The Information Mural:
Increasing Information Bandwidth in Visualizations

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-96-25
October 1996

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

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[18]. For many of these applications, 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+context simultaneously[25, 21, 5, 1, 2, 13, 17, 4, 24, 22, 23, 16, 27, 14]. 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 typically utilize abstraction, 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 result in a loss of information that might be useful to the observer.

Our technique increases the visual information bandwidth available to visualization applications. 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, irregardless of the size of the compressed representation.

There are several different types of information spaces which can be represented using information murals:

- 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 an average of the data values, or only a subset of the data.

- Time-oriented visualizations often span many computer screens if laid out completely. These types of views are particularly prevalent in software visualization[19] and monitoring applications.
- Visualizations which 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.
- 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 describes the Information Mural technique in detail. Following this, visualization applications which utilize Information Murals are presented, along with discussion of existing visualization systems which might take advantage of the technique.

2 Technique

Imagine some visual representation of a large information space, made up of distinct elements each with their own representation. An Information Mural of this information is to fit in some area of `I x J` pixels; assume there is a "bin" associated with each pixel. The position of each information element is first scaled to fit into the available space. As each element is "drawn" in the mural using an imaginary pen, different amounts of "ink" fall into different bins, in a manner similar to anti-aliasing strategies in computer graphics. As each subsequent element is drawn, the amount of ink will build up in different bins, depending on the amount of overlap of the elements.

The resulting Information Mural is created by mapping the amount of ink in each bin (the information density) to some visual attribute. In a *grayscale* mural, the shade of each pixel corresponds proportionally to the amount of ink in each bin. Instead of using grayscale variation, an *equalized intensity* variation over the entire color scale can also be used. With the *raindrop* mural, the amount of ink in each bin makes a "puddle" centered around that pixel, so pixels with more ink will appear larger. Color can then be added to the mural to convey other attributes of the informational elements, while still preserving the density mapping.

Information Murals of certain information spaces may be inappropriate. The distribution of information in the original image may be such that a useful Information Mural cannot be created. For example, a grayscale mural showing a graph of a symmetric function with a short period will be a dark bar with a thickness equal to the amplitude of the data.

2.1 Basic Algorithm

The basic algorithm for creating an Information Mural is listed below. The algorithm takes an image of `M x N` elements and scales it into a mural of `I x J` pixels. In addition to the data structures which store the information, the algorithm requires an `I x J` array of floats. The algorithm listed below does not handle attribute colors on top of the density representation.

```
1. for each $i,j$ set $mural\_array[i][j]$ to zero
2. for each element $m,n$ of information
     a. compute $x = n/N * J$, $y = m/M * I$
     b. determine the proportion of this point that lies in each of the four surrounding $mural\_array$
        entries (totals to 1.0):
        $mural\_array[floor(x)][floor(y)]$
        $mural\_array[floor(x)][ceil(y)]$
        $mural\_array[ceil(x)][floor(y)]$
        $mural\_array[ceil(x)][ceil(y)]$
     c. add each of the proportions determined in the previous step to the existing values of each
        corresponding $mural\_array$ entry
          i. update $max\_mural\_array\_value$ to keep track of the maximum $mural\_array[][]$ value
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, or to pixel size,
        depending on the type of mural being created
     b. color and draw the pixel at $i,j$ of the mural based on mapping
        computed in the previous step
```

For improved efficiency, steps 2b and 2c can replaced by a single step which adds 1.0 to $mural\_array[floor(x)][floor(y)]$ and updates the $max\_mural\_array\_value$. This avoids having to compute a number of floor's and ceil's and the percentages lying in each surrounding pixel, effectively eliminating the anti-aliasing aspect from the mural. For many applications, the gain in performance from using the aliased mural outweighs any slight changes in appearance.

2.2 Advanced 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, it should be recognized that bandwidth limitations imposed by each pixel mean that the mural may not be able to show attribute colors for every piece of data at the same time. While the Information Mural technique increases information bandwidth by changing intensity or size of pixels to represent density, a pixel is by definition a single color. What if the mural compresses 50 data points into the same pixel, 5 of which are to be colored blue, 13 red, 6 yellow, and so on–how should that pixel be rendered? It does not make sense to mix rgb values, because an observer might be confused if equal parts of red and green data values make a yellow pixel. Thus, we choose to color each pixel according to the attribute color that occurs most frequently at that point 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 pixel density mapping: the maximum for the resulting pixel color? the maximum of all colors? The only way that really makes sense is to treat the intensity at each pixel uniformly (cumulatively over all colors), and compute the mapping with respect to the maximum of intensity as is done in the basic algorithm.

