The Information Mural: A Technique for Displaying and Navigating Large Information Spaces

Dean F. Jerding and John T. Stasko

Abstract—Information visualizations must allow users to browse information spaces and focus quickly on items of interest. Being able to see some representation of the entire information space provides an initial gestalt overview and gives context to support browsing and search tasks. However, the limited number of pixels on the screen constrain the information bandwidth and make it difficult to completely display large information spaces. The *Information Mural* is a two-dimensional, reduced representation of an entire information space that fits entirely within a display window or screen. The Mural creates a miniature version of the information space using visual attributes, such as gray-scale shading, intensity, color, and pixel size, along with antialiased compression techniques. Information Murals can be used as stand-alone visualizations or in global navigational views. We have built several prototypes to demonstrate the use of Information Murals in visualization applications; subject matter for these views includes computer software, scientific data, text documents, and geographic information.

Index Terms—Information visualization, software visualization, data visualization, focus+context, navigation, browsers.

1 Information Murals

ALTHOUGH large quantities of information are becoming available on-line, the information itself is useless without effective display and access mechanisms. Information visualizations can utilize visual and audible channels to convey information to the observer. The visual channels include attributes such as size, shape, color, intensity, texture, font, etc. Independent of the visual channels used, visual bandwidth is limited by the number and size of pixels on the screen.

The design of a particular information visualization is very much dependent on the task(s) it is intended to support. Plaisant et al. have categorized different types of tasks, including image generation, open-ended exploration, diagnostic, navigation, and monitoring [27]. For many of these applications, particularly within the context of a browser, a global view of the information is important as a navigational aid or as an analysis tool. Global views are used to provide context for more detailed views, to help formulate a search, identify patterns, or make a gestalt overview.

As the information visualization field matures, visualizations must scale to larger and more complex information spaces. Different visualization techniques have been proposed to increase the amount of information that can be displayed on the screen at the same time, both to create global views and to portray focus and context simultaneously. However, all visualizations must be created using the limited number of pixels on the screen; this often severely constrains a designer's ability to create global overviews of large information spaces.

Our *Information Mural* technique allows 2D visual representations of large information spaces to be created even when the number of informational elements greatly outnumbers the available pixels. Current methods for depicting such large information spaces, discussed in more detail later in this paper, typically utilize abstraction, aggregation, overplotting, or sampling to create a view of the entire space. Or, scrollbars are used to allow access to different parts of the information. All of these techniques can result in a loss of information that might be useful to the observer.

An Information Mural is a 2D, miniature representation of an entire information space that uses visual attributes, such as color and intensity, along with an antialiasing-like compression technique, to portray attributes and density of information. The goals of our technique can be summarized as follows:

- Create a representation of an entire (large) information space that fits completely within a display window or screen.
- Mimic what the original visual representation of the information would look like if it could be viewed in its entirety, i.e., containing the same visual patterns.
- Minimize the loss of information in the compressed view, regardless of the size of the compressed representation.

The primary motivation for the development of Information Murals was to create global overviews for scalable software visualizations. However, there are several different types of information spaces that can be represented using Information Murals:

• *The authors are with the Graphics, Visualization, and Usability Center, College of Computing, Georgia Institute of Technology, Atlanta, GA 30332-0280. E-mail: stasko@cc.gatech.edu, Dean.Jerding@sciatl.com.*

For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number 106984.

- Time-oriented visualizations often span many computer screens if laid out completely. These types of views are particularly prevalent in software visualization [28], [32] and monitoring applications.
- Visualizations that contain miniature representations of information are forced to make trade-offs in deciding what visual attributes of the information can be included at small scales.
- A text file or document usually does not fit entirely on the screen because its vertical dimension far exceeds its horizontal dimension. Displays of textual information thus often utilize scrollbars to provide navigation through a document.
- Graphs of data often require some compression technique to fit on the screen. Scaling and rounding of data values is often necessary to draw the entire graph. Other alternatives are to display a statistic, such as the average of the data values, or only a subset of the data.
- Images might be represented using Information Murals. Although an image usually fits on a screen, it is often desirable to change the size of the image. As an image is shrunk, information in the image is inevitably lost.

Information Murals allow global views of large information spaces to be constructed. Such contextual information directly supports analytical and navigational tasks that a user performs while interacting with informational displays.

The next section of this document presents a brief, high-level overview of the Information Mural technique. Following this, visualization applications that utilize Information Murals are presented, illustrating a number of different domains to which the technique can be applied. The next section then describes the different Mural algorithms in detail and explains how the Mural can be integrated as a widget in a user interface. Finally, we describe related work and how the Mural technique compares to other visualization methods. We also critique the Mural technique, assessing its relative advantages and disadvantages with respect to common tasks accomplished with the aid of visualizations.

2 Information Mural Technique

Imagine some visual representation of a large information space whose resolution is $M \times N$ pixels, much larger than the screen resolution. For simplicity, assume that it is a black and white image, and we want to display it on the screen in $X \times Y$ pixels. A simple algorithm just scales the pixels in the original image into the available space, overwriting pixels that happen to overlap. A destination pixel will look the same (be turned on) if one pixel from the original image happens to map to that screen location or if 100 or more pixels happened to fall there. This effect is known as *aliasing* [10].

The idea of the Information Mural technique is to make the density of overlap visually apparent. It draws on strategies for antialiasing in computer graphics by varying intensity of the screen pixels to convey the underlying density of pixels from the original sample. To construct an $X \times Y$ Information Mural of the original image, the position of each pixel in the $M \times N$ representation is first scaled to fit into the available space. As each source pixel is then "drawn" in the Mural using an imaginary pen, different amounts of "ink" fall into bins for each screen pixel. As each subsequent pixel from the original is drawn, the amount of ink will build up in different bins, depending on the amount of original pixels that map to the same screen pixel.

The resulting Information Mural is then created by mapping the amount of ink in each screen pixel (the information density) to some visual attribute. In a *gray-scale* Mural, the shade of each screen pixel is determined by the proportion of ink in its bin relative to the maximum amount of ink in a single bin. Thus, areas that are most dense with information are rendered the brightest (or darkest). Color can then be added to convey other attributes of the information while still preserving the density mapping. Instead of using gray-scale variation, an *equalized intensity* variation over the entire color rainbow can also be used. Details of the mapping from pixel density to the color scale, more precise descriptions of the Mural algorithms, and how a Mural can be used by applications as a widget are all discussed later in Section 4. In the next section, we present a number of examples that show how a Mural can be used for various tasks in different application domains.

3 Applications

Information Murals can be used as global views of information spaces, both for analysis purposes and for navigation. Without a good visual representation, a global view cannot serve as an effective navigation tool. Furthermore, the usefulness of a visualization tool often depends on the effectiveness of its navigation capabilities: Can the user navigate quickly to locate an area of particular interest? Used as a background in a navigational widget, Murals provide informational context to support panning and zooming of more detailed focus views. By adding panning and zooming within the global view itself, an Information Mural can be used as a stand-alone visualization.

Below are some snapshots from visualization applications we have built using Information Murals. These applications contain many different forms of information, from software to data to text documents, some of which were mentioned in [19]. The examples are broken down here by data domain. Section 4 that follows then characterizes the examples according to the fundamental *task* they are facilitating.

One of our original prototypical object-oriented views is called the *Execution Mural* (Fig. 1). This view is used to examine message traces from object-oriented programs [18], [20]. The view is similar to an event trace diagram for object-oriented message sequences rotated 90 degrees, such that classes are assigned rows on the vertical axis and a message from one class to another is drawn as a vertical line connecting the source and destination classes. The horizontal axis then represents time or the sequence of messages. Classes can be listed vertically according to their alphabetical order, by their appearance order in source files, or by viewer specification.

