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

Dean F. Jerding and John T. Stasko

Graphics, Visualization, and Usability Center
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280

{dfj,stasko}@cc.gatech.edu

Technical Report GIT-GVU-97-24
(Replaces Technical Report GIT-GVU-96-25)

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 grayscale shading, intensity, color, and pixel size, along with anti-aliased 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.

Keywords: 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, Carr, and Shneiderman have categorized different types of tasks, including image generation, open-ended exploration, diagnostic, navigation, and monitoring[30]. 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 article, typically utilize abstraction, aggregation, over-plotting, 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 anti-aliasing 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, *ie.*, 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:

- Time-oriented visualizations often span many computer screens if laid out completely. These types of views are particularly prevalent in software visualization[31] and monitoring applications.

- Visualizations that contain miniature representations of information are forced to make tradeoffs 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 x 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 `I x J` 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*[9].

The idea of the Information Mural technique is to make the density of overlap visually apparent. It draws on strategies for anti-aliasing in computer graphics by varying intensity of the screen pixels to convey the underlying density of pixels from the original sample. To construct an `I x J` Information Mural of the original image, the position of each pixel in the `M x 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 *grayscale* 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 grayscale 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 differnt 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.

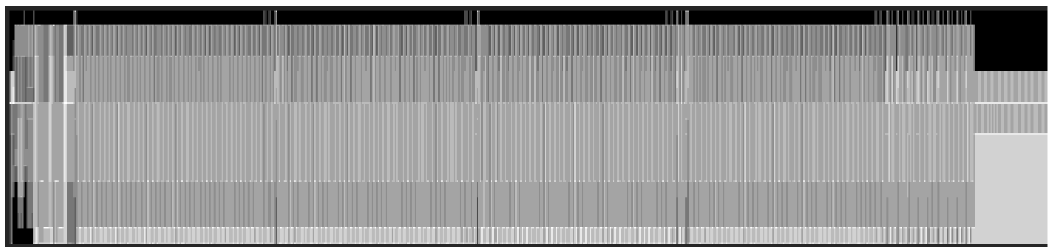
Below are some snapshots from visualization applications we have built using Information Murals. These applications contain many different forms of information, from software to data to text documents, some of which were mentioned in [21]. The examples are broken down here by data domain. Section 4 that follows then characterizes the examples according to the fundamental *task* they are facilitating.

3.1 Software Visualization

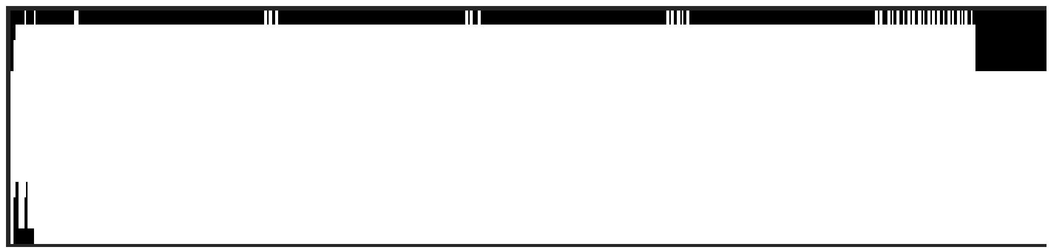
The Information Mural technique originated in our software visualization research into visualization of object-oriented (OO) program executions[19, 20]. Murals are currently being used in a suite of views to support program understanding during design recovery, validation, and reengineering tasks[22].

3.1.1 Object-Oriented Message Traces

Imagine an event trace diagram for object-oriented message sequences turned on its side, 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. Now imagine that you can



(a)



(b)

Figure 1: (a) Mural of object-oriented message trace of over 50,000 messages, drawn in an area 500 pixels wide. (b) Same diagram drawn by just over-plotting (without the mural technique).

see an event trace diagram of an entire program execution, which might contain hundreds of thousands of messages. Fig. 1a shows a grayscale, aliased Information Mural of a message trace from a bubble sort algorithm animation built using the Polka toolkit [35], 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 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. 1b.

Illustrating a program execution via the mural allows the viewer to perceive phases in the execution as well as the classes participating in each phase, which can be an important factor in software analysis[25]. Essentially, it provides an overview of the entire run that serves as context for various program understanding tasks. As will be mentioned in the following example, this type of mural of a program execution is valuable for navigation and focus operations.

One of the views from our prototype OO program visualization suite is called the *Execution Mural* (Fig. 2). This view is used to examine message traces from object-oriented programs[20]. Again, time runs from the left to right in the mural, and classes are laid out along the y-axis. Classes can be listed vertically according to their alphabetical order, by their appearance order in source files, or by viewer specification.

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 navigational area that includes

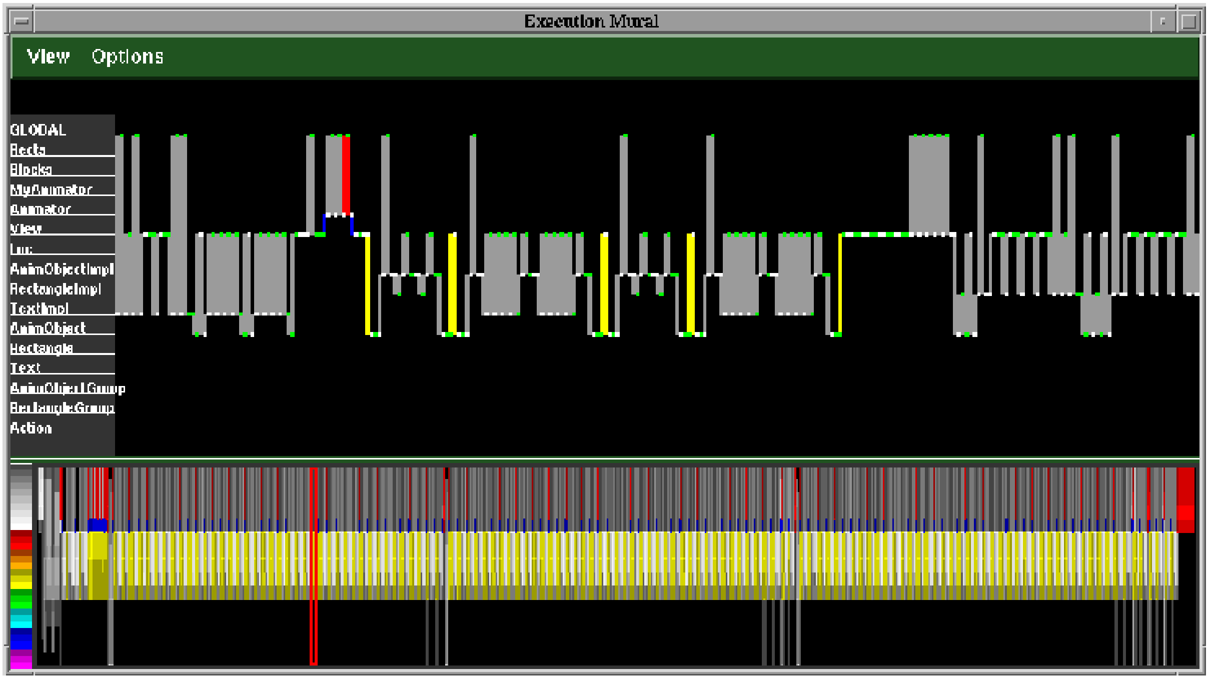


Figure 2: Execution Mural view of bubble sort algorithm animation built using the Polka animation toolkit.