This leads to the alternative for computing attribute colors that we have chosen to implement. To reduce space requirements, a single *mural_array* of floats is used to keep track of overall information density at each pixel. A list of shorts, one for each possible attribute color, is kept with each *mural_array* entry to record how many points of each attribute color have been drawn. The tradeoff here is that in keeping a single intensity value and a count of colors, we could end up with an inaccurate reflection of exactly how much of the intensity is due to each color. For example, five anti-aliased blue points each contributing 0.1 intensity to a pixel and one anti-aliased red point contributing 1.0 intensity would result in a blue pixel of 1.5 intensity. This problem only arises in building an anti-aliased mural, because when anti-aliasing is not done each point always contributes 1.0 intensity to a single pixel.

To solve the attribute color bandwidth limitation mentioned above, different options could be added to an implementation. For example, pixels could be rendered cyclically through all attribute colors which they contain, or selectively displayed if they contain a particular attribute color.

2.3 Implementation

While the previous sections on the Information Mural algorithm mentioned many implementation considerations, this section will discuss how Information Murals can be included in visualization applications.

We have implemented an Information Mural as an abstract widget which 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. Additionally, it can act as a global view for more detailed views by providing a "navigation rectangle" which 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 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 methods which 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. All of the drawing methods (`MuralDrawPoint()`, `MuralDrawLine()`, `MuralFrameRectangle()`, etc.) are passed coordinates in the application defined coordinate system. Whenever the `Mural` needs to be redrawn, it calls the application's `MuralRedrawNeededCB()` callback method. Additionally, whenever the navigation rectangle is moved or the `Mural` is zoomed by the user, the application's `MuralValueChangedCB()` and `MuralZoomedCB()` are called, respectively.

In this way, the application draws the Information Mural in its own coordinate space with respect to the information being displayed, and the `Mural` handles the rendering of the mural in whatever space it has on the screen. User's manipulations of the `Mural` widget are passed back to the application in the application-defined coordinate space as well. The next section gives many examples of applications using both stand-alone `Mural` widgets, and applications which use the `Mural` as a global view through which the user can navigate more detailed views.

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

---

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

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 [11]. The following discussion also mentions related visualization work which could take advantage of the extra information bandwidth provided by Information Murals.

3.1 Software Visualization

The Information Mural technique originated in our software visualization research into visualization of object-oriented (OO) program executions[9, 10]. Murals are currently being used in a suite of views to support program understanding during design recovery, validation, and reengineering tasks[12].

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 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 [26], 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.

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[10]. 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 a mural of the entire message trace, and a navigation rectangle indicating where the focus area fits in the entire 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 which 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 which motivated the work described in [12] 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 which shows a subset of the execution that can fit in

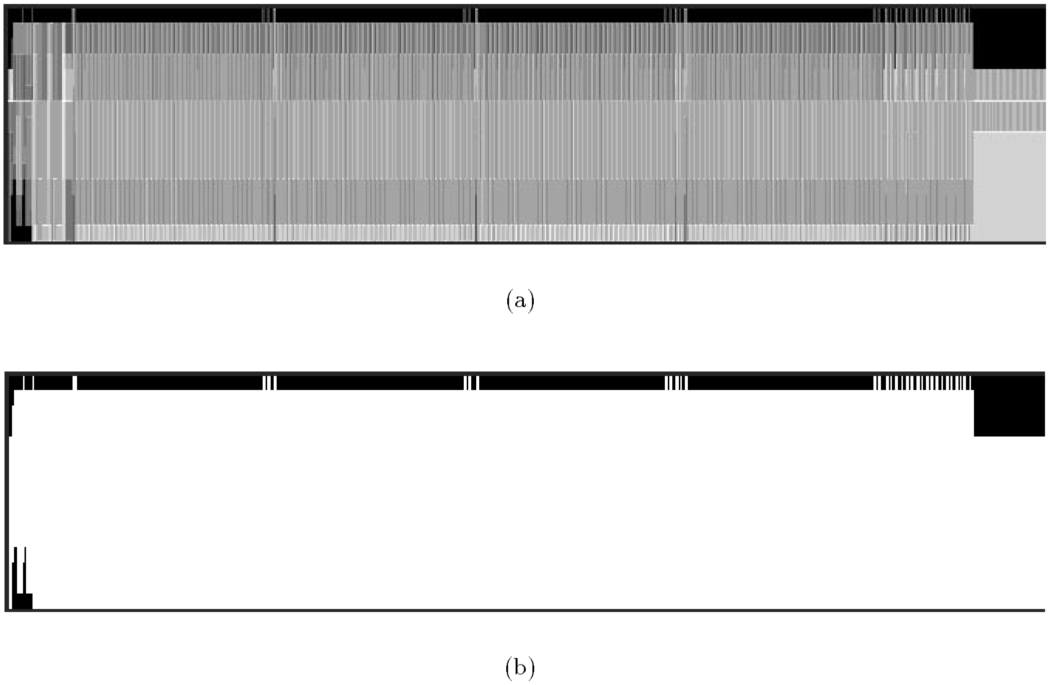


Figure 1: (a) Mural of object-oriented message trace of over 50,000 messages, drawn in an area 500 pixels wide. (b) Same diagram drawn by just over-plotting (without the mural technique).

