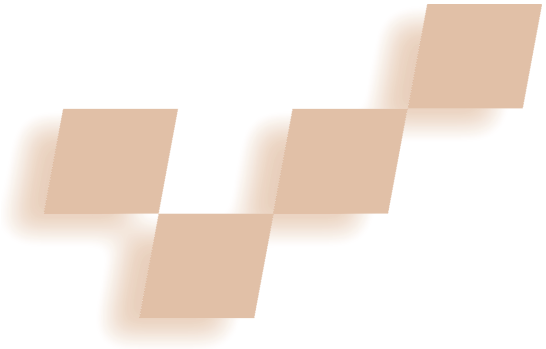
Principles for Information Visualization Spreadsheets



Ed Huai-hsin Chi, John Riedl, Phillip Barry, and Joseph Konstan
University of Minnesota

The visualization spreadsheet provides a framework for exploring large and complex data sets. Structuring user interactions using a spreadsheet paradigm creates a powerful tool for information visualization.

Spreadsheets have proven highly successful for interacting with numerical data, such as applying algebraic operations, defining data propagation relationships, manipulating rows or columns, and exploring "what-if" scenarios. Spreadsheet techniques have recently been extended from numeric domains to other domains.[1,2] Here we present a spreadsheet approach to displaying and exploring information visualizations, with large, abstract, multidimensional data sets that are visually represented in multiple ways. We illustrate how spreadsheet techniques provide a structured, intuitive, and powerful interface for investigating information visualizations.

An earlier version of this article appeared in the proceedings of the 1997 Information Visualization Symposium.[3] Here we refocus the discussion to illustrate principles that make the spreadsheet approach powerful. These principles show how we can perform many user tasks easily in the visualization spreadsheet that prove much more difficult using other approaches.

Why spreadsheets?

The visualization spreadsheet's benefit comes from enabling users to build multiple visual representations of several data sets, perform operations on these visualizations together or separately, and compare and contrast them visually. These operations are becoming ever more important as we realize certain interaction capabilities are critical, such as exploring different views of the data interactively, applying operations like rotation or data filtering to a group of views, and comparing two or more related data sets. These operations fit naturally into a spreadsheet environment. These benefits derive from the way spreadsheets span a range of user interactions. On the one hand, spreadsheets directly benefit end users, because the direct manipulation interface makes it easy to view, navigate, and interact with the data. On the other hand, spreadsheets provide a flexible and easy-to-learn environment for user programming.

The success of spreadsheet–based structured interaction eliminates many of the stumbling blocks in traditional programming environments. Spreadsheet developers create templates that enable end users to reliably repeat often-needed computations without the effort of redevelopment or coding. Users do not have to worry about the data dependencies between data sets or memory management. These programming idiosyncrasies are taken care of automatically. By providing a natural environment to explore and apply operations on data, visualization spreadsheets easily enable the exploration of data sets.

What is a visualization spreadsheet?

Based on our experiences and drawing on others' past work,[1-3] we define the spreadsheet paradigm's characteristics as follows:

- The *tabular layout* lets users view collections of visualizations simultaneously. Cells can handle large data sets instead of a few numbers.
- *Operators* are available for generating or modifying cell contents. The visualization spreadsheet includes a rich variety of operators for different types of data sets. Operators can be applied across a specified range of operand cells, such as a whole column.
- The spreadsheet keeps track of the *dependencies between cells* and automatically updates the cells appropriately when they are manipulated.

These characteristics give the spreadsheet paradigm its fundamental advantages. We borrow these charac-

0272-1716/98/$10.00 © 1998 IEEE

teristics from numeric spreadsheets. The fundamental advantages form a set of design constraints that we consider "non-negotiable." A tool must satisfy these characteristics to merit the term "visualization spreadsheet."

How does our work fit into past research?

Using a spreadsheet for visualization extends past research quite naturally. Our work focuses on information visualization and the issues that arise prominently in that domain. We build upon the experiences of other spreadsheet researchers and include a variety of different visual representations and operations useful for interacting with the data.

The image spreadsheets—Interactive Image Spreadsheet[2] and Levoy's Spreadsheet for Images[1]—focused on images and the associated image operations. We take an approach similar to Levoy's system in using Tcl as the command language, but we extend the spreadsheet beyond images into information visualization and its associated tasks and operations.

Our work most resembles Finesse.[4] We built on past work by discovering the principles from which the visualization spreadsheet derives its power. Finesse has a limited number of cell primitives, whereas our prototype—built on top of the Visualization Toolkit (VTK)[5]—allows a wide variety of geometric primitives. Our work differs from Finesse because our prototype uses a command language and allows animation, dynamic visual filtering, and dynamic mapping of variables to representation. Using a command language, our prototype also lets users construct their own visual representations of their data. Finesse focuses on financial data, whereas our system can be tailored to any information visualization tasks.

In contrast to the visualization spreadsheet, existing large visualization systems generally present a single visualization at a time. The benefit of data-flow network visualization systems such as IBM Data Explorer (http://www.almaden.ibm.com/dx/) is its presentation of the visualization process as a sequence of operator modules. The screen space is allocated to present the operators, rather than the operands. Visualization spreadsheets are better suited than data-flow networks for applications in which most of the screen real estate should focus on the intermediate results.