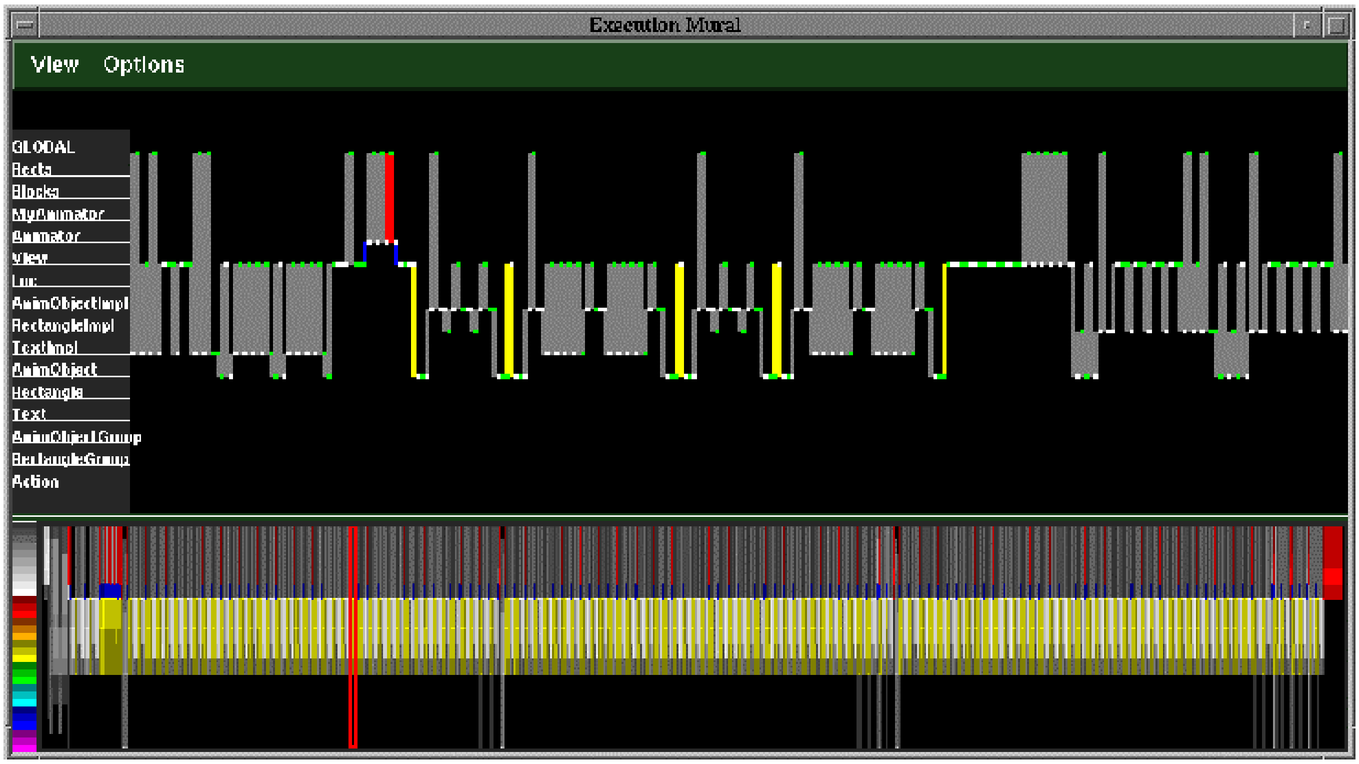
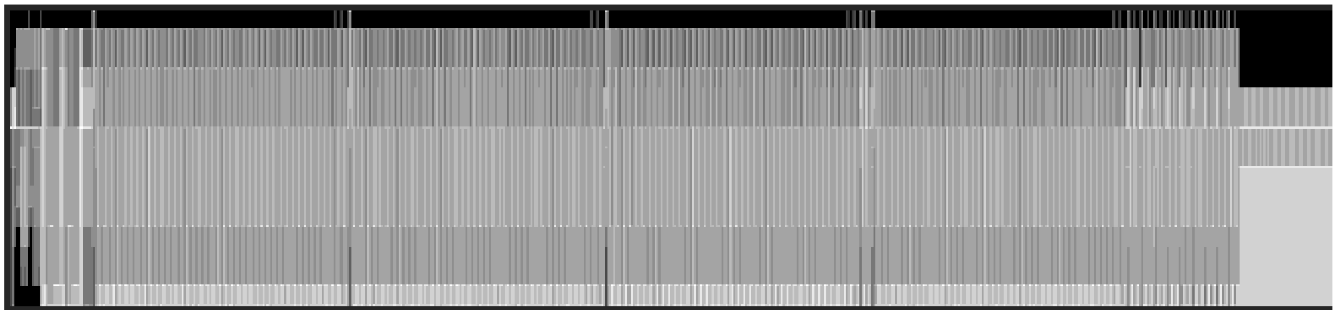
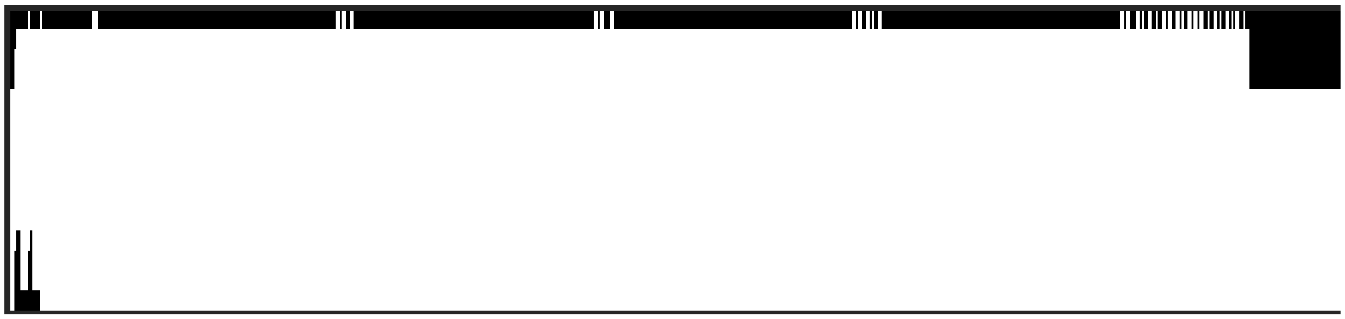


Fig. 1. Execution Mural view of bubble sort algorithm animation built using the Polka animation toolkit.



(a)



(b)

Fig. 2. (a) Mural of object-oriented message trace of over 90,000 messages, drawn in an area 500 pixels wide. (b) Same diagram drawn by just overplotting (without the Mural technique).

The upper portion of the view is the focus area where a subset of the messages can be examined in detail. The bottom portion of the view is a Mural widget: a navigational area that includes a Mural of the entire message trace and a navigation rectangle indicating where the focus area fits in the entire execution. The end-user of the Execution Mural can interact with it in a number of ways. First, the focus rectangle in the lower region can be panned to change the focus display above. Second, the user can sweep out an area in the Mural into which the Mural will be zoomed. Third,

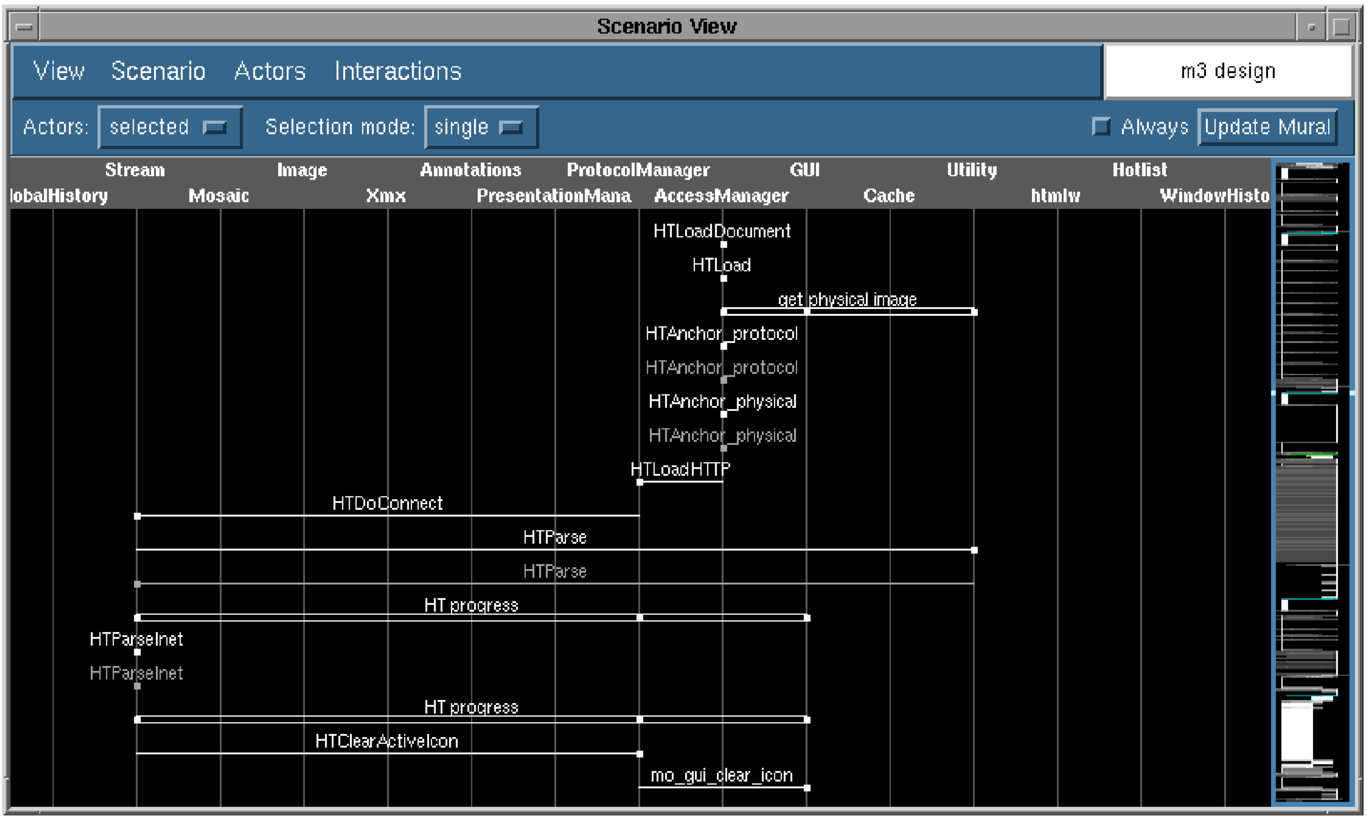


Fig. 3. ISVis Scenario View showing an execution trace of the Mosaic web browser containing 425,000 messages. Information Mural on the right acts as the global overview of the scenario.

the viewer can utilize simple brushing techniques [3] through accompanying user interface actions to selectively show or hide various gray-scale and attribute colors in the Mural. Fourth, the color and size of individual messages can be set, and classes can be selectively shown or hidden.

To compare the global overview using the Mural to an overplotted rendering of the entire image, Fig. 2a shows a gray-scale Information Mural of a message trace from a bubble sort algorithm animation built using the Polka toolkit [33], containing around 20 classes on the vertical axis and over 90,000 messages on the horizontal. Drawing this image in a window 500 pixels wide results in a horizontal information (lossy) compression ratio of over 180:1. For comparison, the same representation without the Mural technique (drawn by scaling each message to the nearest column of pixels and drawing a vertical line with the appropriate end-points) is shown in Fig. 2b.

Illustrating a program execution via the Mural allows the viewer to perceive phases and patterns in the entire execution as well as the classes participating in each phase, which can be an important factor in software analysis [23]. The message coloring in the Mural also allows the location of particular messages throughout the execution to be identified.

Being able to construct and observe global views of various message traces gave us insight into the existence of message patterns and subpatterns in object-oriented programs. Visual patterns can be seen in an entire message trace, and then lower-level patterns as we zoom in on subsequences of the execution. The visual patterns are either the result of similar semantic operations in the code or of iteration as in a `for` loop. One of the weaknesses of the Execution Mural visualization in terms of helping program understanding tasks is that a view of individual messages is really too low-level compared to a user's mental model or system design models such as interaction diagrams. The message patterns we were finding seemed to be useful abstractions to help bridge this gap. These observations motivated the work described in [20] and the subsequent development of the ISVis (Interaction Scenario Visualizer) tool, where we treat repeated sequences of messages as higher-level abstractions that correspond to design-level interaction scenarios.