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 moved around to change the focus display above. Also, the viewer can utilize simple brushing techniques[3] through accompanying user interface actions: the user can select a class or classes, assign them a particular color, and then view the resulting display. This allows the user to view class involvement at different stages of the program execution.

Notice that the color of several different messages has been set in the focus area. The Information Mural technique allows the coloring of information attributes using varied color intensity that still reflects the underlying information density, as is evident by the colored areas in the mural.

The mural gives a quick insight into various phases in the execution, including very repetitive patterns. In fact, being able to construct and observe global views of various message traces gave us insight into the existence of message patterns and sub-patterns in object-oriented programs. It was this observation that motivated the work described in [22] where we treat repeated sequences of messages as higher-level abstractions that correspond to design-level scenarios or language-level idioms. The message coloring in the Execution Mural view also allows the location of particular messages in the execution to be identified; without a global view that can actually “show” every message, it would be difficult to find obscure messages in a lengthy message trace.

Other software visualization tools utilize miniature time-line views to portray execution information. Typically a scrolling view is used that shows a subset of the execution that can fit in the available pixels, or over-plotting occurs as the execution time grows larger. For example, the HotWired visual debugger for `C++` and `Smalltalk` provides both object views

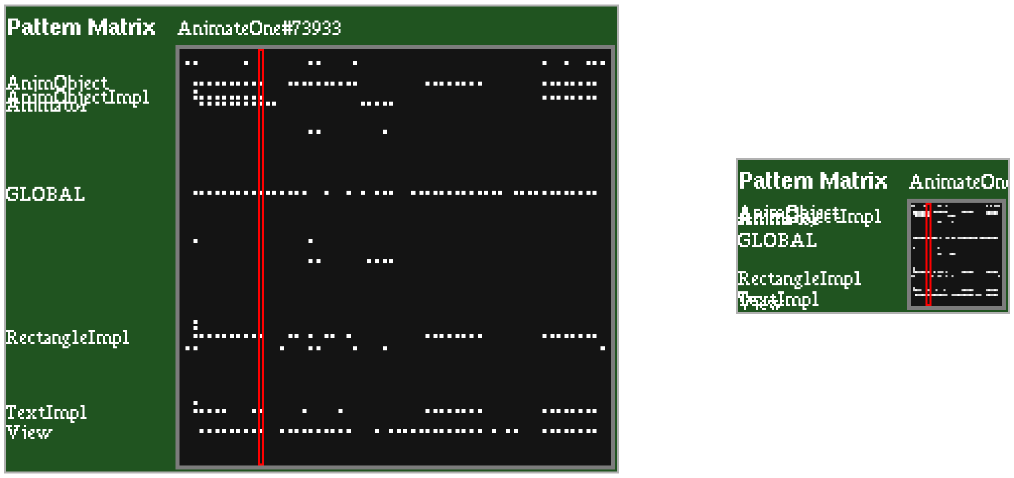


Figure 3: Pattern Matrix view of message patterns identified in an execution of the bubble sort algorithm animation, shown at two different sizes. Rows in the matrix are the classes in the program and columns are marked to indicate class membership in identified patterns. In the larger version, each matrix entry is allocated 4 pixels, while in the smaller version, more than one entry occupies a single pixel.

and a scripting language to create simple program visualizations[27]. A recording strip view is used to portray instance activation over time. The Information Mural technique could be utilized to increase the amount of historical information that can be displayed. As another example, the PV program visualization system provides concurrent, coordinated, and multi-layered views of program behavior[25]. The time-oriented system and process state information views use pixel-level color strips that extend over time to present state history. These views scroll to the right as the program executes, and can be zoomed in to decrease the scale of the strips. Other memory views use colored pixel bands to represent the contents of different memory locations. The Information Mural technique could be used in these views to help alleviate over-plotting problems and allow the views to depict occluded information density when fully zoomed out.

Another view in our OO visualization prototype shows how the Information Mural technique can be used to create scalable matrix views. Our visualization tools identify repeated message sequences in OO program executions. Information about these message patterns is displayed in the *Pattern Matrix* view. The matrix shows patterns as columns and indicates classes and messages that are "members" of observed patterns as entries in the rows. Because there may be several hundred classes and thousands of messages, as well as hundreds of message patterns, there could be more rows or columns than there are pixels in the view.

Entries in the matrix automatically take up available space as the view is resized. So, if there are 50 classes and 500 pixels available in the vertical dimension, each row can take up 10 pixels. However, entry size takes into consideration the scale along both axes, so if there are a large number of message patterns requiring a very small horizontal resolution the vertical resolution will be reduced so as not to render an entry as a vertical line. Fig. 3 shows the same Pattern Matrix view at two different sizes.

An information visualization that could take advantage of Information Murals in a similar way is the Table Lens[32]. The Table Lens is 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 (focus), while others are collapsed to their minimum size, a single row or column of pixels. Fundamentally, however, 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 so that multiple rows or columns could be compressed into the same row or column of pixels. This would give the Table Lens more room to display entries in focus, especially for very large spreadsheets.

3.1.2 Parallel Processor Message Passing

Understanding the execution of parallel and distributed programs is still a particularly challenging problem. Both debugging and performance optimization can benefit from helpful visualizations of a program's execution[26, 15, 16]. Unfortunately, visualizations of the message passing during executions of programs on parallel architectures become very unclear when long durations of time are shown. Typically, scrolling displays are used to capture an entire program execution. An Information Mural provides a way of illustrating more data without scrolling.

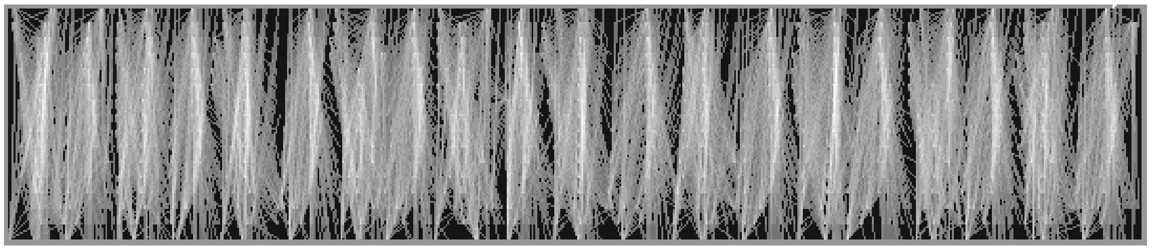
The aliased mural of Fig. 4a shows the kernel integer sort benchmark executing on 16 processors using the PVM distributed system, generated using the PVaniM system built at Georgia Tech[38]. Each processor is assigned a row on the vertical axis, and a message is drawn as a line from one processor to another at the appropriate time coordinates. This particular view uses wall clock timestamps. As is evident from the traditional over-plotted representation shown if Fig. 4b, the mural gives a much better resolution to the image.

As was done in the Execution Mural view, a mural can be used in the background of a global overview to allow more detailed examination of the message passing. Fig. 5 shows the same message trace, this time with messages colored according to message type. The global overview provided by the mural gives an immediate indication of the phases and sub-phases of the algorithm, as well as showing anomalies such as network blockage or processors waiting for others to complete.