Principles of visualization spreadsheets

Via examples, we illustrate the principles behind how the spreadsheet paradigm facilitates information visualization tasks. We illustrate how the visual spreadsheet paradigm facilitates data exploration by enabling researchers to derive comparison data sets using operators such as set addition and subtraction. We also illustrate how the spreadsheet paradigm enables the parallel application of operators to a range of cells, facilitating visual comparison of values in the cells.

In information visualization, another large problem involves user-system interactions: a given data type will have several different visual representations at the user's disposal. We discuss how to use the spreadsheet paradigm to enable the exploration of multiple visual features in the spreadsheet simultaneously. This is especially useful in information visualization, since there are several different visualization representation techniques available at the user's disposal for a given data type. Moreover, by constructing a layout configuration, we show that the user can set up analysis templates for application to many data sets. Equipping users with a set of operations lets them explore data sets in their unique situations by combining the operations in various ways.

We built our system, Spreadsheet for Information Visualization (SIV, pronounced "sieve"), on top of a multi-platform interpreted development system combining Tcl/Tk and the Visualization Toolkit (VTK).[5] VTK provides an object-oriented architecture with many pre-built objects.

Via examples, we illustrate the principles behind how the spreadsheet paradigm facilitates information visualization tasks.

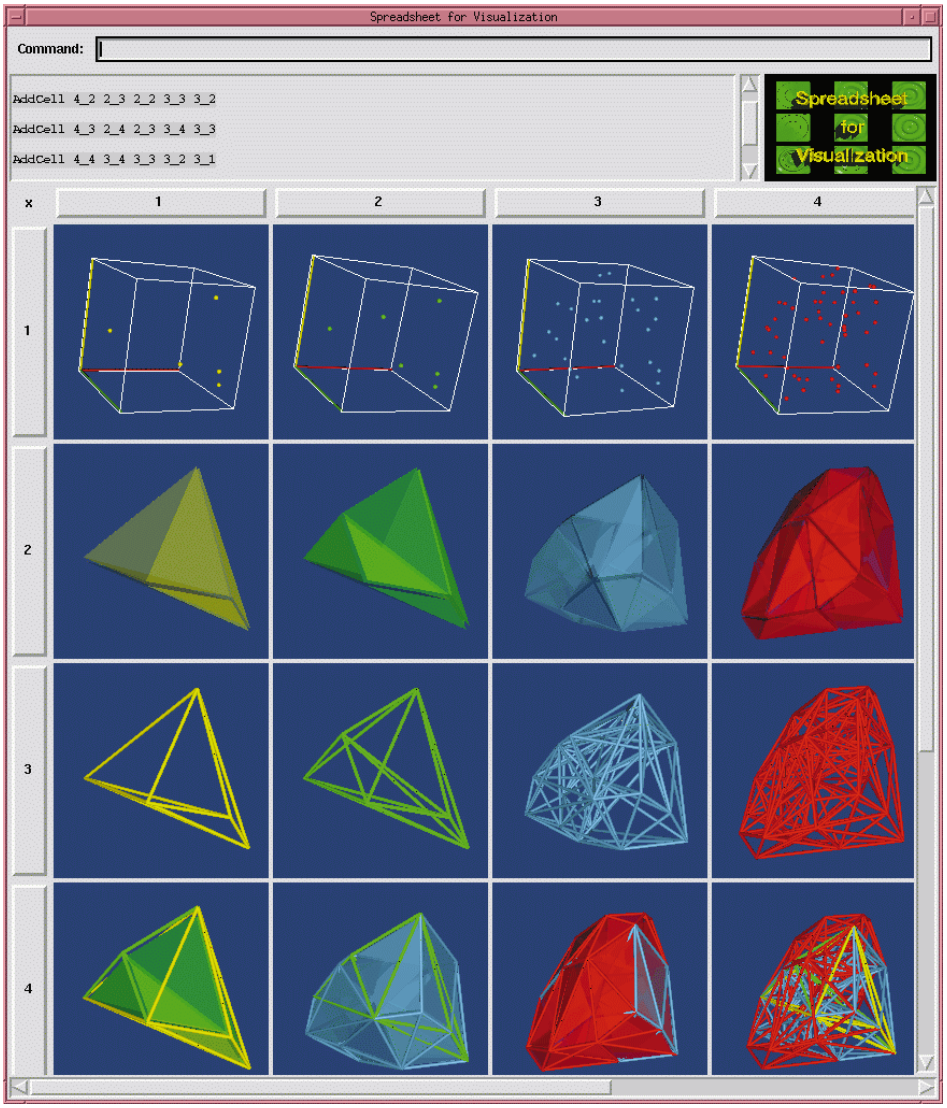
We will demonstrate the principles using SIV in the context of three data domains—molecular biology, time-series matrix visualization, and algorithm visualization. Each of these domains illustrates specific problems our research group encountered in information visualization analysis tasks. Many of the insights gained in this article result directly from this multidisciplinary collaboration. By using a task-centered approach, we illustrate concretely the principles that underlie how the visualization spreadsheet enables users to solve problems in information visualization.