A *Scenario View* from our ISVis tool is shown in Fig. 3. ISVis allows an analyst to browse and analyze scenarios derived from program execution traces. It can be used to compare expected behavior specified as design-level scenarios to actual execution traces, or to reverse engineer behavior and architecture of legacy systems. The view in Fig. 3 is from a case study where ISVis was used. For the case study, we wanted to understand which components of Mosaic must be changed or added to support user-configurable external viewers, whereby Mosaic provides users interactive control over which viewers are used for specific types of web content (MIME types).

The Scenario View is, in fact, a Temporal Message Flow Diagram (TMFD) [6], sometimes called an interaction diagram or event-trace diagram. Actors in the view are assigned columns, and interactions are drawn as lines from source to destination actor in descending time order. Actors

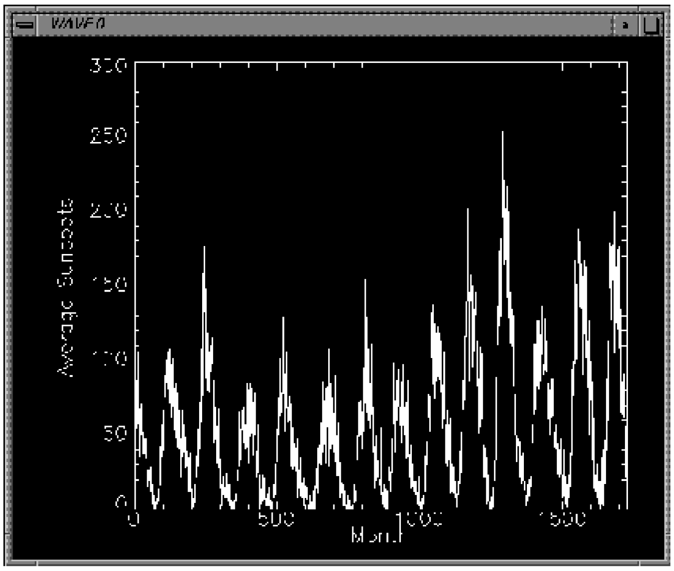


Fig. 4. Plot of average number of sun spots recorded per month, 1850-1993.

can be shown as functions, classes, files, or user-defined components. An Information Mural is used to provide a global overview of the scenario, appearing on the right of the view. Interaction with this view is described in more detail in [17].

It is possible to visually locate candidate interaction patterns and use the global overview to navigate to regions in the scenario where similar sequences of interactions occur. Once a sequence of interactions is selected using the mouse, it can be defined as an interaction pattern. Then, all occurrences of that sequence of interactions in the original scenario are replaced with a reference to the newly defined scenario. While a simple interaction is shown as a line connecting the source and destination actors, a subscenario that occurs within the Scenario View appears graphically as a rectangle containing all of the actors involved in the scenario. If this scenario is then assigned a color, the user can quickly see where and how often the subscenario occurs in the execution trace.

In the Mural at the right side of Fig. 3, four different phases in the first two-thirds of the scenario are evident, one for each HTML document visited during that execution of Mosaic. Repetitive patterns occur as each document is processed. Differences arise from the number of images in each document, another interaction pattern that we found. Early on, the analyst also discovered interaction patterns for the processing of a mouse click on an anchor, of which there are six in the scenario: three in the first two-thirds of the scenario for HTML links and three at the end for the PostScript documents displayed. More detailed results from the Mosaic case study using ISVis can be found in [17].

Many other software visualization tools utilize miniature time-line views to portray execution information, such as the HotWired visual debugger for `C++` and `Smalltalk` [24], the PV program visualization system [23], and the ParaGraph parallel program visualization system [15]. These systems utilize views which are either limited in the amount of time they can portray by the screen real-estate and must be scrolled, or in which overplotting occurs as the execution time grows larger. The Information Mural technique could be utilized to increase the amount of historical information that can be displayed without loss of information due to overplotting.

3.1 Data Visualization

The Information Mural technique is useful for revealing the underlying density of data while viewing very large data sets. Traditional plotting techniques typically overplot points that happen to lie in the same pixel, or they aggregate, average, or smooth data values. Our technique shows the actual density of the information. Incorporated into a data visualization, Murals can support one- or two-dimensional navigation through large data spaces.

In this subsection, we present an example of visualizing sun spot activity. In addition to this example, we have used Murals to visualize other data sets such as river flow or car type/mileage correlations. Much of the data that we visualized was obtained from the StatLib server at Carnegie Mellon University.

Astronomers have been recording the number of sun spots since the 1700s. Because this is such a large dataset, it is typically plotted by showing the monthly averages. Fig. 4 is a plot of the average number of sun spots per month recorded from 1850-1993.

Using the Information Mural technique, we are able to more directly depict very large data sets without averaging. Fig. 5 shows an antialiased Mural of the number of sun spots recorded daily from 1850-1993, over 52,000 readings. Instead of using gray-scale to depict density, a color scale that ranges from dark blue (lowest data density) to bright white (highest data density) is used because it is easier to see outliers using color.

The Information Mural is valuable in that it conveys data density and an overall pattern (periodicity). Another advantage is that, as opposed to averaging techniques, we can see the band of "missing" values between zero and about 10 sun spots, and we can notice that a large number of zero values was recorded (bright spots at bottom of Fig. 5).

With the interactive Information Mural views, it is also possible to incrementally zoom in on sections of the Mural or to sweep out a rectangle to zoom. Fig. 6 shows the sun spot Mural zoomed in on a small area. Fig. 7 shows how the Mural of the entire data set can be placed in the background of a slider, giving context to a more detailed view of the data.

The Mural is not a panacea, however. It does not convey other aspects of the data as well as other techniques. For example, time series data such as this is often depicted via loess curve fitting [8]. This technique better conveys the relative slopes of segments of the data. Furthermore, when the analyst simply desires to learn the relative *quantitative* measures of a data set (means, distributions, frequencies, etc.), some of the standard summary techniques are sufficient, and the graphical requirements of a Mural appear to be overkill. For a more thorough discussion of the variety of visualization techniques available for charting bivariate and trivariate data, see [8].

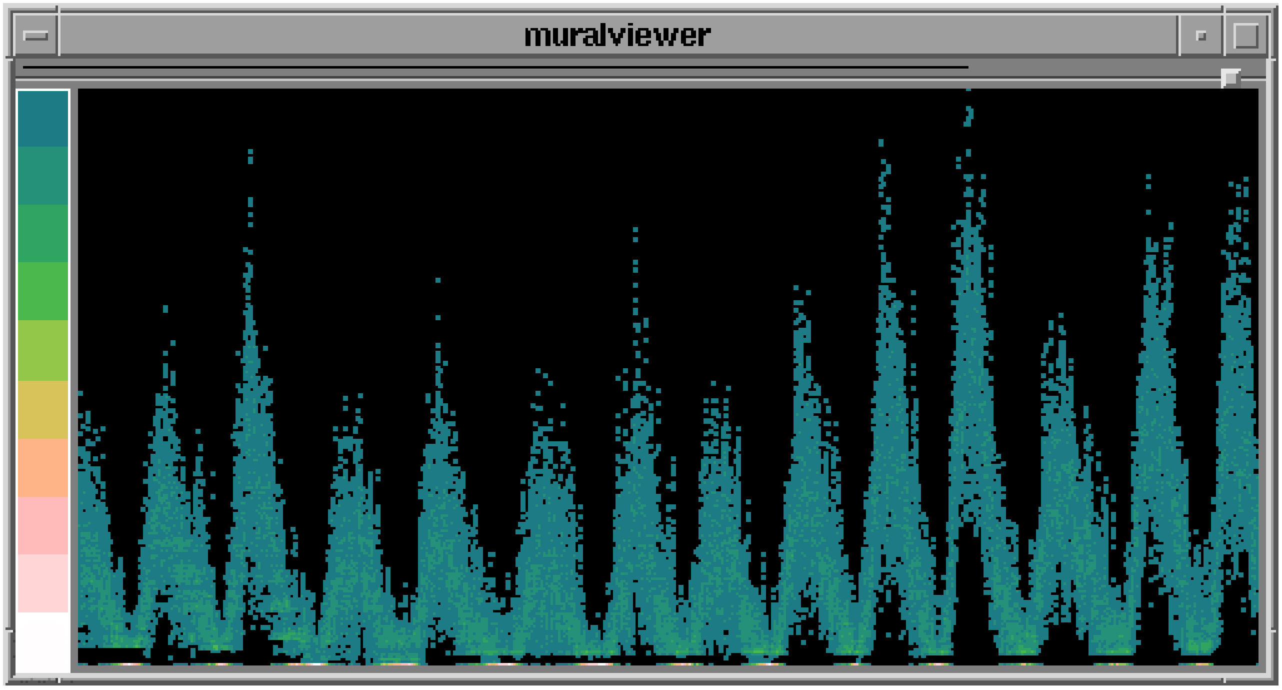


Fig. 5. Mural of the number of sun spots recorded daily, 1850-1993.