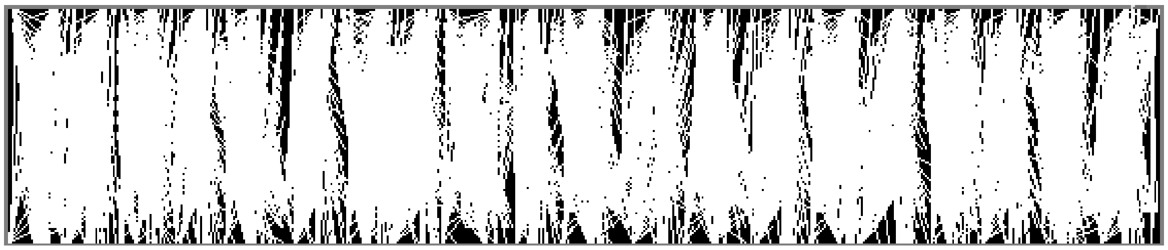
The space-time view of the `PVM` kernel integer sort benchmark shown using ParaGraph[14], a parallel program visualization system, is included as Fig. 6. When the entire run is compressed into the view, messages blur together and make overall patterns less clear. Additionally, if message attribute colors are overlaid in this view, those messages that are drawn "on top" occlude the attributes of those "below". The Information Mural technique would help minimize these effects by automatically computing the correct attribute color and intensity for each pixel after all messages have been drawn.

3.2 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.



(a)



(b)

Figure 4: (a) Mural of parallel program message trace of the kernel integer sort benchmark. (b) Same diagram drawn by just over-plotting (without the mural technique).

Much of this data was obtained from the StatLib server at Carnegie Mellon University.

3.2.1 Sun Spots

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. 7 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. 8 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 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, as opposed to averaging techniques, that 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. 8).

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. 9 shows the sun spot mural zoomed in on a small area. Fig. 10 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

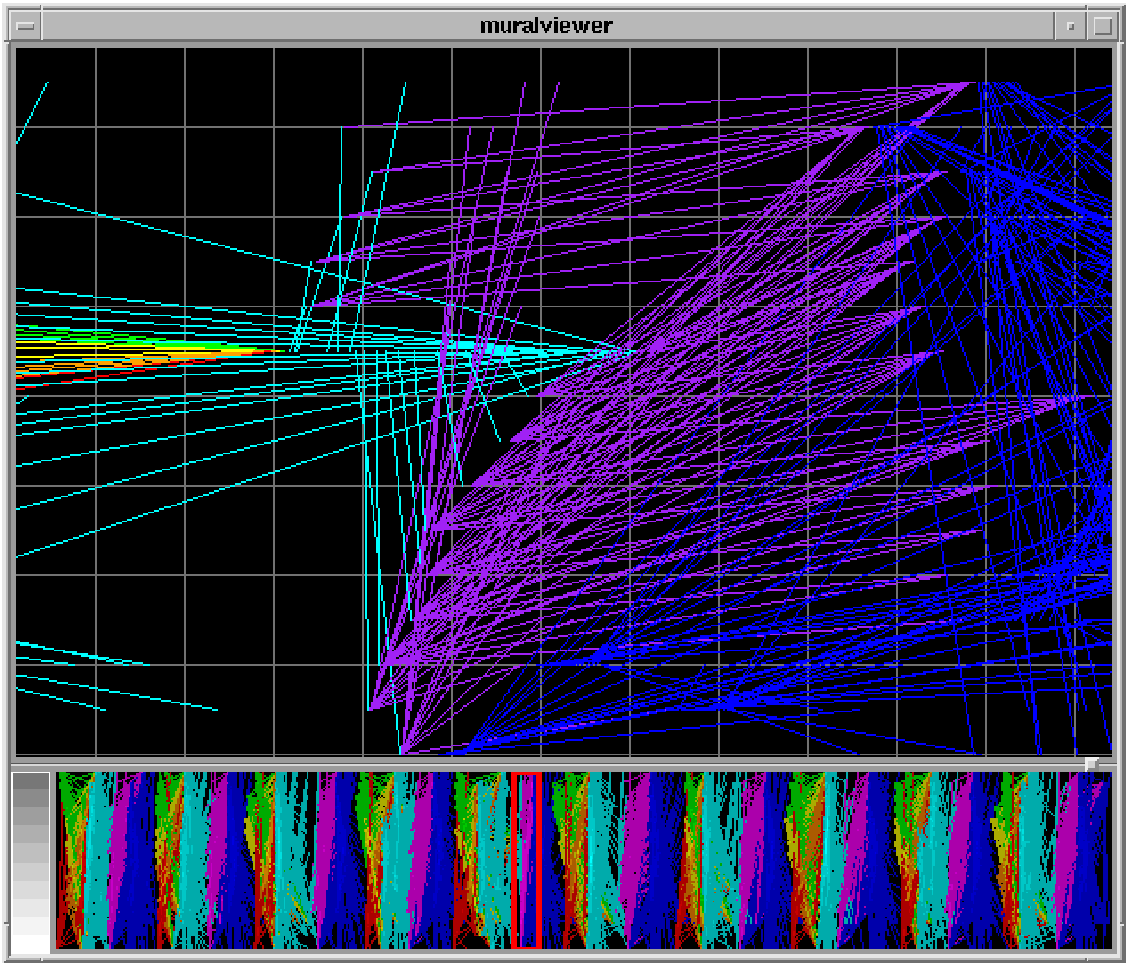


Figure 5: View of message passing in kernel integer sort parallel processor benchmark, with focus area and global overview created using the Information Mural technique.

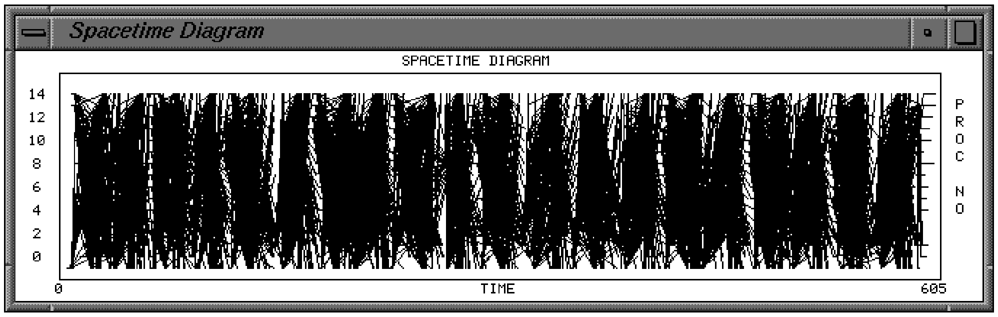


Figure 6: ParaGraph space-time view of message passing in kernel integer sort parallel processor benchmark.

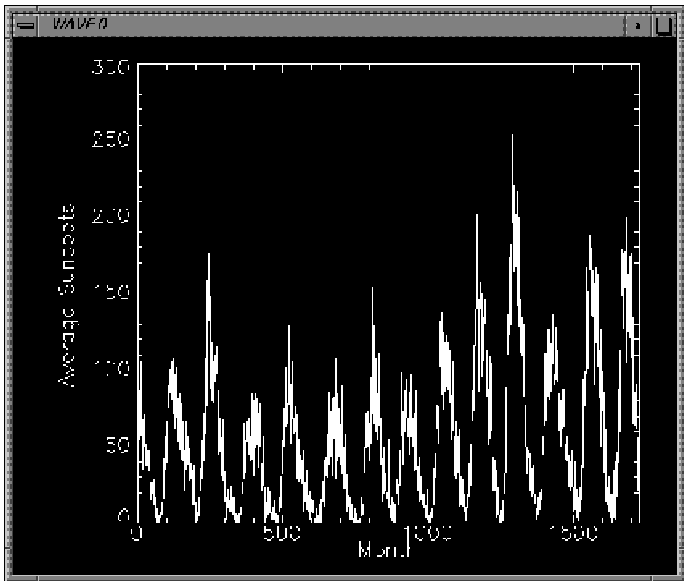


Figure 7: Plot of average number of sun spots recorded per month, 1850-1993.