SIV is quite scalable and can handle any data sets importable into VTK—one of the many advantages of using an existing visualization toolkit. Each cell view can occupy its own window for finer detail, and its dependency relationships appear in a formula entry box. SIV is capable of handling 16 megabytes of terrain data points, and volume visualization of size $64 \times 64 \times 64$ voxels or larger. For example, molecular biologists—end users in our initial design evaluation—have used SIV on sequence similarity data sets as large as several hundred pages long.

Derive comparison data sets

In the data exploration process, much user interaction involves applying operators to data sets. The visualization spreadsheet facilitates these interactions by letting users 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.

The spreadsheet paradigm provides a simple interface for performing value operators that derive new data sets, such as subtraction and addition. Let's illustrate using an algorithm visualization example. Figure 1 shows an algorithm visualization of 3D Delaunay triangulation, which forms tetrahedra from a set of 3D random points generated using random number generators. Even though well studied, the problem of 3D triangulation remains quite non-intuitive for many people. Traditional algorithm visualization techniques



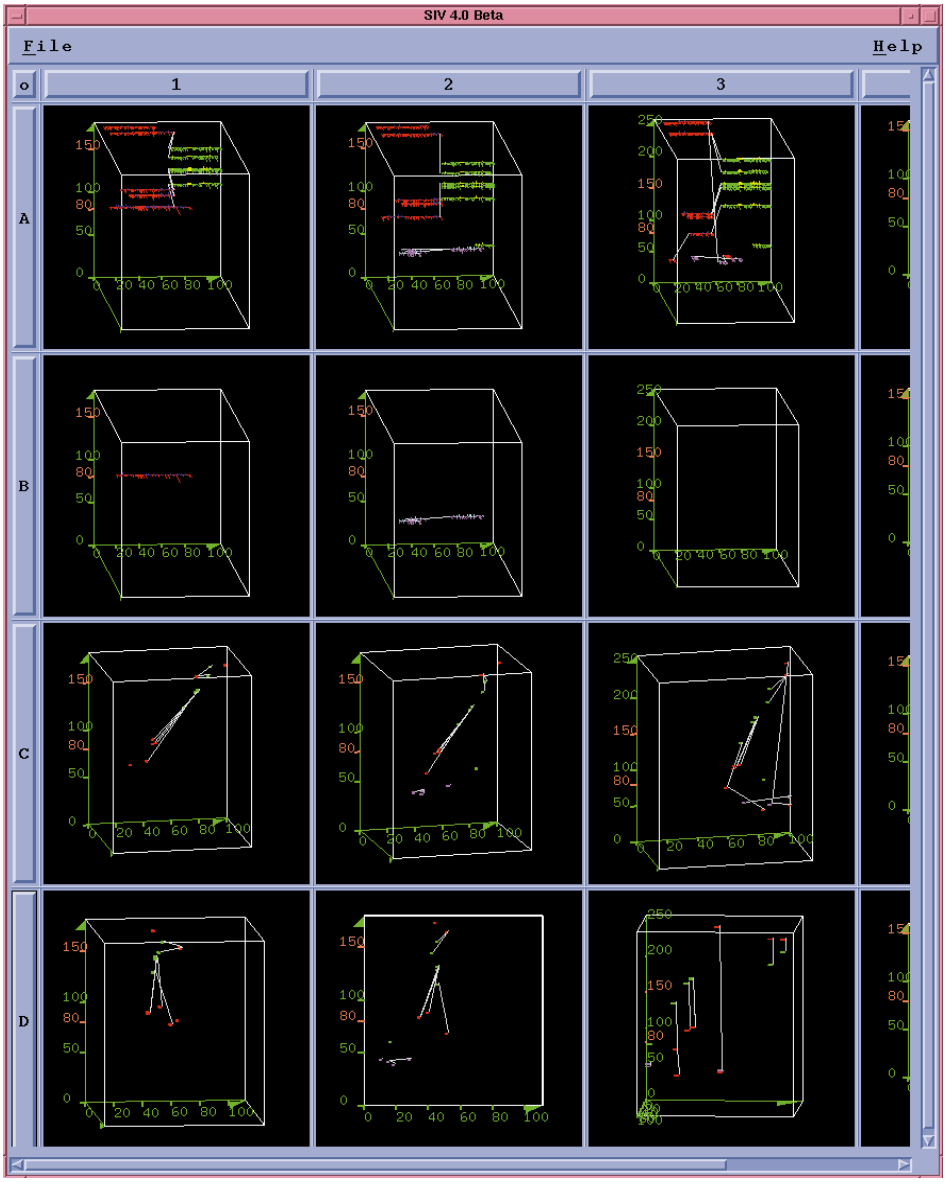
1 **Visualization of 3D random-point generation and Delaunay triangulation of the resulting point set. The columns visualize the algorithm's outcome after 5, 6, 25, and 50 steps, respectively. The last row shows the result of several addition operations (the formula syntax is "command result *operands*"):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

use animation and sequential layouts to show successive steps in order to gain better insights. Here the columns show the algorithm's results 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 transparent tetrahedra after performing 3D Delaunay triangulation. Row 3 represents the tetrahedra using edges between vertices.