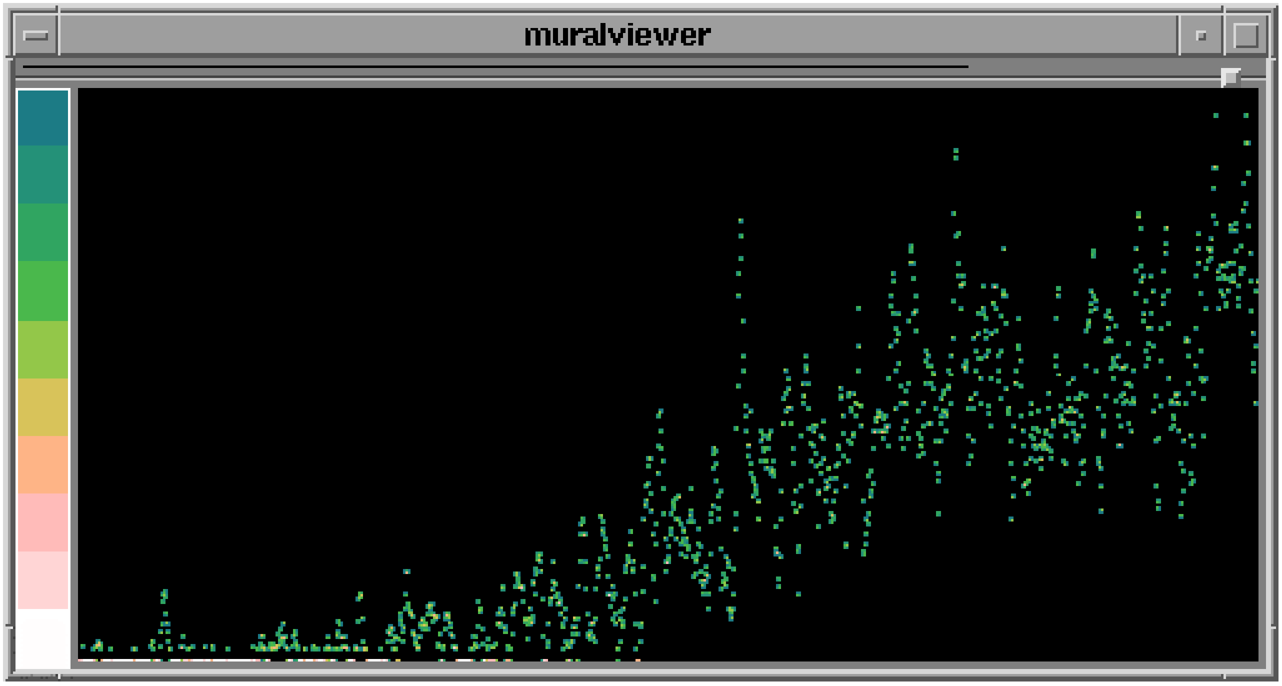


Fig. 6. Mural of the number of sun spots recorded daily, 1850-1993, zoomed in on a small area.

3.2 Information Visualization

Many other forms of information can be displayed using Information Murals. Two such applications, geographic data and text documents, are described below.

3.2.1 Geographic Information

Many organizations, such as the U.S. Census Bureau, create maps of various census statistics, such as population distributions. A common technique used to illustrate geographic data like this is the *chloropleth map* [26]. A chloropleth map breaks down a geographic area into smaller regions that are then given a gray-scale shade or color to indicate some value over that region. How to do this areal breakdown is one of the challenges of mapmaking. Is the breakdown purely spatial/geographic or is a unit, such as city, county, or state used? A second issue is how many shade or color levels are used to render the illustration, and how data are mapped to those levels.

When very large data sets are used, such as continental U.S.A. population figures, and the display region is relatively

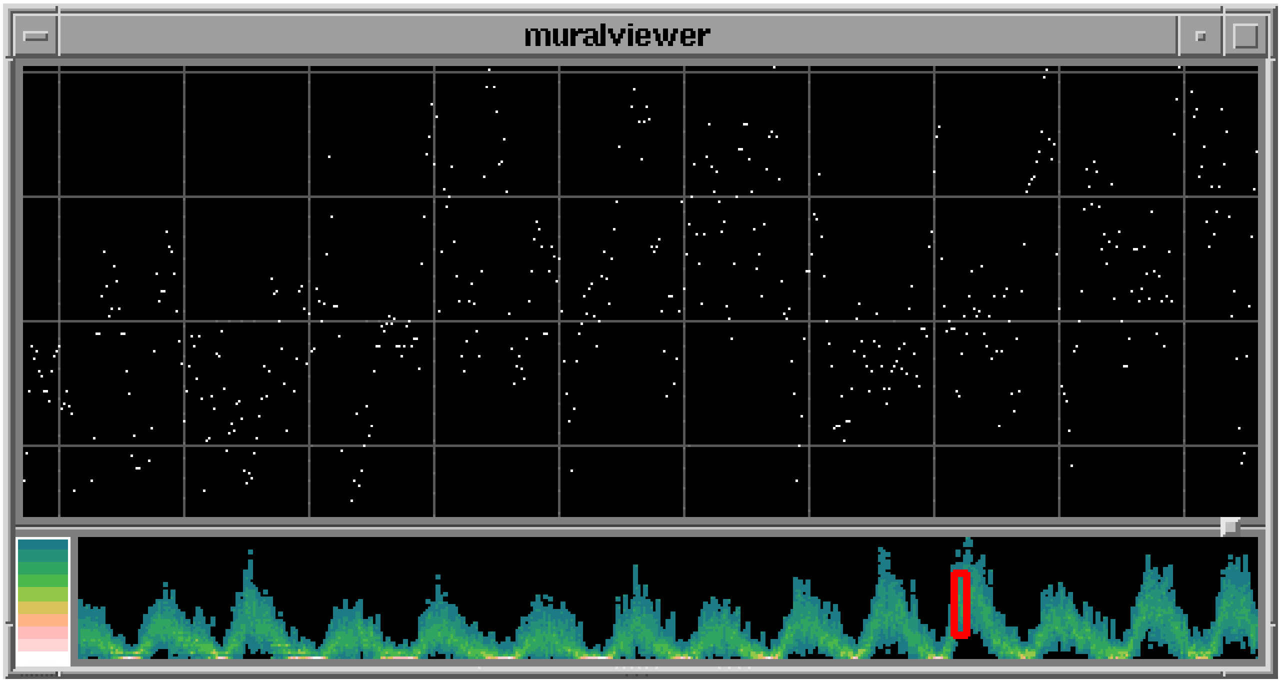


Fig. 7. View of sun spots showing focus area and Mural of entire data set at the bottom.

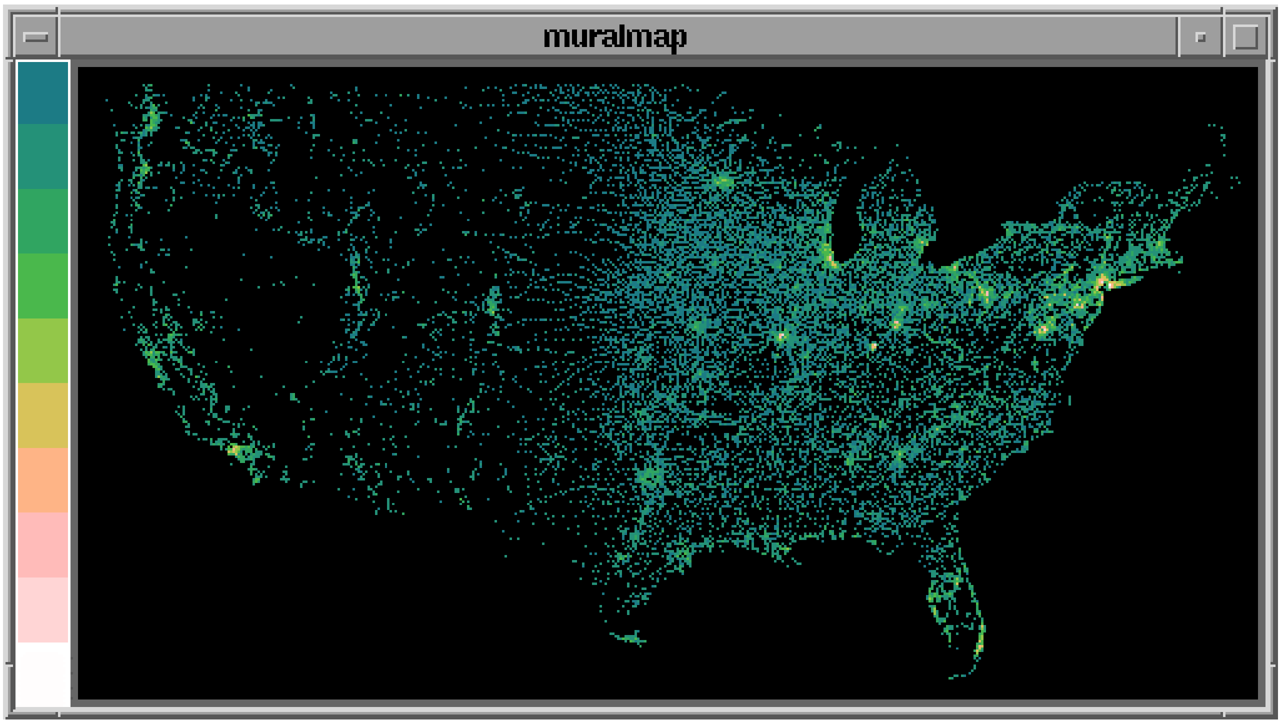
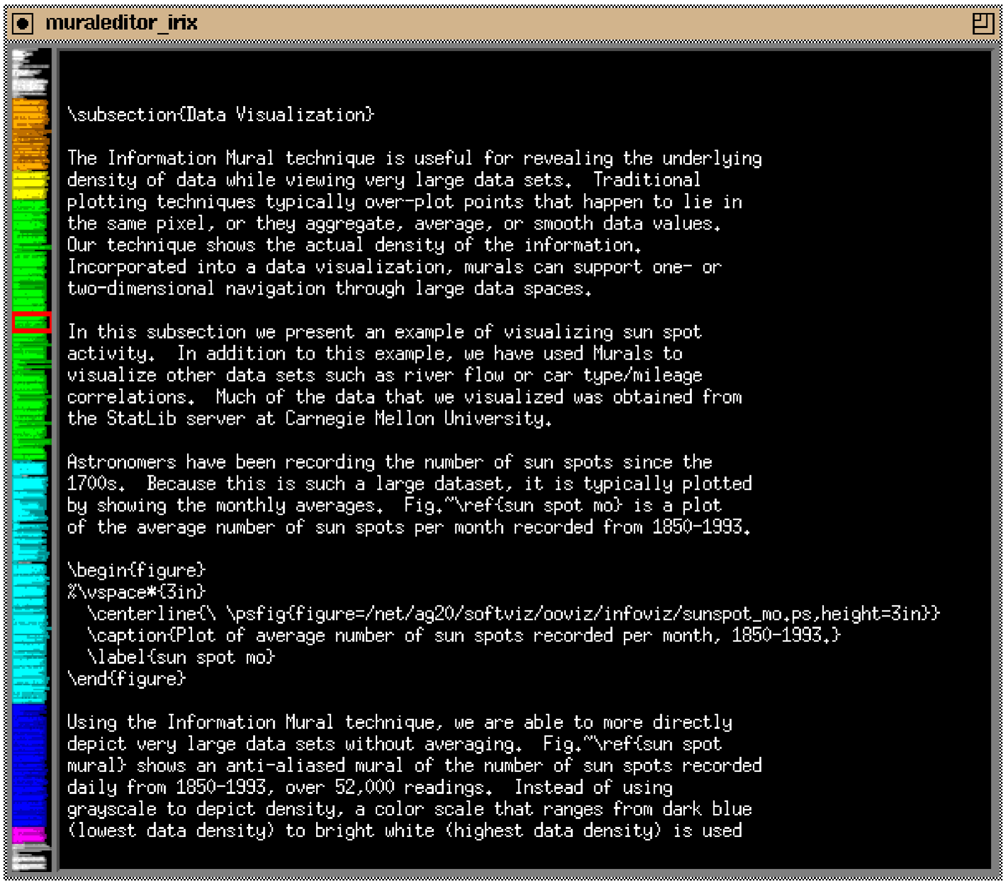