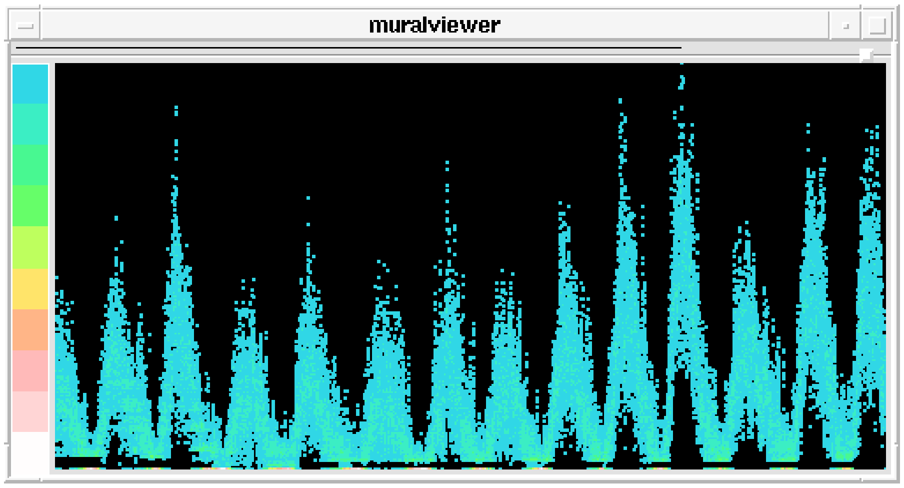


Figure 8: Mural of the number of sun spots recorded daily, 1850-1993.

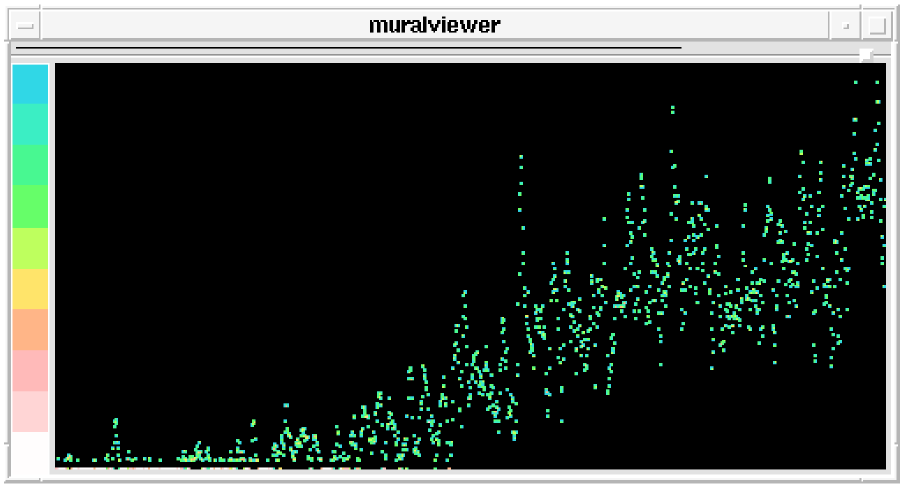


Figure 9: Mural of the number of sun spots recorded daily, 1850-1993, zoomed in on a small area.

curve fitting[7]. 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 [7].

3.2.2 River Flow Data

Another interesting large data set is the mean daily Saugeen river flows, from Jan 1, 1915 to Dec 31, 1979. The anti-aliased mural of this data shows a periodic pattern, with concentrations at the lower values. At first glance, some bright spots occur seemingly randomly above the lower portion of the mural shown in Fig. 11a. Zooming in on a small area at the bottom, we find that the bright spots in the mural are due to single values that occur repetitively (Fig. 11b). We hypothesize that these might be weeks or months in the data where a single value was extrapolated across the entire period to create the daily values. Here the mural technique gives us some quick insight into the structure of the data.

3.2.3 Automobile Data

Another data visualization technique valuable for presenting multivariate data sets is parallel coordinates[18]. The Information Mural technique can be used to assist parallel coordinate data displays. A data set from the Committee on Statistical Graphics of the American Statistical Association (ASA) Second (1983) Exposition of Statistical Graphics Technology contains 406 observations on the following 8 variables: MPG (miles per gallon), number of cylinders, engine displacement (cu. inches), horsepower, vehicle weight (lbs.), time to accelerate from O to 60 mph (sec.), model year (modulo 100), and origin of car (1. American, 2. European, 3. Japanese). Fig. 12a shows a parallel coordinate mural of a subset of the data, including MPG, displacement, horsepower, weight, acceleration, and model year. Part (b) of Fig. 12 shows the standard parallel coordinate view without the mural. In Fig. 12c, color

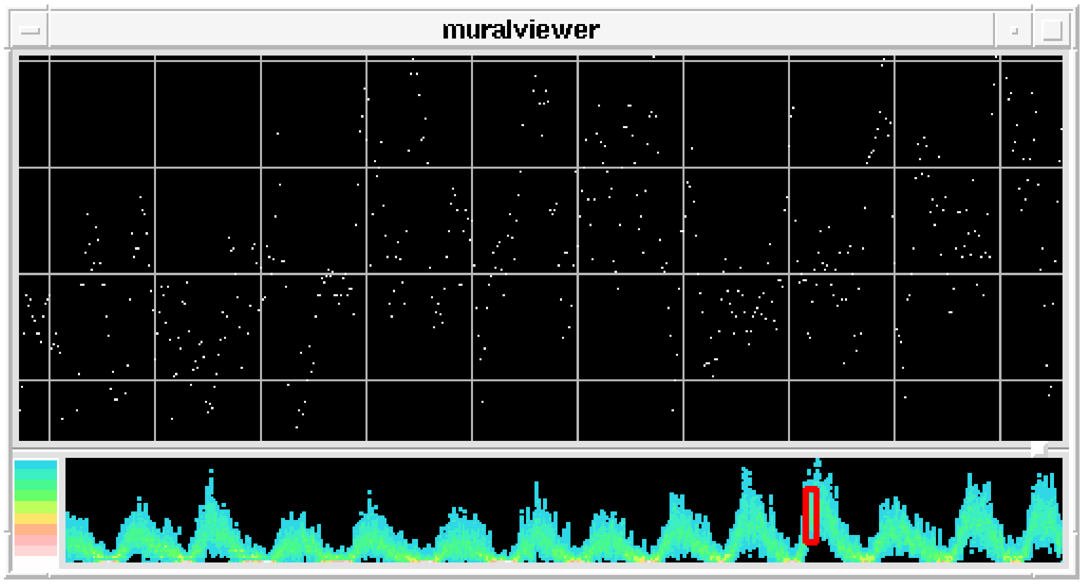


Figure 10: View of sun spots showing focus area and mural of entire data set at the bottom.

has been overlaid on the mural according to the number of cylinders attribute. Notice how the data tuples with fewer cylinders tend to have higher MPG, smaller displacement, less horsepower, and longer acceleration times.

The value of the mural technique in this example is probably not worth the overhead of including it in a parallel coordinate display. The technique does, however, eliminate the "last one drawn appears on top" ordering effect that occurs when drawing colored lines (similar to the problem with the ParaGraph view described above). In a typical parallel coordinate-based system, interactive controls allow the viewer to vary the focus attribute(s).