By adding the geometric contents of cells together, the user can aggregate visualizations to create new representations. The last row (Row 4) aggregates several cells to form new visualizations that show differences between successive steps. Cell $C_{4,1}$ shows the difference between step 5 and 6, whereas $C_{4,2}$ 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 $C_{4,3}$, we see that after 25 steps the convex hull is completely embedded inside the convex hull obtained after 50 steps. Since we know that adding points to the triangulation can only increase the size of the convex hull, this discovery makes sense. We see the blue surfaces and vertices where the convex hull has not changed. Cell $C_{4,4}$ shows the aggregate of adding all the stick models in Row 3 together. These representations arise after many iterations of trying different combinations of the points, sticks, and surface representations of the data in Rows 1, 2, and 3.

Interestingly, these algebraic operations can take on different semantics at multiple levels. At the low level, we can capture the cell images and perform image subtractions by subtracting corresponding pixels. At the mid level, as shown in the above algorithm visualization example, we can perform geometric object algebraic operations. We can define objects and algebraically add them to or subtract them from the scene. At the high level, we can perform algebraic operations based on the particular data domain semantics.

We encountered the need to examine domain semantics for operators in a domain study with molecular biologists exploring DNA sequences. Such studies often compare a given sequence against a database of known sequences, generating thousand-page reports of possible similar regions (alignments) and other information useful to biologists. Based on AlignmentViewer, a previous visualization system we built for this data,[6] we



2 A screen snapshot of visualizing sequence similarity reports after performing three operations. Step 1: Initially, we loaded each column with a slightly different, but related, data set (*A*1 = *B*1 = *C*1 = *D*1, *A*2 = *B*2 = *C*2 = *D*2, *A*3 = *B*3 = *C*3 = *D*3). Step 2: We selected Row *B*, then subtracted cell *A*3 from it (*B*1 = *B*1 – *A*3, *B*2 = *B*2 – *A*3, *B*3 = *B*3 – *A*3). Cell *B*3 contains the empty set as expected. Step 3: We changed Rows *C* and *D* to show different views of Row *A*. The views show different sets of variables using a different representation, thus increasing our ability to see other dimensions of the multivariate data sets simultaneously.

constructed a spreadsheet for the research task of comparing similarity reports. The basic 3D visual representation consists of comb-like glyphs that show the alignments, their similarity, and where they occur along the input sequence. For example, see cell $A1$ in Figure 2. This spreadsheet, built with OpenGL and Motif using C++, includes a computational steering environment for rapidly executing the similarity algorithm on multiprocessor machines. For analysis, it provides animation, filtering, and variable-to-axis mapping capabilities.

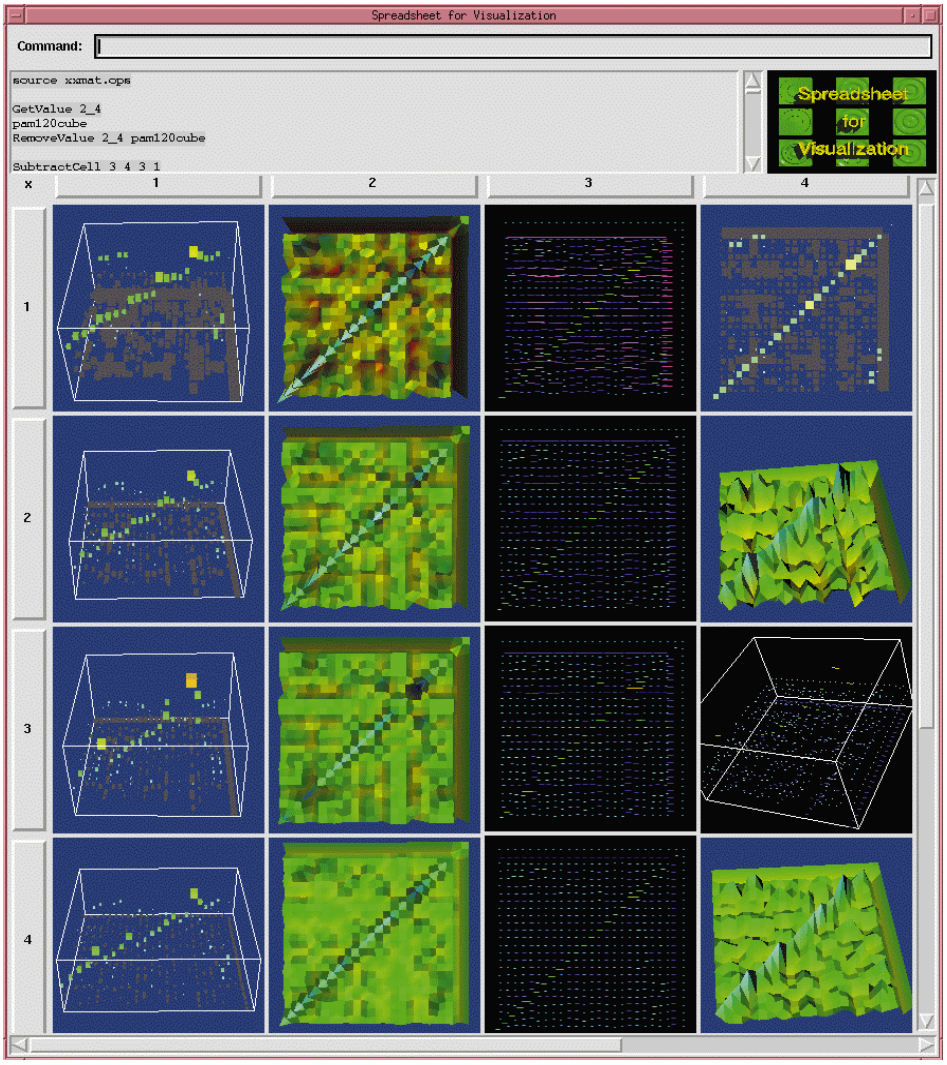
Molecular biologists want to locate differences between several algorithm runs with different algorithmic parameters. Figure 2 shows a snapshot of an example session resulting from a three-step analysis:

Step 1: We loaded each column with data sets generated from the same input sequence by varying a parameter used to specify the algorithm's sensitivity with respect to distantly related versus closely related sequences. We decreased the distance from far to near in columns 1, 2, and 3, respectively.

Step 2: We selected Row B and then subtract cell $A3$ from each cell in that row. Thus, $B1 = B1 - A3$, $B2 = B2 - A3$, $B3 = B3 - A3$. Cell $B3$ contains the empty set, as expected. The cell values are alignment sets, and we defined two alignments as equal if they share a region. Cells $B1$ and $B2$ show alignments found by using far evolutionary distance parameters, but not by the near used in $A3$.

Step 3: At this point, cells in Row C and D still contain the same data sets as the corresponding cells in Row A. We changed the variable-to-axis mapping, resulting in different views of the data sets.

Within the domain-specific semantic level, sometimes several possible definitions exist for the operator. For example, the difference operator above is only one of the three possible interpretations. We can actually



3 Visualization of time-series matrices. 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.

define three different types of equality between alignments, resulting in three difference operators.[3] Likewise, high-level algebraic operations in other domains should rely on the specific semantics of those domains.

With many data domains, the comparison operations are set algebraic rather than numeric. For example, instead of having negative numbers, we have the existence of set membership. An additional operator is set intersection. For instance, A – B creates a new set of items in A except those that are also in B. In a numeric spreadsheet, negativity is often represented using a negative sign, coloring the item red, or putting the number inside of parentheses. For a task-specific operation in the visualization spreadsheet, we can define visibility, colors, or special icons to represent these different set memberships.

The ability to generate comparison data sets proves important in exploring the differences between related data sets. If we know the domain semantics, we can apply this spreadsheet principle to let users algebraically explore differences between data sets. The addition and subtraction operation shown here typify the case of comparing two similar, but not identical, data sets—something of interest to researchers in many fields. The spreadsheet approach makes such algebraic manipulations straightforward.

Apply operators in parallel

One common, but equally important, interaction applies direct manipulation operations such as rotation, translation, and zooming. In a spreadsheet environment, often we want to apply the same operation to multiple cells simultaneously. We have found this feature extremely useful for comparison tasks. For instance, the user can select the first row in Figure 1, then perform rotations simultaneously on all the cells in that row, giving a rotationally coordinated view of the data. Scatter plots in the same orientations provide correspondence between the points in different cells. In general, we have found the end user's parallel application of operators across cells extremely useful.

Besides algebraic operators and simple scene operations, we have found other operations—such as animation and dynamic query filtering—useful under this principle. For example, by selecting a column of cells, the user can apply an animation operation to those cells simultaneously. Or the user can apply a data filtering operator to a row of data to cut out unwanted data

points. As a concrete example, in Step 1 of our sequence similarity example in Figure 2, we load the data sets by first clicking on the column button to select a column, then applying a load-dataset operator to all the cells in that column. In Step 2, we subtract Cell *A*3 from Row *B* by first selecting Row *B* and then applying a subtraction operator to all the cells in that row.

Distributing a single operation across a group of data sets is a common interaction in data exploration. We speed up users' tasks by automating the chore of applying operations to a large number of cells.

View multiple features simultaneously

For a given data type, we can often choose from many different visual representation techniques. Often, a technique contributes to finding one visual feature, while another visually extracts a different visual feature. Fortunately, the spreadsheet environment assists in organizing and displaying various visual representations. Because our system easily extends to handle new techniques via command modules, it lets us quickly experiment with and compare several representation techniques. Here we illustrate this flexibility in all three data domains.

The algorithm visualization of Figure 1 shows several different visual representations of a 3D Delaunay triangulation. Row 1 represents the point set as 3D scatter plots, showing the spread of points quite well. Row 2 shows the same data using transparent tetrahedra after 3D Delaunay triangulating on the point sets. Through interactive rotation, this representation gives a better view of the points' relative placement. 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 triangulation's interior structure.