Fig. 8. Mural of population density distribution, using data from the 1990 census.

small, creating a meaningful visualization that adequately conveys population densities can be challenging. The Information Mural technique computes information density automatically, making the display of a population density map on a computer screen quite easy. Fig. 8 illustrates an Information Mural of U.S. population data taken from the Tiger Mapping Service U.S. Places File, created from the Census file STF-1A. Each data point provides the population of a census block (small rectangular region) of the country.

To generate the Mural shown here, we consider that population to be at a single geometric point, the center of the census block, and then add that contribution to the appropriate individual screen pixel (the efficient algorithm of Section 4.2). We use 10 shade values from a single hue with a logarithmic mapping of population to shade in order to provide better resolution of smaller populations.

In essence, the Mural provides a form of detailed choropleth map with an individual pixel as the area subunit.



```
\subsection{Data Visualization}

The Information Mural technique is useful for revealing the underlying
density of data while viewing very large data sets.  Traditional
plotting techniques typically over-plot points that happen to lie in
the same pixel, or they aggregate, average, or smooth data values.
Our technique shows the actual density of the information.
Incorporated into a data visualization, murals can support one- or
two-dimensional navigation through large data spaces.

In this subsection we present an example of visualizing sun spot
activity.  In addition to this example, we have used Murals to
visualize other data sets such as river flow or car type/mileage
correlations.  Much of the data that we visualized was obtained from
the StatLib server at Carnegie Mellon University.

Astronomers have been recording the number of sun spots since the
1700s.  Because this is such a large dataset, it is typically plotted
by showing the monthly averages.  Fig.~\ref{sun spot mo} is a plot
of the average number of sun spots per month recorded from 1850-1993.

\begin{figure}
%\vspace*{3in}
  \centerline{\ \psfig{figure=/net/ag20/softviz/ooviz/infoviz/sunspot_mo.ps,height=3in}}
  \caption{Plot of average number of sun spots recorded per month, 1850-1993.}
  \label{sun spot mo}
\end{figure}

Using the Information Mural technique, we are able to more directly
depict very large data sets without averaging.  Fig.~\ref{sun spot
mural} shows an anti-aliased mural of the number of sun spots recorded
daily from 1850-1993, over 52,000 readings.  Instead of using
grayscale to depict density, a color scale that ranges from dark blue
(lowest data density) to bright white (highest data density) is used
```

Fig. 9. Text editor containing LaTeX document. Mural of the entire file is shown in the background of the scrollbar, with text colored according to section.

When the data is very large, fine-grained, and organized purely geographically (as opposed to structural or political areal aggregation, such as population by county), the Information Mural appears to provide a good tool for map makers. How population values are then classified or mapped to the resulting images shades or colors is still the critical issue, however. Reference [26] describes how varying that mapping can result in markedly different presentations. Fig. 8 is simply one particular mapping we chose. It would be easy to configure a Mural so that the viewer could interactively specify classification/mapping parameters, thus experimenting with different renderings.

3.2.2 Text Documents

While SeeSoft [9] from AT&T's Bell Laboratories introduced a revolutionary miniature representation for text documents, it did have a limit. One row of pixels (or part of a row in later versions) is required for every line in the file. The Information Mural technique can go beyond this limit, allowing many lines in a file to map to a single row of pixels in the miniature representation. On top of a gray-scale Mural representation of a document, color can be used to indicate attributes of the text, such as comments, sections, or keywords.

Fig. 9 is a sample text editor with a Mural in the background of the scrollbar. Color is used to indicate sections in the LaTeX document being browsed. The Mural is constructed by examining the position of each character in the file, scaling that position into the scrollbar, and mapping the resulting density of characters to the intensity scale.

Several previous visualization systems have used the background of a scrollbar to display information about textual documents. The Edit Wear and Read Wear technique colored lines in a scrollbar to represent the reading and writing history of lines in a text file [16]. It is not clear how attributes of lines in large files would be displayed, as one attribute could occlude another. The Information Mural technique would help an application such as this display attributes for files that have more lines than there are rows of pixels in the scrollbar. Chimera's Value Bars have a similar problem when trying to display attributes of lists with more members than there are rows of pixels in the display [5].

Information Murals can also be used to visualize the distribution of keywords in a set of documents retrieved from a search. Figs. 10a, 10b, and 10c show the distribution of keywords in three papers after a search for *visualization* (yellow), *object-oriented* (green), and *OO* (cyan) was performed.

The document in Fig. 10a seems to be about visualization and talks a little about object-oriented in the beginning. Fig. 10b talks about both visualization and object-oriented throughout the document, and Fig. 10c discusses object-oriented and visualization in the beginning and in the end. Miniature views such as these could be utilized in search applications to display the results of a search and give users

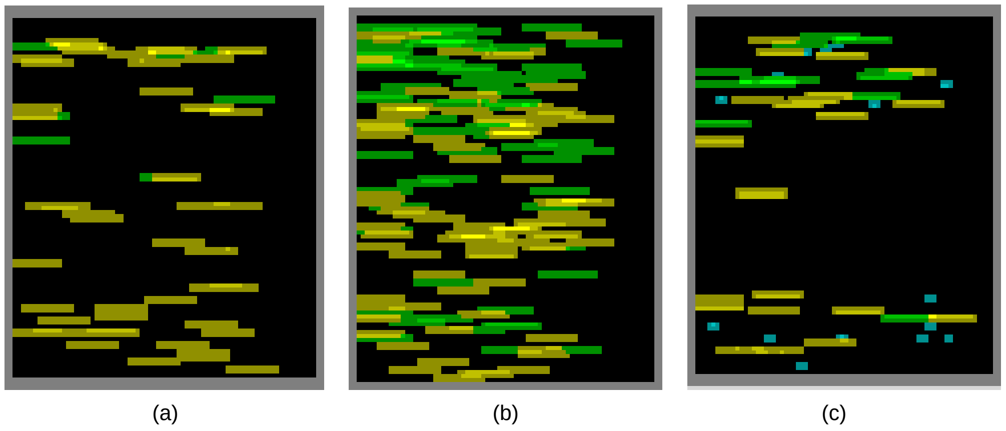


Fig. 10. Murals showing keyword distribution for search on “visualization” (yellow), “object-oriented” (green), and “OO” (cyan) in three documents.

more information about the documents retrieved. This information would aid a user in determining document relevance, in addition to a simple numerical ranking.

The TileBar visualization technique uses gray-scale tile images that correspond to a thematic breakdown of a document to visually display relevance information to a keyword search [14]. This technique is more complicated and can require more space than just visually depicting the location of the keywords using an Information Mural. It does, however, make the direct comparison of keyword locations possible across documents of different lengths.

4 Information Mural Algorithms

In Section 2, we presented a high-level discussion of how the Information Mural works. The next three subsections describe various algorithms for creating Information Murals, followed by a discussion of how the Mural is actually used by applications as a widget. The original algorithm was developed to solve a real problem in our software visualization research: how to render global overviews of lengthy time-oriented visualizations. Modifications to the original algorithm were made to improve performance and then to support attribute colors. All algorithms for creating an Information Mural take an input image at a scale of $M \times N$ pixels and render it as Mural of $X \times Y$ pixels. In addition to the data structures that store the original information, the algorithms require an $X \times Y$ array of floats. While the algorithms describe the transformation from an original image into a Mural, in the actual implementation of the Mural widget described in Section 4.4, there is not a physical original image; the client application draws points and lines via the Mural widget *at the scale of the original image* and the Mural widget translates these points and lines into a Mural of the appropriate size.

4.1 Original Algorithm

The original algorithm listed below creates an Information Mural in a manner very similar to weighted area sampling with overlapping weighting functions [10]. In this version, a pixel from the original image contributes proportionally to the intensity of the four surrounding screen pixels that it covers when scaled into the $X \times Y$ Mural. The intensity contributed to each of the surrounding pixels is computed as follows:

1) construct a unit square connecting the centers of the four surrounding destination pixels,
2) use the scaled location of the center of the original pixel to divide the square into four quadrants, and then
3) the area of the quadrant diagonally opposite to each of the four destination pixels is the amount of intensity contributed to that pixel.