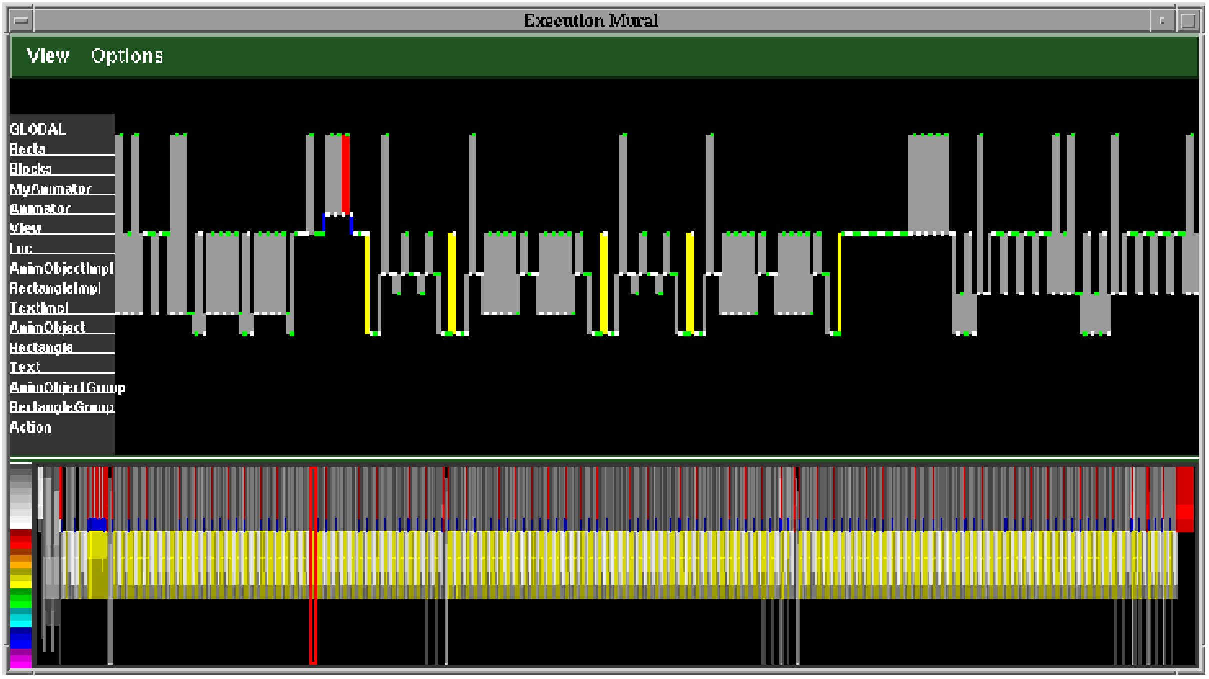


Figure 2: Execution Mural view of bubble sort algorithm animation built using the Polka animation toolkit.