Our sequence similarity spreadsheet also permits changing the visual representation via a mapping tool. In Figure 2, the cells in Row *C* and *D* contain the same data sets as the corresponding cells in Row *A*, but we changed the mapping in Row *C* and *D* 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 improves a user's ability to see more variables simultaneously. In this spreadsheet, a click-and-point interface controls the operations. The user loads the columns with data one column at a time and changes the data mapping of each row using the mapping tool dialog box. We implemented the mapping tool as a pull-down menu for each axis.

Exploring multiple features is also important in the domain of time-series matrices. Two major difficulties arise in dealing with time-series matrices: identifying differences in the matrix values between successive matrices, and finding an easy way to view and explore simultaneously the different features extracted by different representations. For example, the "cityscape" representation shows the matrix values as 3D blocks, whereas the "heatmap" representation shows the values as colored tiles.[7] Fortunately, the spreadsheet environment deals well with these difficulties.

We encountered two matrix series in trying to solve problems with molecular biologists studying evolution's effect on genetic sequences. Evolution accepts certain substitutions of one amino acid by another. PAM (Point-Accepted Mutations) and Blosum (Blocks Substitution Matrix) are two matrix series with each matrix representing substitution probabilities at a given evolutionary distance.[8] An element M_{ij} of a matrix specifies the relative probability of substituting the amino acid *i* for *j* 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. These matrix series' detailed nature encodes a large amount of information.[8] For example, biologists use these matrices in the calculation of similarity between sequences. Biologists want to understand the nature of these matrices because of their mathematical and biological complexity.

We used SIV to gain a better understanding of PAM and Blosum, which were calculated from different sets of information sources. To understand the differences between the matrices requires visually comparing a number of different matrices simultaneously. In Figure 3, the first, second, third, and fourth rows of cells visualize the PAM40, PAM120, PAM250, and Blosum62 matrix, respectively. We found being able to quickly bring in data and lay them out in different ways extremely useful. For example, the last row shows the Blosum62 matrix after 7 lines of commands.

By constructing several modules for different visual representations of matrices, we used our spreadsheet to answer specific scientific questions on these amino acid substitution time-series matrices. In Figure 3, the tabular layout shows different visual representations in different columns. The values in the cells are the same across each row, but we varied the visual representation to bring out different features of the data set.

We discovered several novel patterns in these matrices. The first column uses a cube representation that maps positive matrix values to the volume, height, and color attributes of the cubes. This representation 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 second column uses a "carpet plot" that maps values to the height and color of a 3D surface (using a rainbow color map with any negative entry mapped to red). The carpet plot technique shows that the matrices have different ranges of values (the colors get brighter and brighter from top to bottom).

The third column uses a bar-plot representation that maps values to the bars' length, height, and color attributes. The bar-plot technique makes comparing a specific entry from matrix to matrix easy and shows the overall decreasing trend of most off-diagonal entries. The fourth column shows various representations in different rotational orientations.

By vertically scanning the spreadsheet, the user can detect differences between matrices quickly. As we can

Further Reading in Spreadsheet-based Interactions

People have long used tables to organize information. The spreadsheet extends the tabular organization of information naturally by letting the user specify and interact with the contents of the cells and the interconnections between the cells. The spreadsheet paradigm has been suggested in earlier works for domains such as images, volume visualization, and financial data. Here we suggest further reading 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.[1]

Statisticians have examined visualizing higher dimensional point sets by a table of projections. For example, one multivariate analysis tool, the scatter matrix, is a table of scatter plots.[2] Visualization researchers have applied similar ideas, but in different ways, to produce a table of views of a single data set.[3,4] 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 they include some interactivity, such as rotation, translation, and zooming.

Spreadsheets for images

The first spreadsheet that allows displaying images in a cell is the Analytic Spreadsheet Package (ASP).[5] Levoy's Spreadsheets for Images system[6] and Hasler et. al.'s IISS system[7] examine ways to profitably extend the spreadsheet paradigm to images (as well as to other data sets—Levoy briefly mentions 3D volumes). For example, Levoy shows how a spreadsheet can be used to examine an image processing pipeline, and Hasler shows that 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 building applications rapidly. Their success attests to the usefulness of modular, easy-to-use, extensible tools for visualization tasks. Examples of such systems include ConMan,[8] AVS,[9] Iris Explorer (http://www.nag.co.uk/welcome%5Fiec.html), IBM Data Explorer (http://www.almaden.ibm.com/dx/), and Visualization Toolkit (VTK).[10]

Visual interactive spreadsheets

Past work in the visualization community has produced interactive tables for specific applications, including systems such as TableLens,[11] Focus,[12] and a graphical financial spreadsheet called Finesse.[13] TableLens is designed for browsing tabular numerical information represented using bar graphs. The Focus

see from all the columns, the diagonals of these matrices have strong values—which makes sense, since evolution favors the identity substitution (no mutation). From the second column we see that the matrices differ substantially because the colors get brighter and brighter from top to bottom. The last row shows the Blosum62 matrix, and we see its values clearly differ from any of the PAM matrices shown.