Fig. 11 shows the computation for a pixel at (m, n) in the original image. Note that the algorithm does not consider

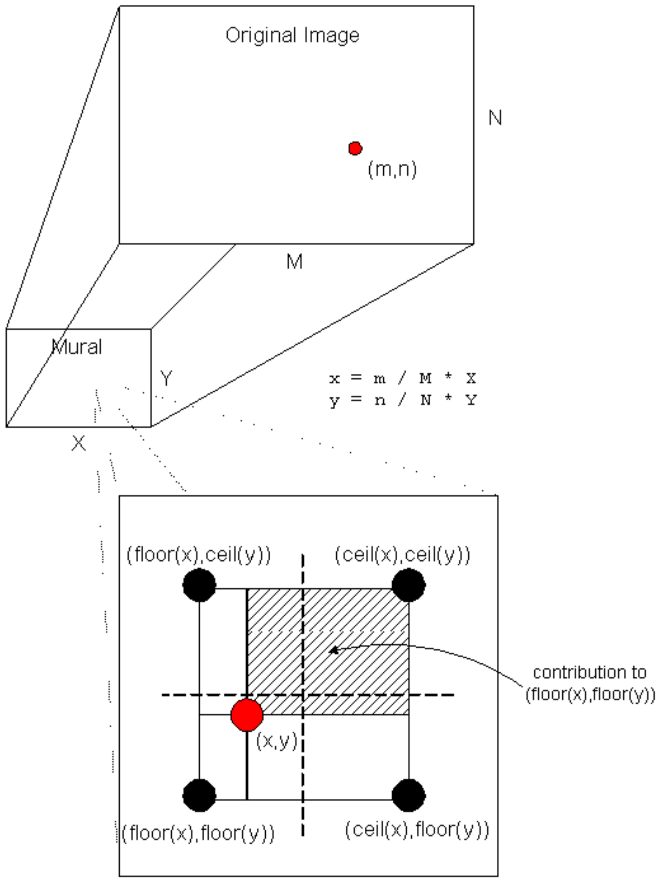


Fig. 11. Information Mural algorithm example, scaling pixel at (m, n) in $M \times N$ original image to (x, y) in $X \times Y$ Mural. Contribution of (x, y) to pixel (floor(x), floor(y)) is shown with diagonal cross-hatch.

the physical location of the area contributions to correctly render subpixel geometries, as in some polygon rendering algorithms [1]. We feel this approach is not required given that Murals are typically of 2D synthetic, abstract images where geometry of overlapping or intersecting polygons is not as important as in rendering 3D scenes.

This algorithm is essentially area sampling using a weighted filter of size $(2M/X - 1) \times (2N/Y - 1)$ to filter the original $M \times N$ binary image into $X \times Y$ pixels. However, when computing intensity of a screen pixel, the total intensity at a given instance of the filter is divided by the maximum intensity over all filter instances instead of the area of the filter (as in traditional area sampling). When this divisor is less than the area of the filter, we are effectively "brightening" a sparse original image. Often the divisor will, in fact, be the area of the filter, when at least one instance of the filter finds all pixels on in the original image.

```
1. for each $x, y$ set $mural\_array[y][x]$ to zero
2. for each pixel $m, n$ in the original representation
   a. compute $x = m / M * X, y = n / N * Y$
   b. compute the area of the quadrants defined by the point $x, y$ and a unit square connecting each of surrounding pixels ($floor(x), floor(y); floor(x), ceil(y); ceil(x), floor(y); ceil(x), ceil(y)$)
   c. add the area of the diagonally opposite quadrant to each $mural\_array$ entry:
        $mural\_array[floor(y)][floor(x)]$
        $mural\_array[ceil(y)][floor(x)]$
        $mural\_array[floor(y)][ceil(x)]$
        $mural\_array[ceil(y)][ceil(x)]$
   d. update $max\_mural\_array\_value$ if one of the four new $mural\_array[][]$ values is a new maximum
3. for each $x, y$ in the $mural\_array$
   a. map the value $mural\_array[y][x] / max\_mural\_array\_value$ to a gray-scale or color intensity varying scale, depending on the type of Mural being created
   b. shade the pixel at $x, y$ of the Mural based on the mapping computed in the previous step
```

4.2 Efficient Algorithm

For improved efficiency, the second algorithm eliminates the weighting filter computations performed in Steps 2b and 2c of the original Mural. This effectively results in a "sharper" image, eliminating the blurring due to overlapping weighting functions. The benefits that weighted area sampling provide for rendering photorealistic images are not as important for Murals of synthetic, abstract images often found in information visualizations.

Another way to look at this version of the algorithm is as a nonoverlapping box filter of size $M/X \times N/Y$ used to filter the original $M \times N$ image. Like in the original algorithm, the total intensity of the source pixels at a given instance of the filter is divided by the maximum intensity over all instances, instead of the area of the filter. This maximum is less than or equal to the area of the filter (when all pixels are on in a given instance of the filter, we are doing unweighted area sampling with a scale factor of unity).

```
1. for each $x, y$ set $mural\_array[y][x]$ to zero
2. for each pixel $m, n$ in the original representation
   a. compute $x = m / M * X, y = n / N * Y$
   b. add 1.0 to $mural\_array[floor(y)][floor(x)]$
   c. update $max\_mural\_array\_value$ if the new $mural\_array[floor(y)][floor(x)]$ is greater than the existing maximum
3. for each $x, y$ in the $mural\_array$
   a. map the value $mural\_array[y][x] / max\_mural\_array\_value$ to a gray-scale or color intensity varying scale, depending on the type of Mural being created
   b. shade the pixel at $x, y$ of the Mural based on the mapping computed in the previous step
```

4.3 Attribute Color Algorithm

We considered two alternative ways that attribute colors could be added to an Information Mural. Before discussing the positives and negatives of each approach, the context of the problem should be mentioned. First, since many pixels in the original image contribute to a single pixel in the Mural and since a screen pixel is by definition a single color, the Mural may not be able to show attribute colors for every piece of data at the same time. What if the Mural compresses 50 pixels from the original image into the same screen pixel, five of which are to be colored blue, 13 red, six yellow, and so on—how should that screen pixel be rendered? It does not make sense to mix RGB values as is done in standard antialiasing, because an observer might not deduce that equal parts of red and green original pixels make a yellow screen pixel. Thus, we chose to color each screen pixel according to the attribute color that occurs most frequently at that pixel in the Mural.

One way to compute this would be to keep track of the intensity for each color separately, requiring a *mural_array* of floats for each different attribute color. Note that just keeping a red, green, and blue array would not work, because colors should not be mixed for the reason mentioned above. Besides the large space requirements, another problem is determining which maximum intensity value should be used to compute the resulting screen pixel density mapping. Without attribute colors, it is obvious: render relative to the maximum over all of the screen pixels. However, with attribute colors, is the reference the maximum density of the resulting color component? Or, is it the maximum density over all possible colors? The third and final option is to treat the intensity at each pixel independently from the attribute colors and, thus, compute the density mapping relative to the maximum of intensity as is done in the previous algorithms.

This leads to the alternative for computing attribute colors that we have chosen to implement. In addition to the array for determining screen pixel density, a list of shorts, one for each possible attribute color, is kept with each *mural_array* entry to record how many pixels from the original image of each attribute color have overlapped each screen pixel. This method was chosen for simplicity, compactness, and efficiency; yet, we sacrifice the ability to correctly perform area sampling—we will end up with an inaccurate reflection of exactly how much of the intensity is due to each color. For example, suppose five antialiased blue original pixels each contribute 0.1 intensity to a screen pixel and one antialiased red pixel contributes 0.8 intensity, the result is a blue screen pixel of 1.3 intensity. This problem only arises in building a Mural using overlapping weighted area sampling because, when nonoverlapping unweighted sampling is done, each point always contributes 1.0 intensity to a single pixel.

Of course, our choice of presenting the pixel color of the maximum contributor has the consequence that the second, third, fourth, and so on, relative contributing colors are not shown, thus "hiding" some of the information in the original image. Interactive Murals can be configured, however, to allow the viewer to query the underlying attribute colors of a section of the Mural and display the results in a separate view. Alternatively, Murals that cyclically render pixels according to all colors contained are possible. For a more general discussion of a variety of these potential interactive brushing techniques, see [25]. This is simply a challenging problem in which the particular task being performed should dictate the approach to be followed.

4.4 Implementation

While the previous subsections on the Information Mural algorithm mentioned many implementation considerations, this subsection will discuss how Information Murals are actually included in visualization applications. In practice, a Mural is *not* constructed directly from an entire original image, but drawn incrementally at a resolution matching that of the source image.

We have implemented an Information Mural as an abstract widget that can be used by an application, just like a scrollbar, drawing area, or other graphical widget. The widget can be used purely for output to display an Information Mural or, more usefully, it can act as a global view for more detailed views by providing a "navigation rectangle" that can be panned and zoomed by the user. The implementations built have been in `C++` on top of `X Windows` and `Motif`, with some also utilizing the `Vz` visualization framework.[1] The `Mural` class we have implemented provides a basic application interface to create, layout, and draw a Mural. Client applications must inherit from the `Mural_Client` class to receive interaction notification messages (method calls) that the application may choose to implement.

When an instance of a `Mural` is created, the application defines the coordinate system in which the Information Mural will be drawn. If the `Mural`'s navigation capabilities are to be used, the initial position and size of the navigation rectangle must also be set. Whenever the `Mural` needs to be redrawn, it calls the application's `MuralRedrawNeededCB()` callback method. The application then calls whichever primitive drawing routines it needs to construct the Mural, such as (`MuralDrawPoint()`, `MuralDrawLine()`, `MuralFrameRectangle()`, etc.). These routines are passed coordinates in the application-defined coordinate system (typically that of the original image of the information space). Additionally, whenever the navigation rectangle is moved or the `Mural` is zoomed by the user, the application's `MuralValueChangedCB()` and `MuralZoomedCB()` are called, respectively, such that the application can update any focus area of the information being displayed.

1. `Vz` is a proprietary cross-platform visualization framework developed by Bell Laboratories, Naperville, Illinois.

In this way, the application draws the Information Mural in its own coordinate space with respect to the information being visualized, and the `Mural` widget handles the rendering of the Mural in whatever space it has on the screen. A user's manipulations of the `Mural` widget are passed back to the application in the application-defined coordinate space as well. Such abstraction makes it easy for an application to use a `Mural` widget to implement a resizable global overview. This feature emphasizes the value of the Mural widget versus an application using rendering hardware, such as the Open GL accumulation buffer [12], to do antialiasing of a scene as it is drawn.

Several other parameters of the `Mural` widget can be changed by the client application. First, the type of color scale is chosen (gray or equalized intensity), as well as the start intensity of the scale, end intensity, and the number of steps in the scale. Based on human abilities to differentiate color levels, we typically use a color scale with 10 steps. Another parameter allows the application to set the mapping from pixel intensity to the level in the color scale. For example, a linear mapping equally distributes the range of computed pixel intensities to the steps in the color scale. A logarithmic mapping allows pixels in the lower range of the computed intensity to be allocated more steps in the color scale.