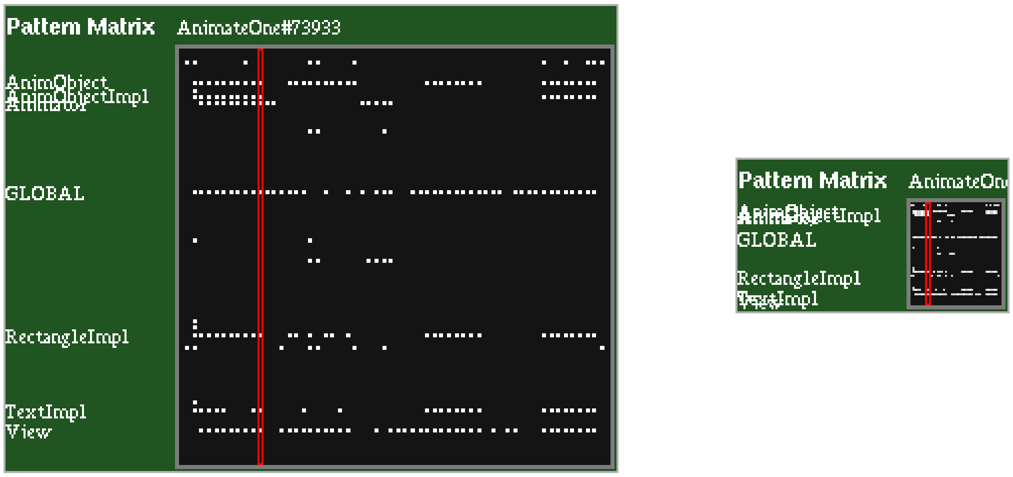


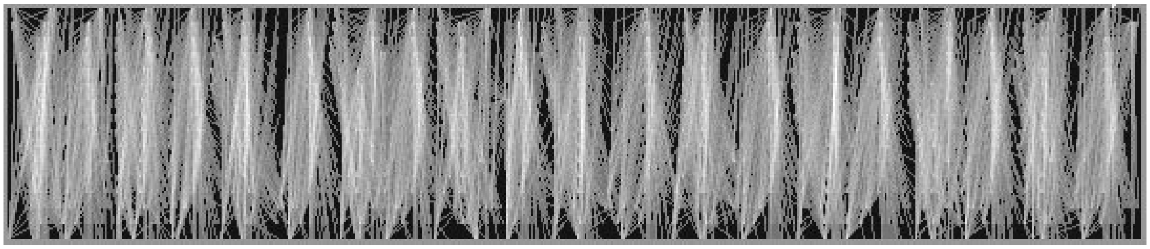
Figure 3: Pattern Matrix view of message patterns identified in an execution of the bubble sort algorithm animation, shown at two different sizes. Rows in the matrix are the classes in the program and columns are marked to indicate class membership in identified patterns. In the larger version, each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

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 and a scripting language to create simple program visualizations[16]. 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[15]. The time-oriented system and process state information views use pixel-level color strips which 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.

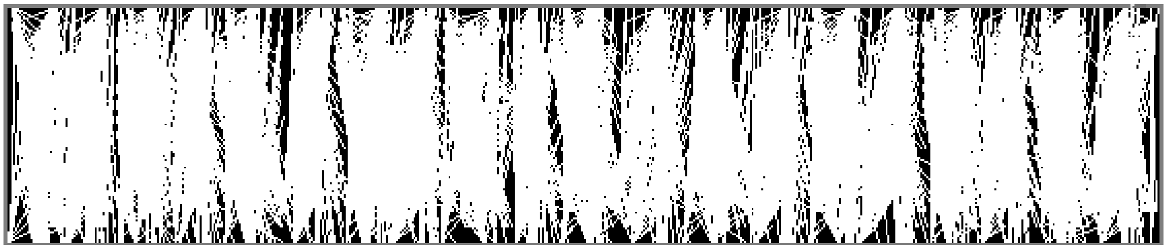
Another view in our OO visualization prototype shows how the Information Mural technique can be used to create scalable matrix views. Our visualization tools identify repeated message sequences in OO program executions. Information about these message patterns is displayed in the *Pattern Matrix* view. 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 which could take advantage of Information Murals in a similar way



(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).

is the Table Lens[20]. The Table Lens applies fish-eye viewing techniques to table-oriented data like a spreadsheet. By combining symbolic and graphical representations, the Table Lens can show various rows, columns, or cells at different levels of focus. When rows or columns are collapsed to their minimum size, they are allocated a single row or column of pixels. 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

Visualizations of the message passing during executions of programs on parallel architectures become very unclear when long durations of time are shown. The aliased mural of Fig. 4a shows the kernel integer sort benchmark executing on 16 `PVM` processors, generated using the PVaniM system built at Georgia Tech[28]. 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.