Propagating view changes in parallel to multiple cells proved highly valuable in this data analysis situation. By selecting a row, we can compare the various visual representations in the same orientation. Alternatively, we can select a column and compare different matrices using the same visual representation.

Our experience shows that the spreadsheet's elegant organization allows interesting combinations of 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.

Create analysis templates

The spreadsheet lets users create templates to reliably repeat often-needed computations without redevelopment or coding. This advantage—evident in numerical spreadsheets—translates easily into visualization spreadsheets. Users can construct their own layouts in situations that programmers cannot foresee and reuse them over and over again. This single easily understood, easily configured tool can handle multiple situations. Users, already familiar with tables, can immediately start organizing their data in this spreadsheet metaphor. For example, 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 lay out two data sets next to each other or compare two groups of

interactive table, modeled after TableLens, allows sophisticated navigation via sorting and hiding of information contained in the table.[12] Focus resembles 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 lie on fixed grids and contain four representation primitives—line plots, 3D surface plots, heat maps, or 3D bar graphs.

The NoPumpG prototype[14] 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. It is compared to a spreadsheet because of its data dependency capabilities.

References

1. P.S. Brown and J.D. Gould, "An Experimental Study of People Creating Spreadsheets," *ACM Trans. Office Information Systems*, Vol. 5, pp. 258-272, July 1987.
2. W. Cleveland and M. McGill, eds., *Dynamic Graphics for Statistics*, Wadsworth and Brooks/Cole, Belmont, Calif., 1988.
3. J. van Wijke and R. van Liere, "Hyperslice: Visualization of Scalar Functions of Many Variables," *Proc. IEEE Visualization 91*, IEEE CS Press, Los Alamitos, Calif., 1991, pp. 119-125.
4. V. Anupam et al., "Dataspace: 3D Visualizations of Large Databases," *Proc. IEEE Information Visualization Symp. 95*, IEEE CS Press, Los Alamitos, Calif., 1995, pp.82-88 and 144-145, 1995.
5. K.W. Piersol, "Object-Oriented Spreadsheets: The Analytic Spreadsheet Package," *Proc. Conf. on Object-Oriented Programming Systems, Languages, and Applications* (*Sigplan Notices*, Vol. 21, No. 11), N. Meyrowitz, ed., ACM Press, New York, 1986, pp. 385-390.
6. M. Levoy, "Spreadsheets for Images," *Proc. Siggraph 94*, A. Glassner, ed., ACM Press, New York, 1994, pp. 139-146.
7. A.F. Hasler, K. Palaniappan, and M. Manyin, "*A High Performance Interactive Image Spreadsheet (IISS)*," *Computers in Physics*, Vol. 8, May/June 1994, pp. 325-342.
8. P.E. Haeberli, "ConMan: A Visual Programming Language for Interactive Graphics," *Computer Graphics* (Proc. Siggraph 88), J. Dill, ed., ACM Press, New York, 1988, pp. 103-111.
9. C. Upson et al., "The Application Visualization System: A Computational Environment for Scientific Visualization," *IEEE Computer Graphics and Applications*, July 1989, pp.30-42.
10. W.J. Schroeder, K.M. Martin, and W.E. Lorensen, *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*, Prentice Hall, Englewood Cliffs, N.J., 1996.
11. R. Rao and S.K. Card, "The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus + Context Visualization for Tabular Information," *Proc. ACM CHI 94 Conf. on Human Factors in Computing Systems*, (Vol. 1, Information Visualization), ACM Press, New York, 1994, pp. 318-322.
12. M. Spenke, C. Beilken, and T. Berlage, "Focus: The Interactive Table for Product Comparison and Selection," *Proc. ACM Siggraph Symp. on User Interface Software and Technology*, ACM Press, New York, 1996, pp. 41-50.
13. A. Varshney and A. Kaufman, "Finesse: A Financial Information Spreadsheet," *Proc. IEEE Information Visualization Symp. 96*, IEEE CS Press, Los Alamitos, Calif., 1996, pp. 70-71.
14. N. Wilde and C. Lewis, "Spreadsheet-based Interactive Graphics: From Prototype to Tool," *Proc. ACM CHI 90 Conf. on Human Factors in Computing Systems* (Application Areas), ACM Press, New York, 1990, pp. 153-159.

data using adjacent columns. This flexibility contributed to the numerical spreadsheets' success.

For example, in Figure 2 the user set up an organization that enables the immediate detection of differences between different but related data sets. For example, even viewers without molecular biology training can see the similarity in the data sets' general structure, but also that some alignments present in cells *A*2 and *A*3 do not appear in *A*1. Users can now take advantage of their visual comparison abilities to detect differences between data sets.

As another example, the table's columns and rows increase the number of dimensions we can see simultaneously. In Figure 1, the columns show several snapshots of the 3D Delaunay algorithm's steps. So in this case, the columns represent the time dimension. With the same analysis template the user can analyze several different runs of the algorithm, examining a different random-point generator each time.

Figure 3 demonstrates an analysis template of different visual representations set up for visualizing a series of matrices. Simply applying other matrix values to the cells enables multiple analysis. Configuring the spreadsheet lets us see how templates can adapt to a wide variety of tasks, such as showing the time dimension, different data sets, or different visual representations.