Section 3 gives many examples of applications using both stand-alone `Mural` widgets and applications that use the `Mural` as a global view through which the user can navigate more detailed views. In the next section, we include a short discussion of the Information Mural technique's limitations.

4.5 Limitations

The Information Mural technique is not without limitations or aspects that may restrict its utility. Limitations of the method itself or problems caused by the content of the original image are mentioned here (see Section 5 for a more detailed discussion of the use of Information Murals in particular visualization tasks). One such aspect, as just described above, is the Mural's use of gray-scale shading or density and, then, the potential addition of color. Density is low in the ordering of elementary graphical perception tasks [7]. Distinguishing fine variations or levels of detail in gray-scale is difficult for people. Shading and density are better suited for illustrating strong patterns or providing a stimulus for an impression of data, tasks for which the Mural is useful.

When color is added, this situation is confounded. Color is better suited for portraying categorical data, rather than continuous values. Using attribute colors to categorize information in a Mural follows this paradigm. The use of color to illustrate numerical ordering and values can be problematic [36]. While this is clearly a problem with the equalized intensity color scale in Murals, we have observed that it is easier to spot low-intensity outliers using color rather than grayscale. Cleveland has suggested color scales that may be more appropriate, such as varying intensity of two hues [8]. Additionally, the work of Bergman et al. has attempted to automate the selection of appropriate colormaps based on date type, data range, spatial frequency, and visualization task [4]. Their recommendations confirm our selection of a gray-scale colormap, but also suggest other viable alternatives.

Furthermore, the context of color, i.e., what other colors adjoin or surround a particular color, strongly influences a person's perception of the color [37]. The use of many different color hues and lightnesses may result in a perceived merging of adjacent colors. Because Murals rely on pixel-level detail, it can be difficult, if not impossible, to notice a single yellow pixel in a sea of gray, for example. For a good summary of the potential dangers of using color in visualizations, see Chapter 11 of [26]. Other related work in the choice of color for data visualizations includes [13].

Another potential liability for the Mural technique concerns the type of data set or image that it is capturing. A scaled down image of a periodic function will eventually compress a single cycle of that function into a single column of screen pixels as the frequency of the function increases (as it does when the duration of the function we are trying to display increases). The result in a typical rendering of this function is simply a band with an amplitude equal to the amplitude of the function. However, the gray-scale Mural gives a bit more information by showing higher intensity at the values that the function takes on more frequently. For example, a Mural of a square wave shows a dark band with bright extremes, and a pulse function will show a dark band with only the lower extreme bright.

If a data set is large enough, say a million values, somewhat random in its distribution, and is mapped to a small window, such as 100×100 pixels, the resulting Mural presents a fuzzy gray cloud. Murals are best for illustrating data sets with noticeable characteristics or patterns.

Finally, to display a Mural or even to display a focus region within a Mural requires examining all the pertinent data points or source image values. This can be computationally slow in browsers requiring selected redisplays. We have included some optimizations and used clever data structures to alleviate this problem somewhat, but manipulating extremely large source data sets still can result in slower displays than desired.

5 Discussion and Evaluation

In Section 3, we presented a number of different applications of the Information Mural technique. All share the notion of visualizing a large data set in a "small" display window. The examples clearly differ in the domain from which the data was generated: software visualization, sun spot activity, census data, text files, and so on. But, more importantly, the examples differ in the fundamental task being conducted and the role or function of the Mural. Three fundamental tasks and accompanying Mural roles are evident in these examples:

- Browsing or navigating through information spaces with the aid of a global overview.
- Discovering attributes of or relationships within multidimensional data from a visualization.
- Studying geographic or spatial data to understand its characteristics.

The first task involves browsing large information spaces. The Information Mural technique itself is not a solution to this problem. Rather, a Mural can be used as the global overview in a larger browser system. For example, Plaisant et al. describe many different styles of information browsers [27]. Some of these browser categories, notably the Single Coordinated Pair, Tiled Multilevel, Free zoom and multiple overlap, and the Bifocal view, utilize global overviews. The authors note

> Dense global views provide experts with direct access to details that would otherwise require several zooming operations (even if these global views appear unreadable to others!).

Information Murals provide a technique for showing overviews that have high fidelity to the way the viewer envisions the data set, thus, they are ideal in this application.

Plaisant et al. further identify five classes of tasks that can be accomplished via browsing: image generation, open-ended exploration, diagnostic, navigation, and monitoring. It is difficult to assess how applicable a Mural itself is to each of these tasks. Rather, a Mural embedded in a particular style of browser could be assessed at the level of assistance provided. The previous section contained a number of examples (software visualization, sun spot, text file) where a Mural was used as an aid to a browsing task.

The second main task for which a Mural can be used is as a data analysis tool to a statistician or anyone seeking to uncover relationships within large data sets. The sun spot example can be thought of as addressing this task. Many other data graphing or visualization techniques do exist, such as averaging, aggregation, box plots, level plots, curve smoothing, and so on. See [8] for an introduction to many of these techniques.

For discovering particular attributes or relationships of data, Information Murals will be inferior to specific instances of these existing techniques. For example, Cleveland describes a plot of time series data of melanoma cases in the state of Connecticut over many years (an Information Mural will be similar to this type of data plot). Viewing this graphic allows one to see the upward trend. But, only by graphing the residuals of a loess trend fit to the data is one able to observe periodic oscillations within the data. A Mural provides no notion of this data attribute.

We feel that a Mural is best used to give a viewer an overall impression and close appreciation for the elements from a large data set. Contrast this with an averaging technique that generates one pixel as the display element for a set of 50 data points. Clearly, an infinite number of sets of 50 different values all could sum to that same average. The Mural technique better illustrates how individual values contribute to an overview visualization.

One other advantage that some of the traditional graphing techniques may hold over the Mural is rendering time. The simpler algorithms for illustrating averages, aggregations, or box plots could display more quickly than a Mural. This difference may be noticeable on repeated change redisplays of extremely large data sets.

The third main task for which an Information Mural is applicable is the presentation of geographic or spatial data. The U.S. Census data example and, to a certain degree, the "words within a text file" example fall within this area. As discussed earlier, a Mural provides a foundation for implementing a type of pixel-oriented chloropleth map of detailed spatial data. Many specialized mapping software packages are available for this task, and the Mural is not a replacement for these, but this example does illustrate the flexibility of the Mural technique for different types of applications. Other potential visualization techniques for illustrating geographic or spatial data include level plots and 3D terrain diagrams.

Overall, we feel that the best application for an Information Mural is the first task above: as a global overview for navigation in a browser. The Mural technique is a relatively straightforward algorithm to implement and it adapts well for different data domains. We have, as mentioned earlier, encapsulated it into a generic user interface widget for a number of different uses.

We feel that Murals can be helpful tools in data visualization and analysis tasks also, but they are clearly not a substitute for existing techniques. Rather, a Mural can be one more asset in the repertoire of analytical tools that scientists employ for examining and understanding large data sets.

Finally, we believe that Information Murals could be used as visualization tools by cartographers for geographic data. As with all illustrations of this type, however, a Mural would only be as good as the data aggregation and classification mappings chosen by the person building the visualization.

6 OTHER RELATED WORK

One area of related work on which the Information Mural is based is the fundamental antialiasing research of computer graphics [1], [10]. The same ideas that help reduce "jaggies" in realistic scenes allow us to combine many data values and display an image that captures data density and distribution. As discussed earlier, we are able to utilize simple antialiasing, area sampling, and filtering techniques and still gain powerful results.

The Information Mural presented here can be thought of in general terms as one of a set of potential methods for visualizing large data sets. The Mural is well suited for conveying an overview, global view, or the *context* aspect in the *focus + context* visualization paradigm common in the research community currently.

One of the best-known focus + context techniques is the fisheye lens [11], [31]. In a fisheye view, one area serves to enclose both the focus and context components. One or more regions are "expanded" to show more detail, while surrounding regions are "de-emphasized" or made smaller. Fisheye views typically involve some distortion, however. The Mural is better suited for separate focus and context regions, similar to that done in [2]. One could envision placing a focus filter or lens on top of a Mural, much as done in the Movable Filter or Magic Lens [35] techniques, as a way of providing an integrated focus + context view.

An information visualization that could take advantage of the Information Mural technique is the Table Lens [29], a visualization technique for illustrating tabular data. It can present relatively large tables using a fish-eye technique: Some rows or columns can be expanded (in focus), while others are collapsed to their minimum size, a single row or column of pixels. The Table Lens is restricted to illustrating a table with a number of rows or columns less than or equal to the number of pixels available. The Information Mural technique would allow the Table Lens to compress the representation beyond this limit, giving the Table Lens more room to display the table entries that are in focus.

Other information visualization techniques exist for presenting large data sets. The Cone Trees and Perspective Wall of the Information Visualization system [30] use natural 3D perspective to generate a form of focus + context view. Treemaps [21] use a "slice-and-dice" rectangular region technique for visualizing information. The SeeSoft [9] system mentioned earlier in this article represents lines in a text file by lines of pixels. Fundamentally, however, all these information visualization techniques are best suited for structured, hierarchical data, such as files in a directory structure or text items in a large database. The Mural technique, although related in the goal of presenting large information spaces, is better suited for raw data visualizations of low dimensionality.

A number of alternative data visualization techniques exist for illustrating the kind of data shown here with the Information Mural. Mentioned earlier were traditional averaging, aggregation, and box plot techniques.

Keim's "recursive structure" technique [22] uses similar screen real estate to portray data sets of similar magnitude to those with the Mural. Keim's technique, however, does not preserve the gestalt natural form of the plotted data. It is a visualization technique that must be learned to be understood. Its advantage over the Mural is that it can scale up to data sets of higher dimensionality and is, in a sense, more "information dense."