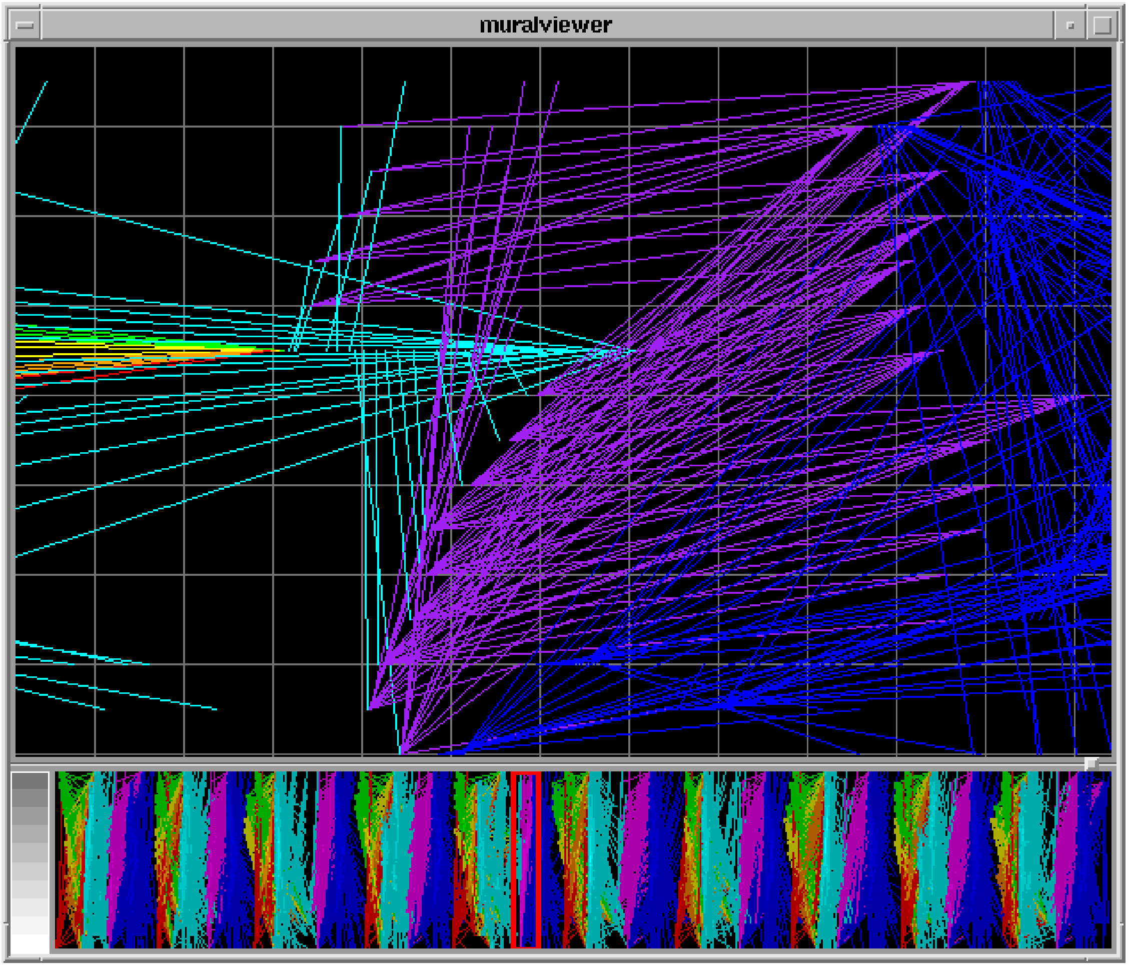


Figure 5: View of message passing in kernel integer sort parallel processor benchmark, with focus area and global overview created using the Information Mural technique.

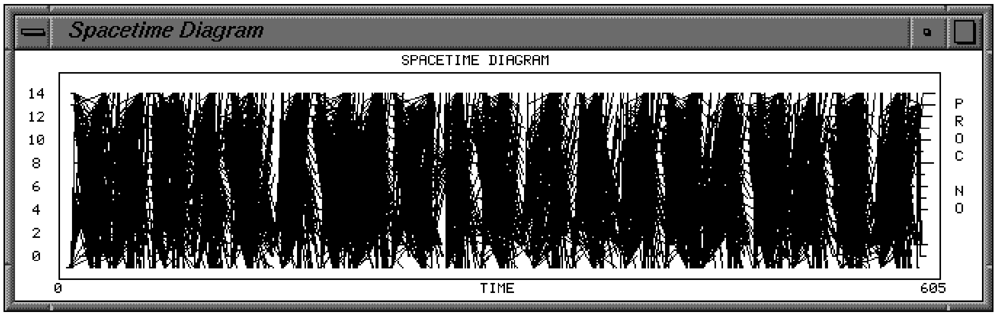


Figure 6: ParaGraph space-time view of message passing in kernel integer sort parallel processor benchmark.

The space-time view of the `PVM` kernel integer sort benchmark shown using ParaGraph[7], 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 which 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. 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. 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 do not have to worry about the size of the dataset. 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 which goes from dark blue (lowest data density) to bright white (highest data density) is used because it is easier to see outliers using color.

Plotting statistics such as averages is commonly done to analyze large amounts of data. However, in the monthly view we do not see the band of "missing" values between zero and about 10 sun spots, nor do we notice that a large number of zero values were recorded (bright spots at bottom of Fig. 8).