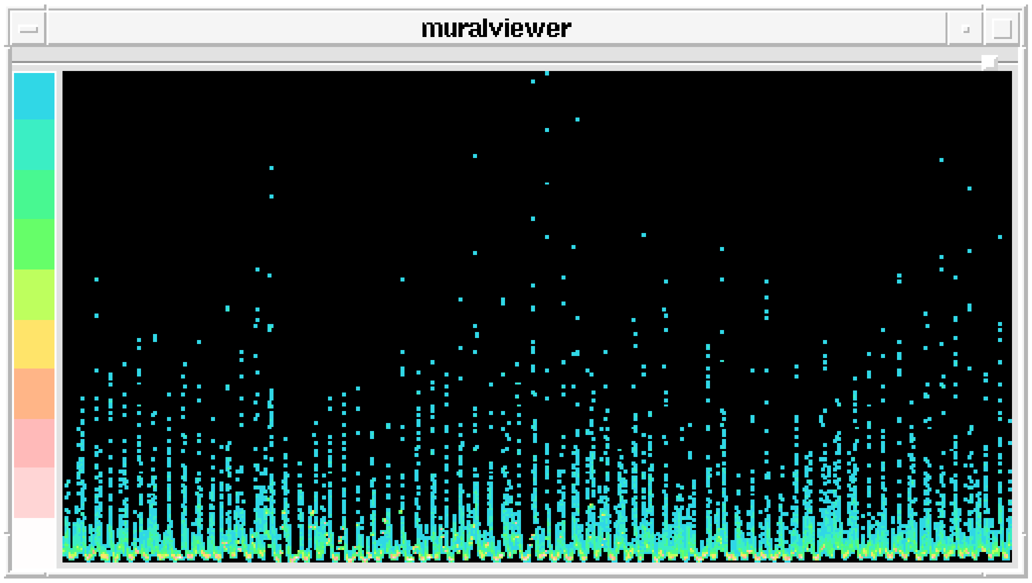
3.3 Information Visualization

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

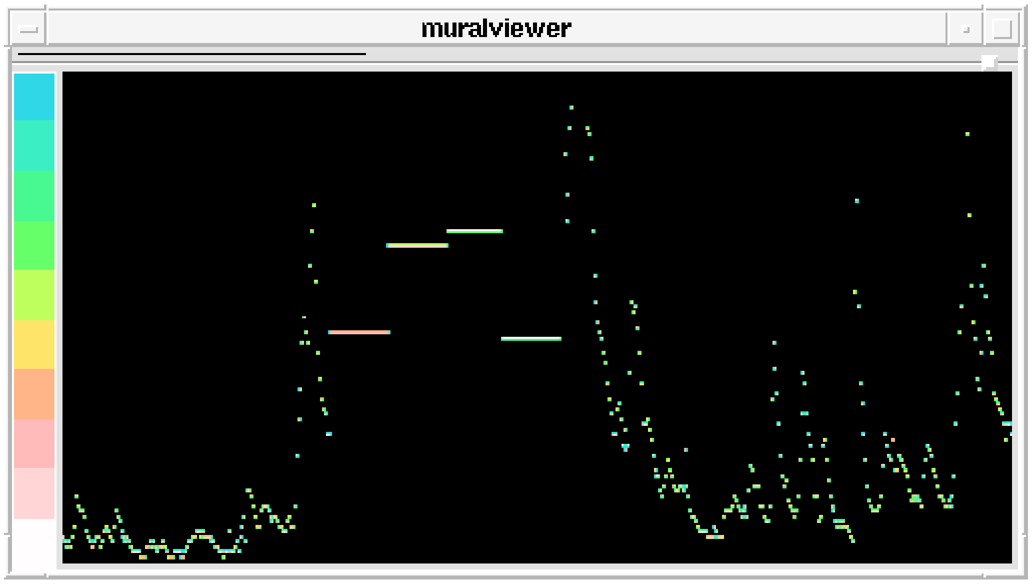
3.3.1 Geographic Information

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*[29]. A chloropleth map breaks down a geographic area into smaller regions that are then given a grayscale shade or color to indicate some value over that region. How to do this areal breakdown is one of the challenges of map-making. 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 USA population figures, and the display region is relatively small, creating a meaningful visualization that adequately conveys population densities can be challenging. The Information Mural technique com-

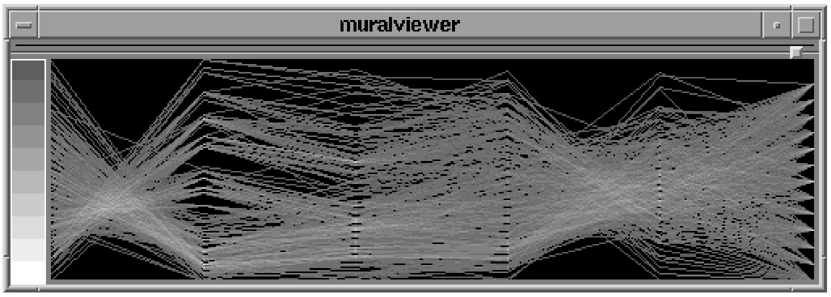


(a)

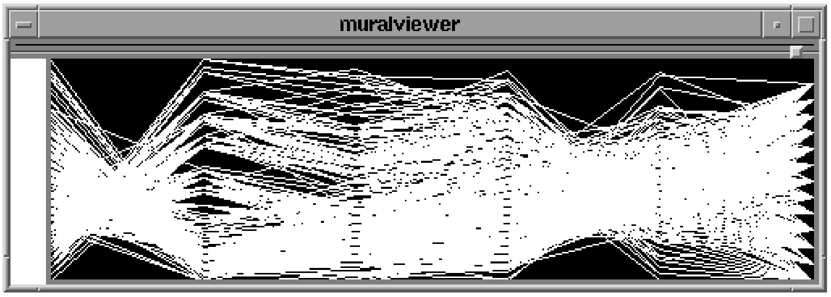


(b)

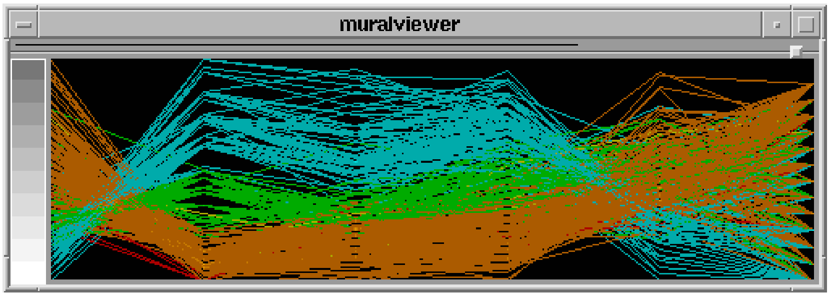
Figure 11: (a) Mural of the mean daily river flow rates of the Saugeen river, 1915-1979. (b) Part (a) zoomed on small area at the bottom of the mural.



(a)



(b)



(c)

Figure 12: (a) Mural of a parallel coordinate view of automobile data showing MPG, engine displacement, horsepower, weight, acceleration, and model year (1970-1982). (b) Standard parallel coordinate view of the data. (c) Color overlaid for number of cylinders (3 = red, 4 = orange, 5 = yellow, 6 = green, 8 = cyan).