Stasko and Muthukumarasamy introduce a visualization technique for illustrating extremely large bivariate data sets [34]. Their technique is similar to box plots in aggregating consecutive regions of data values into a rectangle. The horizontal position of the rectangle defines its data set position, and the top and bottom height of the rectangle denote the relative maximum and minimum values within the region. The height of a bright horizontal line within the rectangle denotes the mean value in the region, and the shading of the region indicates the "sortedness" of its data values.

7 Conclusion

An Information Mural is a 2D, graphical representation of a large information space that fits entirely within a display window or screen. The miniature representation is drawn using antialiasing compression techniques and intensity shading, and is useful for visualizing trends and patterns in the overall distribution of information. By adding panning and zooming capabilities to Information Murals, they can be used as stand-alone visualizations or as global views along with more detailed informational displays.

The Information Mural technique can be integrated into various information visualization applications to help display large information spaces. In browsing information or examining a large data set, it is often useful to start with a global overview of the information. Information Murals can convey more information about large data spaces than traditional techniques, and allow overviews of certain types of information spaces to be created when before they could not. Another advantage of the Information Mural technique is that the application need not concern itself with how much space is available to render the information—the density and attribute mappings are computed automatically based on the available screen space for the view.

Acknowledgments

This research was supported in part by the U.S. National Science Foundation through grant CCR-9628149.

References

[1] J. Barros and H. Fuchs, “Generating Smooth 2-D Monocolor Line Drawings on Video Displays,” *Proc. 1979 SIGGRAPH Conf.*, vol. 13, pp. 260-269, Aug. 1979.

[2] D.V. Beard and J.Q. Walker, “Navigational Techniques to Improve the Display of Large Two-Dimensional Spaces,” *Behaviour and Information Technology*, vol. 9, no. 6, pp. 451-466, 1990.

[3] R.A. Becker and W.S. Cleveland, “Brushing Scatterplots,” *Technometrics*, vol. 29, no. 2, pp. 127-142, 1987.

[4] L.D. Bergman, B.E. Rogowitz, and L.A. Treinish, “A Rule-Based Tool for Assisting Colormap Selection,” *Proc. IEEE Visualization ‘95 Conf.*, pp. 118-125, 1995.

[5] R. Chimera, “Value Bars: An Information Visualization and Navigation Tool for Multiattribute Listings (Demo Summary),” *Proc. ACM SIGCHI ‘92 Conf. Human Factors in Computing Systems*, pp. 293-294, 1992.

[6] W. Citrin, A. Cockburn, J. von Kanel, and R. Hauser, “Using Formalized Temporal Message-Flow Diagrams,” *Software Practice and Experience*, vol. 25, pp. 1,367-1,401, 1995.

[7] W.S. Cleveland, *The Elements of Graphing Data*. Pacific Grove, Calif.: Wadsworth and Brooks / Cole, 1985.

[8] W.S. Cleveland, *Visualizing Data*. Summit, N.J.: Hobart Press, 1993.

[9] S.G. Eick, J.L. Steffen, and E.E. Sumner, Jr., “SeeSoft—A Tool for Visualizing Line Oriented Software Statistics,” *IEEE Trans. Software Eng.*, vol. 18, no. 11, pp. 957-968, Nov. 1992.

[10] J.D. Foley, A. Van Dam, S.K. Feiner, and J.F. Hughes, *Computer Graphics Principles and Practice*. Reading, Mass.: Addison-Wesley, 1990.

[11] G.W. Furnas, “Generalized Fisheye Views,” *Proc. ACM SIGCHI ‘86 Conf. Human Factors in Computing Systems*, pp. 16-23, Apr. 1986.

[12] P. Haeberli and K. Akeley, “The Accumulation Buffer: Hardware Support for High-Quality Rendering,” *Proc. 1990 SIGGRAPH Conf.*, pp. 309-318, Aug. 1990.

[13] C.G. Healey, “Choosing Effective Colours for Data Visualization,” *Proc. IEEE Visualization ‘96 Conf.*, pp. 263-270, San Francisco, Oct. 1996.

[14] M.A. Hearst, “TileBars: Visualization of Term Distribution in Full Text Information Access,” *Proc. ACM SIGCHI ‘95 Conf. Human Factors in Computing Systems*, pp. 59-66, Denver, Colo., 1995.

[15] M.T. Heath and J.A. Etheridge, “Visualizing the Performance of Parallel Programs,” *IEEE Software*, vol. 8, no. 5, pp. 29-39, Sept. 1991.

[16] W.C. Hill, J.D. Hollan, D. Wroblewski, and T. McCandless, “Edit Wear and Read Wear.,” *Proc. ACM SIGCHI ‘92 Conf. Human Factors in Computing Systems*, pp. 3-9, May 1992.

[17] D.F. Jerding and S. Rugaber, “Using Visualization for Architectural Localization and Extraction,” *Proc. Fourth Working Conf. Reverse Eng.*, pp. 56-65, Oct. 1997.

[18] D.F. Jerding and J.T. Stasko, “The Information Mural: A Technique for Displaying and Navigating Large Information Spaces,” *Proc. IEEE Visualization ‘95 Symp. Information Visualization*, pp. 43-50, Oct. 1995.

[19] D.F. Jerding and J.T. Stasko, “Using Information Murals in Visualization Applications,” *Proc. 1995 Symp. User Interface Software and Technology (Demonstration)*, pp. 73-74, Nov. 1995.

[20] D.F. Jerding, J.T. Stasko, and T. Ball, “Visualizing Interaction Patterns in Program Exections,” *Proc. 1997 Int’l Conf. Software Eng.*, pp. 360-370, May 1997.

[21] B. Johnson and B. Shneiderman, “Tree-Maps: A Space Filling Approach to the Visualization of Hierarchical Information Structures,” *Proc. IEEE Visualization ‘91 Conf.*, pp. 284-291, Oct. 1991.

[22] D. A. Keim, H.-P. Kriegel, and M. Ankerst, “Recursive Pattern: A Technique for Visualizing Very Large Amounts of Data,” *Proc. IEEE Visualization ‘95 Conf.*, pp. 279-286, Oct. 1995.

[23] D. Kimelman and B. Rosenburg, “Strata-Various: Multi-Layer Visualization of Dynamics in Software System Behavior,” *Proc. IEEE Visualization ‘94 Conf.*, pp. 172-178, Oct. 1994.

[24] C. Laffra and A. Malhotra, “Hotwired—A Visual Debugger for C++,” *Proc. USENIX Sixth C++ Technical Conf.*, Apr. 1994.

[25] A.R. Martin and M.O. Ward, “High Dimensional Brushing for Interactive Exploration of Multivariate Data,” *Proc. 1995 IEEE Visualization Conf.*, pp. 271-278, Oct. 1995.

[26] M. Monmonier, *How to Lie With Maps*, second ed. Chicago, Ill.: Univ. of Chicago Press, 1996.

[27] C. Plaisant, D. Carr, and B. Shneiderman, “Image-Browser Taxonomy and Guidelines for Designers,” *IEEE Software*, vol. 12, no. 2, pp. 21-32, Mar. 1995.

[28] B.A. Price, R.M. Baecker, and I.S. Small, “A Principled Taxonomy of Software Visualization,” *J. Visual Languages and Computing*, vol. 4, no. 3, pp. 211-266, Sept. 1993.

[29] R. Rao and S. K. Card, “The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus + Context Visualization for Tabluar Information,” *Proc. ACM SIGCHI ‘94 Conf. Human Factors in Computing Systems*, pp. 318-322, Boston, Apr. 1992.

[30] G.G. Robertson, S.K. Card, and J.D. Mackinlay, “Information Visualization Using 3D Interactive Animation,” *Comm. ACM*, vol. 36, no. 4, pp. 57-71, Apr. 1993.

[31] M. Sarkar and M.H. Brown, “Graphical Fisheye Views of Graphs,” *Proc. ACM SIGCHI ‘92 Conf. Human Factors in Computing Systems*, pp. 83-91, May 1992.

[32] *Software Visualization: Programming as a Multimedia Experience*, J. Stasko, J. Domingue, M.H. Brown, and B.A. Price, eds. Cambridge, Mass.: MIT Press, 1998.

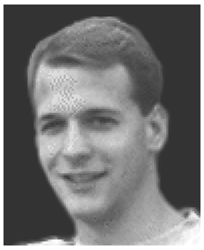
[33] J.T. Stasko and E. Kraemer, “A Methodology for Building Application-Specific Visualizations of Parallel Programs,” *J. Parallel and Distributed Computing*, vol. 18, no. 2, pp. 258-264, June 1993.

[34] J.T. Stasko and J. Muthukumarasamy, “Visualizing Program Executions on Large Data Sets,” *Proc. 1996 IEEE Symp. Visual Languages*, pp. 166-173, Sept. 1996.

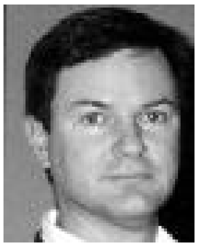
[35] M.C. Stone, K. Fishkin, and E.A. Bier, “The Movable Filter as a User Interface Tool,” *Proc. ACM SIGCHI ‘94 Conf. Human Factors in Computing Systems*, pp. 306-312, Apr. 1992.

[36] E.R. Tufte, *The Visual Display of Quantitative Information*. Cheshire, Conn.: Graphics Press, 1983.

[37] E.R. Tufte, *Envisioning Information*. Cheshire, Conn.: Graphics Press, 1990.



Dean Jerding received his PhD in computer science from the Georgia Institute of Technology in 1997, as a student in the Graphics, Visualization, and Usability Center. He earned his MS in computer science from Georgia Tech in 1996 and a BS in electrical engineering from the University of Virginia in 1988. Dean's interests include software architecture and object-oriented design, visualization, and user-interface design. He is currently a staff engineer at Scientific-Atlanta, Inc. working on software architecture and user-interfaces for the next generation digital multimedia home communications terminal.



John Stasko received his BS degree in mathematics from Bucknell University in 1983 and his ScM and PhD degrees in computer science from Brown University in 1985 and 1989, respectively. He is an associate professor in the Graphics, Visualization, and Usability Center and the College of Computing at the Georgia Institute of Technology. His current research interests include software and information visualization, human-computer interaction, programming environments, and software agents.