With the stand-alone 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

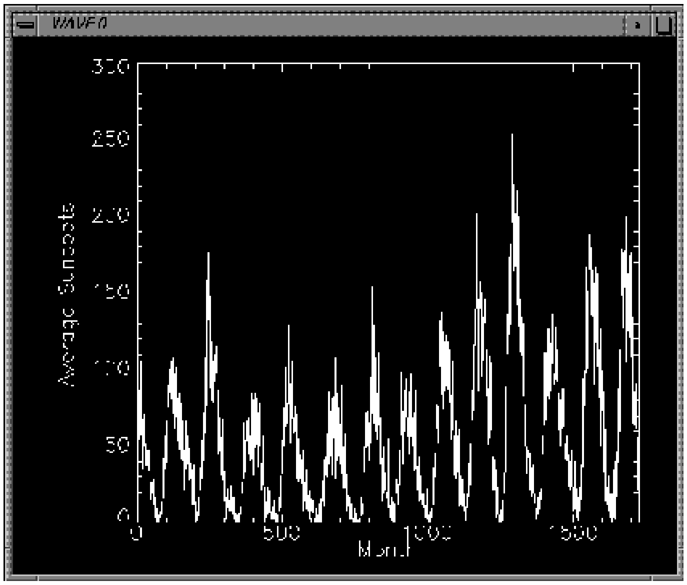


Figure 7: Plot of average number of sun spots recorded per month, 1850-1993.

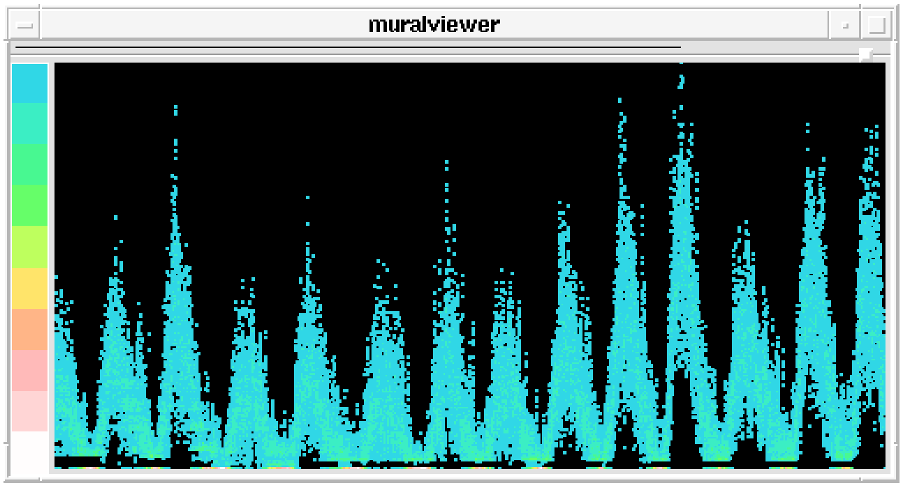


Figure 8: Mural of the number of sun spots recorded daily, 1850-1993.

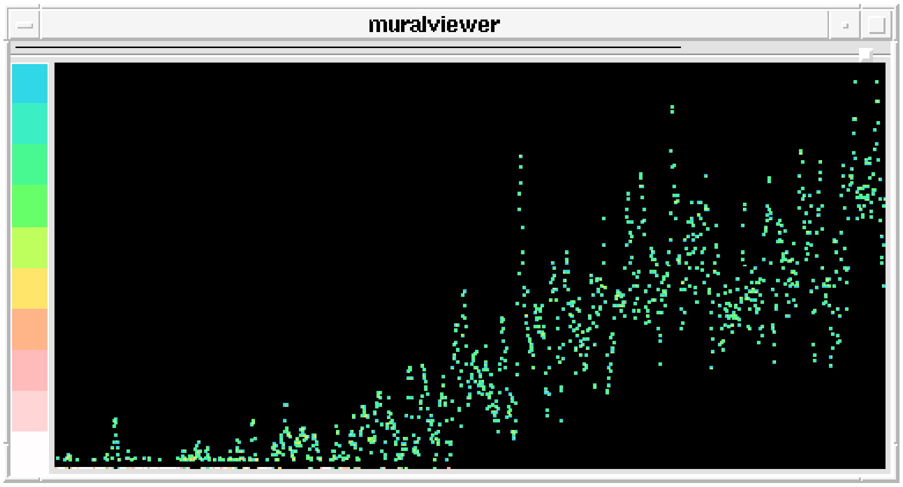


Figure 9: Mural of the number of sun spots recorded daily, 1850-1993, zoomed in on a small area.

background of a slider, giving context to a more detailed view of the data.

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

The Information Mural technique can be used to create 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 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).