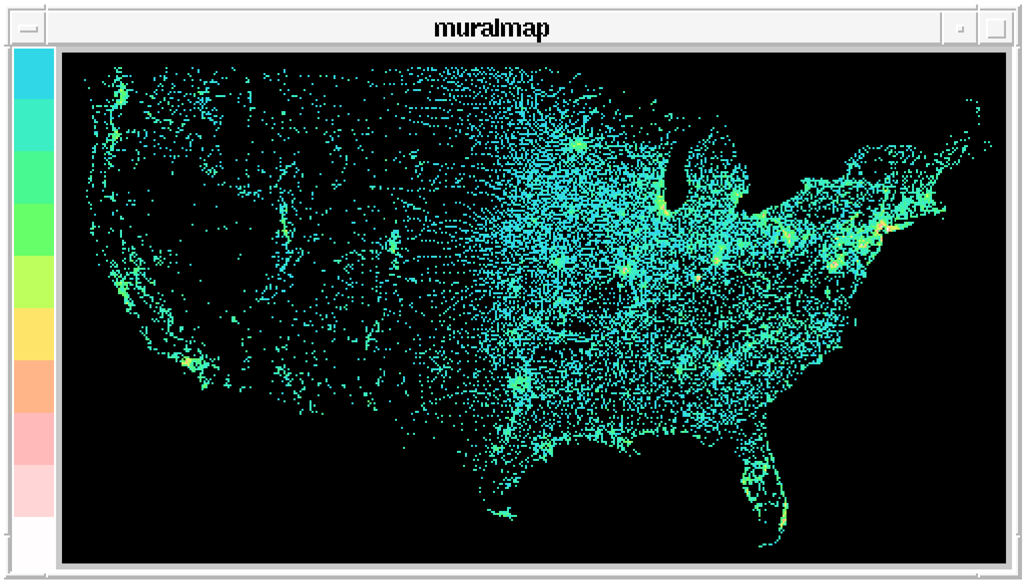


Figure 13: Mural of population density distribution, using data from the 1990 census.

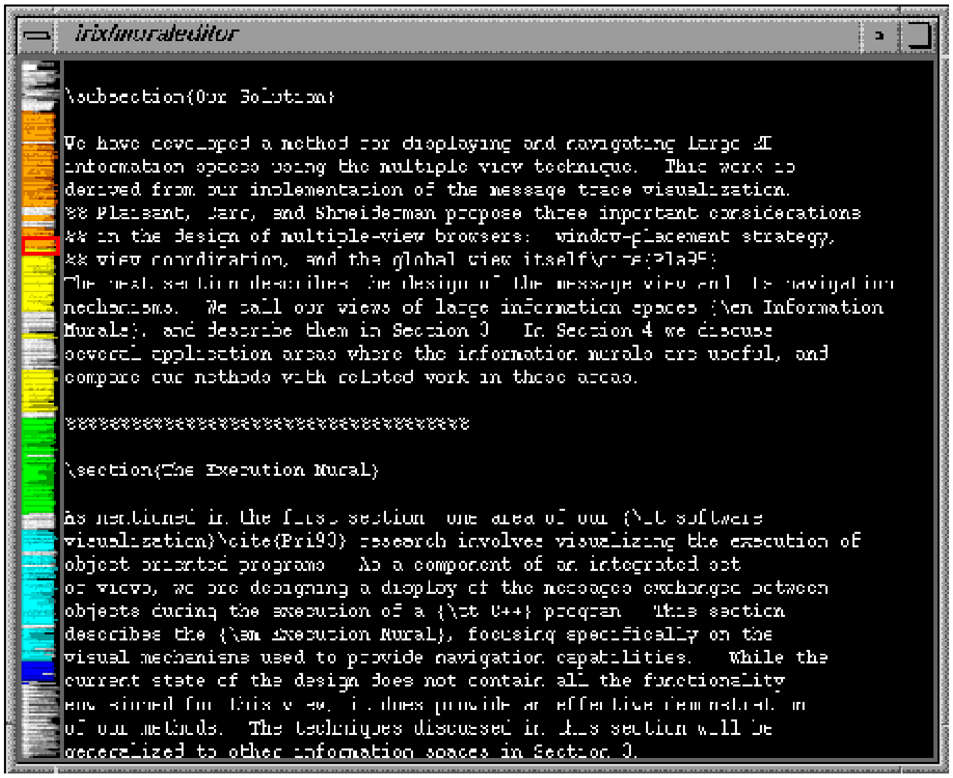
putes information density automatically, making the display of a population density map on a computer screen quite easy. Fig. 13 illustrates an Information Mural of US population data taken from the Tiger Mapping Service U.S. Places File, created from the Census file STF-1A. Each data point provides 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 2.2). 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 chloropleth map with an individual pixel as the area sub-unit. 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 mapmakers. How population values are then classified or mapped to the resulting images shades or colors is still the critical issue, however. [29] describes how varying that mapping can result in markedly different presentations. Figure 13 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.

3.3.2 Text Documents

While SeeSoft[8] 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 grayscale mural representation of a document, color



```
\subsection{Our Solution}

We have developed a method for displaying and navigating large AT
information spaces using the multiple view technique.  This work is
derived from our implementation of the message trace visualization.
%% Plaisant, Carr, and Shneiderman propose three important considerations
%% in the design of multiple-view browsers:  window-placement strategy,
%% view coordination, and the global view itself\cite{Pla95}
The next section describes the design of the message view and its navigation
mechanisms.  We call our views of large information spaces {\em Information
Murals}, and describe them in Section 3.  In Section 4 we discuss
several application areas where the information murals are useful, and
compare our methods with related work in these areas.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

\section{The Execution Mural}

As mentioned in the first section, one area of our {\it software
visualization}\cite{Pri93} research involves visualizing the execution of
object oriented programs.  As a component of an integrated set
of views, we are designing a display of the messages exchanged between
objects during the execution of a {\tt C++} program.  This section
describes the {\em Execution Mural}, focusing specifically on the
visual mechanisms used to provide navigation capabilities.   While the
current state of the design does not contain all the functionality
envisioned for this view, it does provide an effective demonstration
of our methods.  The techniques discussed in this section will be
generalized to other information spaces in Section 3.
```

Figure 14: Text editor containing LATEXdocument. Mural of the entire file is shown in the background of the scrollbar, with text colored according to section.

can be used to indicate attributes of the text, such as comments, sections, or keywords.

Fig. 14 is a sample text editor with a mural in the background of the scrollbar. Color is used to indicate sections in the LATEXdocument 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[17]. 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. Fig. 15a-c 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. 15a seems to be about visualization, and talks a little about object-oriented in the beginning. Fig. 15b talks about both visualization and object-oriented throughout the document, and Fig. 15c discusses object-oriented and visualization in the beginning and in the end. Miniature views such as these could be utilized in search applica-

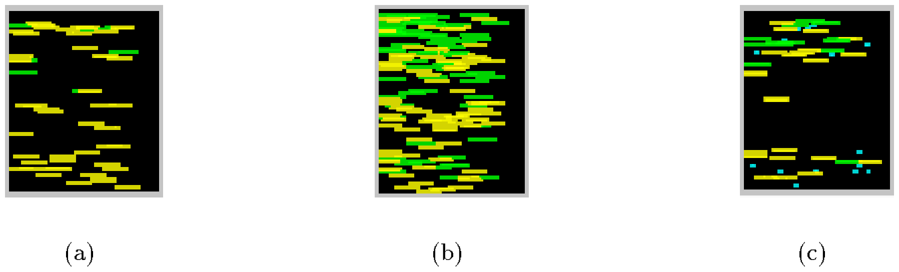


Figure 15: Murals showing keyword distribution for search on "visualization" (yellow), "object-oriented" (green), and "OO" (cyan) in three documents.

tions to display the results of a search and give users 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 grayscale tile images that correspond to a thematic breakdown of a document to visually display relevance information to a keyword search[13]. 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 x N` pixels and render it as Mural of `I x J` pixels. In addition to the data structures that store the original information, the algorithms require an `I x J` 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 2.4 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[9]. 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 `I x J` 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. 16 shows the computation for a pixel at (m,n) in the original image. Note that the algorithm does not consider the physical location of the area contributions to correctly render sub-pixel 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/I - 1) x (2N/J - 1)` to filter the original `M x N` binary image into `I x J` 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 $i, j$ set $mural\_array[i][j]$ to zero
2. for each pixel $m, n$ in the original representation
    a. compute $x = m/M * I$,  $y = n/N * J$
    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:
       $mural\_array[floor(x)][floor(y)]$
       $mural\_array[floor(x)][ceil(y)]$
       $mural\_array[ceil(x)][floor(y)]$
       $mural\_array[ceil(x)][ceil(y)]$
    d. update $max\_mural\_array\_value$ if one of the four new $mural\_array[][]$
       values is a new maximum
3. for each $i, j$ in the $mural\_array$
    a. map the value $mural\_array[i][j]/max\_mural\_array\_value$
       to a grayscale or color intensity varying scale,
       depending on the type of Mural being created
    b. shade the pixel at $i, j$ 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 photo-realistic 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 non-overlapping box filter of size `M/I x N/J` used to filter the original `M x 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 maximimum 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).