As these examples show, the tabular layout's flexibility lets users construct different analysis templates for different tasks and thus contributes to the power of spreadsheet-based environments. Spreadsheets are familiar, flexible, easily configurable, and excellent for interactive comparison tasks. Coupled with simple, direct, manipulation operations applied in parallel, we see how users can tailor the spreadsheet to individual situations on-the-fly.

Conclusion

Visualization research spans a remarkable range of scientific disciplines and corresponding visualization

techniques, with certain operations needed across the entire range. These operations include visually comparing visualizations of two different data sets and performing algebraic operations on two or more visualizations, such as visualizing the difference between two data sets. Furthermore, the need to explore multiple visual representations simultaneously arises in information visualization in particular because different techniques often extract different visual features.

Over the past year we learned that a spreadsheet approach to visualization provides a powerful and intuitive technique for addressing these interaction issues. The principles discussed in this article apply across a wide range of visualization applications, helping spreadsheet users understand how to take advantage of the power of the paradigm and assisting spreadsheet developers understand how to structure their tools.

You should not use a visualization spreadsheet in certain situations, however. First, if a single view of the data suffices in a particular application domain, then a task-specific visualization program should be used. Second, a traditional data-flow system works wonderfully when one single view suffices and the ability to specify complex operations using a point-and-click interface is important. The spreadsheet emphasizes the intermediate operands, while the data-flow systems tend to emphasize the operators needed to achieve a desired result.

Future work includes better understanding how to make the framework flexible for easy extension into many application domains and dealing with screen real-estate issues by employing multiple fixed-head monitors or a large-screen device like the PowerWall. We intend to involve more end users in the evaluation process to better understand the spreadsheet design paradigm. ■

Acknowledgments

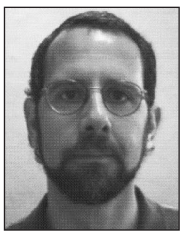
This work has been supported in part by the National Science Foundation under grants BIR 9402380 and CDA 9414015. We wish to thank the reviewers and the members of the genomic database group at the University of Minnesota for their advice and suggestions.

References

1. M. Levoy, "Spreadsheet for Images," *Computer Graphics* (Proc. Siggraph 94), Vol. 28, No. 4, ACM Press, New York, 1994, pp. 139-146.
2. A.F. Hasler, K. Palaniappan, and M. Manyin, "A High-Performance Interactive Image Spreadsheet (IISS)," *Computers Physics*, Vol. 8, No. 3, May/June 1994, pp. 325-342.
3. E.H. Chi et al., "A Spreadsheet Approach to Information Visualization," *Proc. Information Visualization Symp. 97*, IEEE CS Press, Los Alamitos, Calif., 1997, pp. 17-24, 116.
4. A. Varshney and A. Kaufman, "Finesse: A Financial Information Spreadsheet," *IEEE Information Visualization Symp. 1996*, IEEE CS Press, Los Alamitos, Calif., 1996, pp. 70-71, 125.
5. W.J. Schroeder, K.M. Martin, and W.E. Lorensen, *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*, Prentice Hall, Englewood Cliffs, N.J., 1996.
6. E.H. Chi et al., "Flexible Information Visualization of Multivariate Data from Biological Sequence Similarity Searches," *IEEE Visualization 96*, ACM Press, New York, Calif., 1996, pp. 133-140, 477.
7. E. Tufte, *The Visual Display of Quantitative Information*, Graphics Press, Cheshire, Conn., 1992.
8. S. Henikoff and J. Henikoff, "Performance Evaluation of Amino Acid Substitution Matrices," *Proteins: Structure, Function, and Genetics,* Vol. 17, 1993, pp. 49-61.

Ed H. Chi *is a PhD candidate in computer science at the University of Minnesota, where he works on information visualization. His area of expertise is software systems for visualization, user interfaces, and computer-human interaction. He received his BS in 1994 and MS in 1996 in computer science from the University of Minnesota. He expects to receive his PhD in late 1998.*

John Riedl *is an associate professor in computer science at the University of Minnesota. His research interests include collaborative systems, distributed database systems, and scientific visualization. He received a BS degree in mathematics from the University of Notre Dame in 1983 and MS and PhD degrees in computer science from Purdue University in 1985 and 1990, respectively.*



Phillip Barry *has been in the Computer Science and Engineering Department at the University of Minnesota since 1989. His areas of expertise are computer-aided geometric design, scientific visualization, and computer graphics. He received his BS and MS from Idaho State University and his PhD from the University of Utah.*

Joseph A. Konstan *is an assistant professor of computer science and engineering at the University of Minnesota. His area of expertise and research is software systems for human-computer interaction, including multimedia systems for flexible presentation, scientific visualization, collaborative filtering, and constraint-based programming tools. He has a PhD in computer science from the University of California at Berkeley.*

Readers may contact Chi at the Dept. of Computer Science, University of Minnesota, 4-192 EE/CS Bldg., Minneapolis, MN 55455, e-mail echi@cs.umn.edu.