interacting with the information visualizations in a structured way. Further research is needed to understand the properties of visualization applications that work well in spreadsheets, to investigate

the appropriate user interfaces at all levels, and to develop a framework to enable rapid development of visualization spreadsheet applications.

Acknowledgments

This work has been supported in part by the National Science Foundation under grants BIR 940-2380 and CDA 9414015.

References

- [1] C. Ahlberg and B. Shneiderman. Visual information seeking: Tight coupling of dynamic query filters with starfield displays. In *Proceedings of ACM CHI'94 Conference on Human Factors in Computing Systems*, volume 1 of *Information Visualization*, pages 313–317, 1994. Color plates on pages 479–480.
- [2] V. Anupam, S. Dar, T. Leibfried, and E. Petajan. Dataspace: 3-d visualizations of large databases. In *IEEE Information Visualization Symposium*, pages 82–88, 144, 145, 1995.
- [3] P. S. Brown and J. D. Gould. An experimental study of people creating spreadsheets. *ACM Transactions on Office Information Systems*, 5(3):258–272, July 1987.
- [4] E. H. Chi, P. Barry, E. Shoop, J. Carlis, E. Retzel, and J. Riedl. Visualization of biological sequence similarity search results. In *IEEE Visualization '95*, pages 44–51. IEEE CS Press, 1995.
- [5] E. H. Chi, J. Riedl, E. Shoop, J. V. Carlis, E. Retzel, and P. Barry. Flexible information visualization of multivariate data from biological sequence similarity searches. In *IEEE Visualization '96*, pages 133–140, 477. IEEE CS Press, 1996.
- [6] W. Cleveland and M. McGill, editors. *Dynamic Graphics for Statistics*. Wadsworth & Brooks/Cole, Belmont, CA, 1988.
- [7] M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt. A model of evolutionary change in proteins. In M. O. Dayhoff, editor, *Atlas of Protein Sequence and Structure, Vol. 5, Suppl. 3*, chapter 22, pages 345–352. National Biomedical Research Foundation, 1978.
- [8] G. W. Furnas. Generalized fisheye views. In *Proceedings of ACM CHI'86 Conference on Human Factors in Computing Systems*, Visualizing Complex Information Spaces, pages 16–23, 1986.
- [9] P. E. Haeberli. ConMan: A visual programming language for interactive graphics. In *Computer Graphics*, volume 22, pages 103–111. ACM SIGGRAPH, August 1988.
- [10] A. F. Hasler, K. Palaniappan, and M. Manyin. A high performance interactive image spreadsheet (IIS). *Computers in Physics*, 8(3):325–342, May/June 1994.
- [11] S. Henikoff and J. Henikoff. Performance evaluation of amino acid substitution matrices. *Proteins: Structure, Function, and Genetics*, 17:49–61, 1993.
- [12] S. E. Hudson. User interface specification using an enhanced spreadsheet model. *ACM Transactions on Graphics*, 13(3):209–239, July 1994.
- [13] S. E. Hudson and S. P. Mohamed. Interactive specification of flexible user interface displays. *ACM Transactions on Information Systems*, 8(3):269–288, 1990.

- [14] Y. K. Leung and M. D. Apperley. A review and taxonomy of distortion-oriented presentation techniques. *ACM Transactions on Computer-Human Interaction*, 1(2):126–160, 1994.
- [15] M. Levoy. Spreadsheet for images. In *Computer Graphics (SIGGRAPH '94 Proceedings)*, volume 28, pages 139–146. SIGGRAPH, ACM Press, 1994.
- [16] T. Munzner, P. Burchard, and E. H. Chi. Visualization through the World Wide Web with Geomview, Cyberview, W3Kit, and WebOOGL. World Wide Web Fall 1994 Conference, Chicago IL, October 1994.
- [17] B. A. Myers. Graphical techniques in a spreadsheet for specifying user interfaces. In *Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems*, User Interface Management Systems, pages 243–249, 1991.
- [18] B. A. Myers, D. A. Giuse, R. B. Dannenberg, B. Vander Zanden, D. S. Kosbie, E. Pervin, A. Mickish, and P. Marchal. Comprehensive support for graphical, highly-interactive user interfaces: The Garnet user interface development environment. *IEEE Computer*, 23(11):71–85, November 1990.
- [19] K. W. Piersol. Object-oriented spreadsheets: The analytic spreadsheet package. In N. Meyrowitz, editor, *Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA)*, pages 385–390, Portland, OR USA, Nov. 1986. ACM Press, New York, NY, USA. Published as SIGPLAN Notices, volume 21, number 11.
- [20] R. Rao and S. K. Card. The table lens: Merging graphical and symbolic representations in an interactive focus+context visualization for tabular information. In *Proceedings of ACM CHI'94 Conference on Human Factors in Computing Systems*, volume 1 of *Information Visualization*, pages 318–322, 1994. Color plates on pages 481–482.
- [21] G. G. Robertson and J. D. Mackinlay. The document lens. In *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology*, Visualizing Information, pages 101–108, 1993.
- [22] G. G. Robertson, J. D. Mackinlay, and S. K. Card. Cone trees: Animated 3d visualizations of hierarchical information. In *Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems*, Information Visualization, pages 189–194, 1991.
- [23] M. Sarkar and M. H. Brown. Graphical fisheye views of graphs. In *Proceedings of ACM CHI'92 Conference on Human Factors in Computing Systems*, Visualizing Objects, Graphs, and Video, pages 83–91, 1992.
- [24] M. Sarkar, S. S. Snibbe, O. J. Tversky, and S. P. Reiss. Stretching the rubber sheet: A metaphor for visualizing large layouts on small screens. In *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology*, Visualizing Information, pages 81–91, 1993.
- [25] W. J. Schroeder, K. M. Martin, and W. E. Lorensen. The design and implementation of an object-oriented toolkit for 3d graphics and visualization. In R. Yagel and G. M. Nielson, editors, *IEEE Visualization '96*, pages 93–100. IEEE CS Press, 1996.
- [26] W. J. Schroeder, K. M. Martin, and W. E. Lorensen. *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. Prentice Hall, 1996.
- [27] M. Spence, C. Beilken, and T. Berlage. FOCUS: The interactive table for product comparison and selection. In *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology*, pages 41–50, 1996.
- [28] E. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 1992.
- [29] C. Upson, T. Faulhaber, Jr., D. Kamins, D. Laidlaw, D. Schlegel, J. Vroom, R. Gurwitz, and A. van Dam. The application visualization system: A computational environment for scientific visualization. *IEEE Computer Graphics and Applications*, pages 30–42, July 1989.
- [30] J. van Wijke and R. van Liere. Hyperslice: Visualization of scalar functions of many variable. In *IEEE Visualization '91*, pages 119–125, Los Altimos, CA, 1991. IEEE CS Press.
- [31] A. Varshney and A. Kaufman. FINESSE: A financial information spreadsheet. In *IEEE Information Visualization Symposium*, pages 70–71, 125, 1996.
- [32] Advanced Visualization System home page. <http://www.avv.com>, Feb. 1997.
- [33] IBM Visualization Data Explorer (DX). <http://www.almaden.ibm.com/dx/>, Feb. 1997. (current as of date).
- [34] IRIS Explorer home page. http://www.nag.co.uk:80/Welcome_IEC.html, Feb. 1997.
- [35] N. Wilde and C. Lewis. Spreadsheet-based interactive graphics: From prototype to tool. In *Proceedings of ACM CHI'90 Conference on Human Factors in Computing Systems*, Application Areas, pages 153–159, 1990.