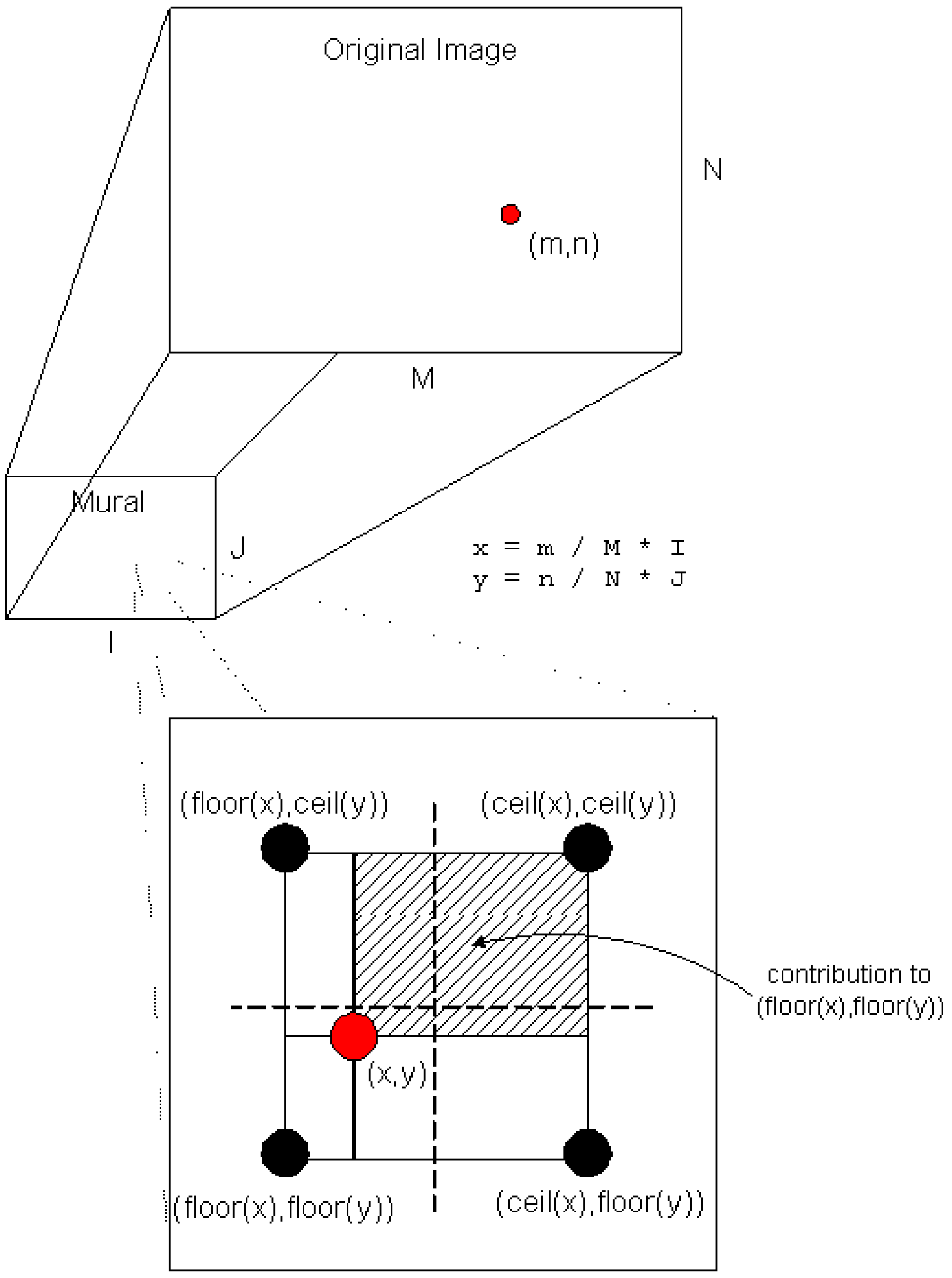


Figure 16: Information Mural algorithm example, scaling pixel at (m,n) in M x N original image to (x,y) in I x J Mural. Contribution of (x,y) to pixel (floor(x),floor(y)) is shown with diagonal cross-hatch.

```
1. for each $i, j$ set $mural\_array[i][j]$ to zero
2. for each pixel $m, n$ in the original representation
     a. compute $x = m/M * I$,  $y = n/N * J$
     b. add 1.0 to $mural\_array[floor(x)][floor(y)]$
     c. update $max\_mural\_array\_value$ if the new $mural\_array[floor(x)][floor(y)]$
        is greater than the existing maximum
3. for each $i, j$ in the $mural\_array$
     a. map the value $mural\_array[i][j]/max\_mural\_array\_value$
        to a grayscale or color intensity varying scale,
        depending on the type of Mural being created
     b. shade the pixel at $i, j$ 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, 5 of which are to be colored blue, 13 red, 6 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 anti-aliasing, 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 anti-aliased blue original pixels each contribute 0.1 intensity to a screen pixel and one anti-aliased 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 non-overlapping 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 [28]. 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.

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 resizeable global overview. This feature emphasizes the value of the Mural widget versus an application using rendering hardware such as the Open GL accumulation buffer[11] to do anti-aliasing of a scene as it is drawn.

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

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. Before moving on to the examples, 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 4 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 grayscale shading or density, and then the potential addition of color. Density is low in the ordering of elementary graphical perception tasks[6]. Distinguishing fine variations or levels of detail in grayscale 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[39]. 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[7]. Additionally, the work of Bergman, Rogowitz, and Treinish 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 grayscale 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[40]. 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 [29]. Other related work in the choice of color for data visualizations includes [12].

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 grayscale 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 100x100 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, car data, 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, Carr and Shneiderman describe many different styles of information browsers[30]. 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, Carr and Shneiderman 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 unover relationships within large data sets. The sun spot, river flow, and car data examples 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. Some, such as parallel coordinates, have even been combined with a Mural, as shown earlier. See [7] 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 US Census data example and, to a certain degree, the "words within a text file" example fall within this area. As discussed earler, 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 3-D 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 among 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.

Other related work

One area of related work on which the Information Mural is based is the fundamental antialiasing research of computer graphics[1, 9]. 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[10, 34]. 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[37] techniques, as a way of providing an integrated focus + context view.

Other information visualization techniques exist for presenting large data sets. The Cone Trees and Perspective Wall of the Information Visualization system[33] use natural 3-D perspective to generate a form of focus + context view. Treemaps[23] use a "slice-and-dice" rectangular region technique for visualizing information. The Table Lens[32] and SeeSoft[8] systems mentioned earlier in this article also introduce innovative visualization techniques. Fundamentally, however, all these information visualization techniques are best suited for structured, hierarchical data such 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[24] 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[36]. 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.

6 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 anti-aliasing 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.

7 Acknowledgments

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

References

[1] J. Barros and H. Fuchs. Generating smooth 2-d monocolor line drawings on video displays. In *Proceedings of the 1979 SIGGRAPH Conference*, volume 13, pages 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*, 9(6):451–466, 1990.

[3] R. A. Becker and W. S. Cleveland. Brushing scatterplots. *Technometrics*, 29(2):127–142, 1987.

[4] L. D. Bergman, B. E. Rogowitz, and L. A. Treinish. A rule-based tool for assisting colormap selection. In *Proceedings of the IEEE Visulaization '95*, pages 118–125, 1995.

[5] R. Chimera. Value Bars: An information visualization and navigation tool for multiattribute listings (demo summary). In *Proceedings of the ACM SIGCHI '92 Conference on Human Factors in Computing Systems*, pages 293–294, 1992.

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

[7] W. S. Cleveland. *Visualizing Data*. Hobart Press, Summit, NJ, 1993.