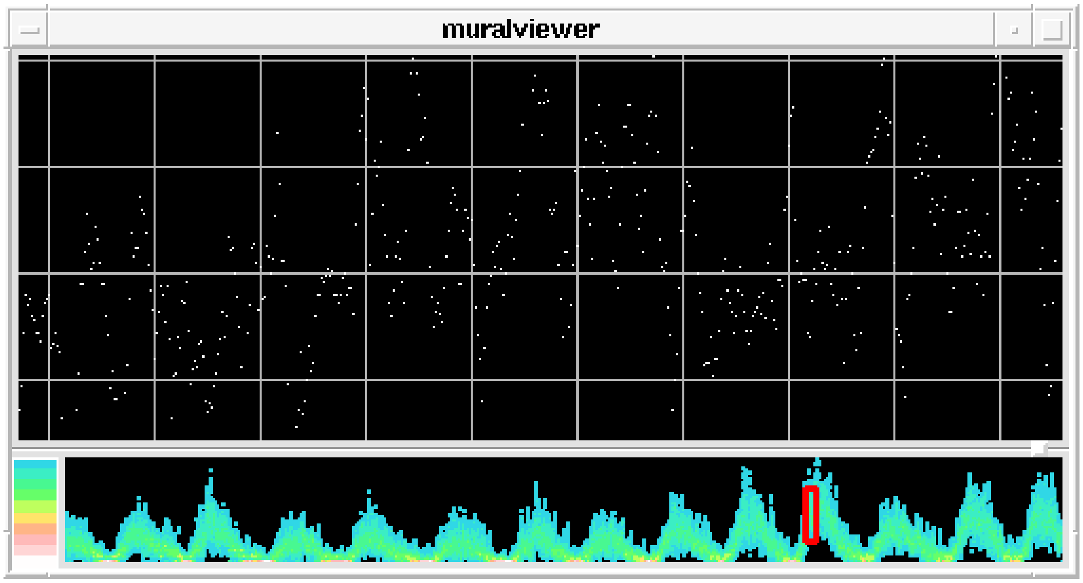


Figure 10: View of sun spots showing focus area and mural of entire data set at the bottom.

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

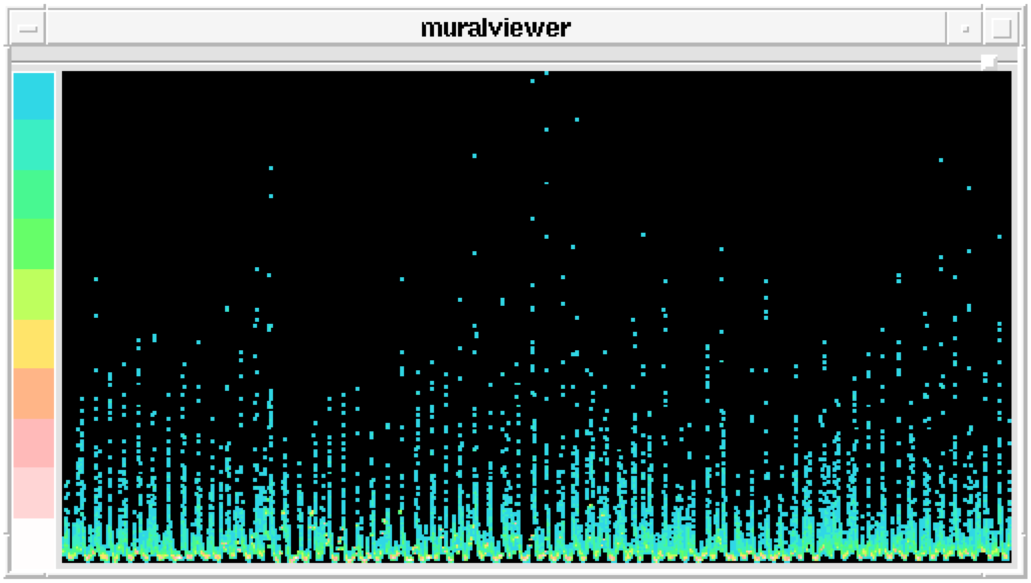
The U.S. Census Bureau creates maps of various census statistics such as population distributions. While their techniques work well for wall-sized maps, the overwhelming scale reduction to display the information on a computer screen causes their algorithm to produce inaccurate results. The Information Mural technique computes information density automatically, making the display of a population density map on a computer screen almost trivial (Fig. 13). The data was obtained from the Tiger Mapping Service U.S. Places File, created from the Census file STF-1A.