[8] S. G. Eick, J. L. Steffen, and E. E. S. Jr. SeeSoft—A tool for visualizing line oriented software statistics. *IEEE Transactions on Software Engineering*, 18(11):957–968, Nov. 1992.

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

[10] G. W. Furnas. Generalized fisheye views. In *Proceedings of the ACM SIGCHI '86 Conference on Human Factors in Computing Systems*, pages 16–23, Boston, MA, Apr. 1986.

[11] P. Haeberli and K. Akeley. The Accumulation Buffer: Hardware support for high-quality rendering. In *Proceedings of SIGGRAPH '90*, pages 309–318, 1990.

[12] C. G. Healey. Choosing effective colours for data visualization. In *Proceedings of the IEEE Visualization '96*, pages 263–270, San Francisco, CA, Oct. 1996.

[13] M. A. Hearst. TileBars: Visualization of term distribution in full text information access. In *Proceedings of ACM SIGCHI '95 Conference on Human Factors in Computing Systems*, pages 59–66, Denver, CO, 1995.

[14] M. T. Heath and J. A. Etheridge. Visualizing the performance of parallel programs. *IEEE Software*, 8(5):29–39, Sept. 1991.

[15] M. T. Heath, A. Malony, and D. Rover. Parallel performance visualization: From practice to theory. *IEEE Parallel and Distributed Technology*, 3(4):44–60, Winter 1995.

[16] M. T. Heath, A. Malony, and D. Rover. The visual display of parallel performance data. *Computer*, 28(11):21–28, Nov. 1995.

[17] W. C. Hill, J. D. Hollan, D. Wroblewski, and T. McCandless. Edit wear and read wear. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*, pages 3–9, May 1992.

[18] A. Inselberg and B. Dimsdale. Parallel coordinates: A tool for visualizing multi-dimensional geometry. In *Proceedings of the 1990 IEEE Visualization Conference*, pages 361–370, San Francisco, CA, 1990.

[19] D. F. Jerding and J. T. Stasko. Using visualization to foster object-oriented program understanding. Technical Report GIT-GVU-94-33, Georgia Institute of Technology, July 1994.

[20] D. F. Jerding and J. T. Stasko. The Information Mural: A technique for displaying and navigating large information spaces. In *Proceedings of the IEEE Visualization '95 Symposium on Information Visualization*, pages 43–50, Atlanta, GA, October 1995.

[21] D. F. Jerding and J. T. Stasko. Using Information Murals in visualization applications. In *Proceedings of the 1995 Symposium on User Interface Software and Technology (Demonstration)*, pages 73–74, Pittsburgh, PA, November 1995.

[22] D. F. Jerding, J. T. Stasko, and T. Ball. Visualizing message patterns in object-oriented program executions. Technical Report GIT-GVU-96-15, Georgia Institute of Technology, May 1996.

[23] B. Johnson and B. Shneiderman. Tree-maps: A space filling approach to the visualization of hierarchical information structures. In *Proceedings of the IEEE Visualization '91*, pages 284–291, San Diego, CA, Oct. 1991.

[24] D. A. Keim, H.-P. Kriegel, and M. Ankerst. Recursive Pattern: A technique fo visualizing very large amounts of data. In *Proceedings of IEEE Visualization '95 Conference*, pages 279–286, Atlanta, GA, October 1995.

[25] D. Kimelman and B. Rosenburg. Strata-Various: Multi-layer visualization of dynamics in software system behavior. In *Proceedings of the IEEE Visualization '94 Conference*, Oct. 1994.

[26] E. Kraemer and J. T. Stasko. The visualization of parallel systems: An overview. *Journal of Parallel and Distributed Computing*, 18(2):105–117, June 1993.

[27] J. Lamping and R. Rao. Laying out and visualizing large trees using a hyperbolic space. In *Proceedings of the 1994 ACM Symposium on User Interface Software and Technology*, pages 13–14, Marina del Rey, CA, November 1994.

[28] A. R. Martin and M. O. Ward. High dimensional brushing for interactive exploration of multivariate data. In *Proceedings of the 1995 IEEE Visualization Conference*, pages 271–278, Atlanta, GA, Oct. 1995.

[29] M. Monmonier. *How to Lie with Maps.* University of Chicago Press, Chicago, IL, 2 edition, 1996.

[30] C. Plaisant, D. Carr, and B. Shneiderman. Image-browser taxonomy and guidelines for designers. *IEEE Software*, 12(2):21–32, March 1995.

[31] B. A. Price, R. M. Baecker, and I. S. Small. A principled taxonomy of software visualization. *Journal of Visual Languages and Computing*, 4(3):211–266, Sept. 1993.

[32] R. Rao and S. K. Card. The Table Lens: Merging graphical and symbolic representations in an interactive focus+context visualization for tabluar information. In *Proceedings of the ACM SIGCHI '94 Conference on Human Factors in Computing Systems*, pages 318–322, Boston, MA, April 1992.

[33] G. G. Robertson, S. K. Card, and J. D. Mackinlay. Information visualization using 3D interactive animation. *Communications of the ACM*, 36(4):57–71, Apr. 1993.

[34] M. Sarkar and M. H. Brown. Graphical fisheye views of graphs. In *Proceedings of ACM SIGCHI '92 Conference on Human Factors in Computing Systems*, pages 83–91, May 1992.

[35] J. T. Stasko and E. Kraemer. A methodology for building application-specific visualizations of parallel programs. *Journal of Parallel and Distributed Computing*, 18(2):258–264, June 1993.

[36] J. T. Stasko and J. Muthukumarasamy. Visualizing program executions on large data sets. In *Proceedings of the 1996 IEEE Symposium on Visual Languages*, pages 166–173, Boulder, CO, Sept. 1996.

[37] M. C. Stone, K. Fishkin, and E. A. Bier. The movable filter as a user interface tool. In *Proceedings of the ACM SIGCHI '94 Conference on Human Factors in Computing Systems*, pages 306–312, Boston, MA, April 1992.

[38] B. Topol, J. T. Stasko, and V. S. Sunderam. Monitoring and visualization in cluster environments. Technical Report GIT-GVU-96-10, Georgia Institute of Technology, March 1996.

[39] E. R. Tufte. *The Visual Display of Quantitative Information.* Graphics Press, Cheshire, CT, 1983.

[40] E. R. Tufte. *Envisioning Information.* Graphics Press, Cheshire, CT, 1990.