3.3.2 Text Documents

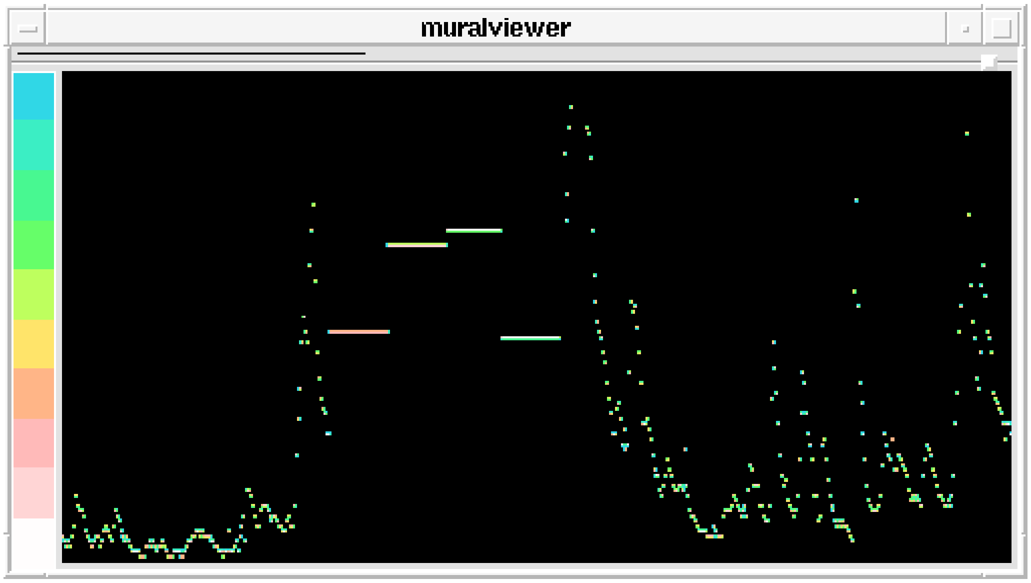
While SeeSoft[4] from AT&T's Bell Laboratories introduced a revolutionary miniature representation for text documents, it did have a limit. One row of pixels (or part of a row in later versions) is required for every line in the file. The Information Mural technique can go beyond this limit, allowing 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 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 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 in-

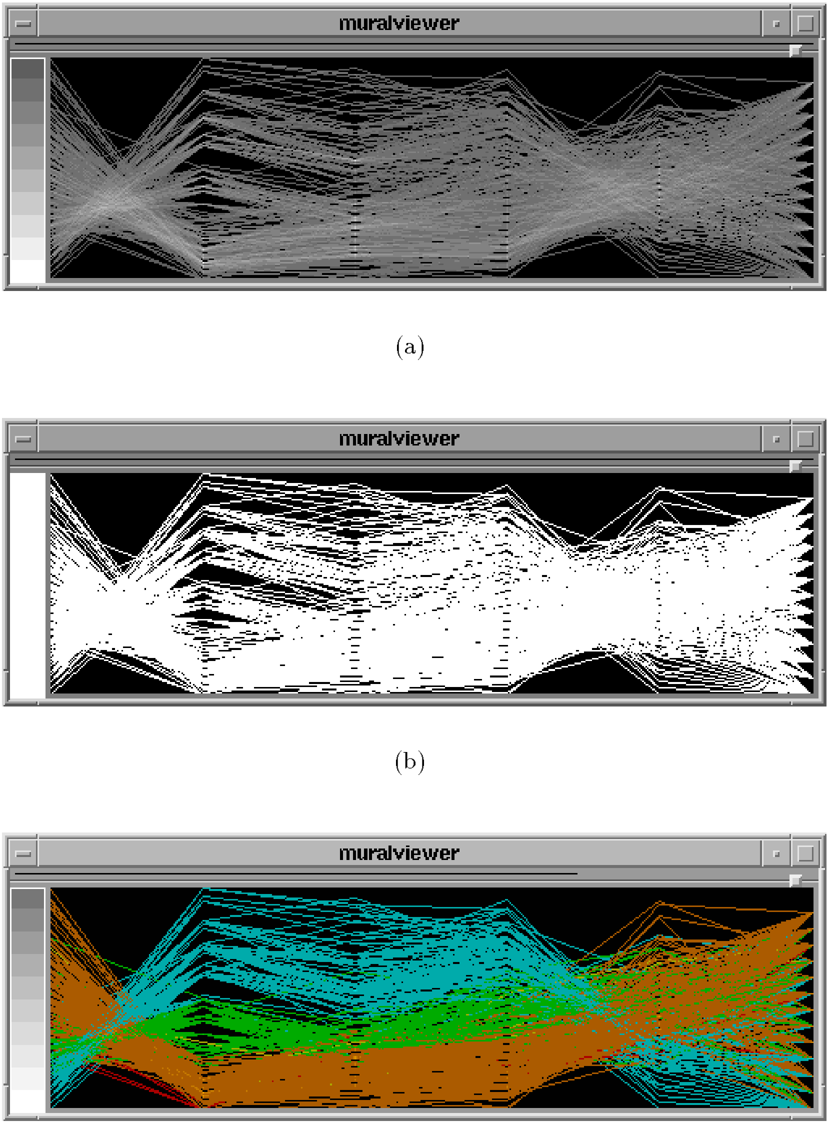


(a)



(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).

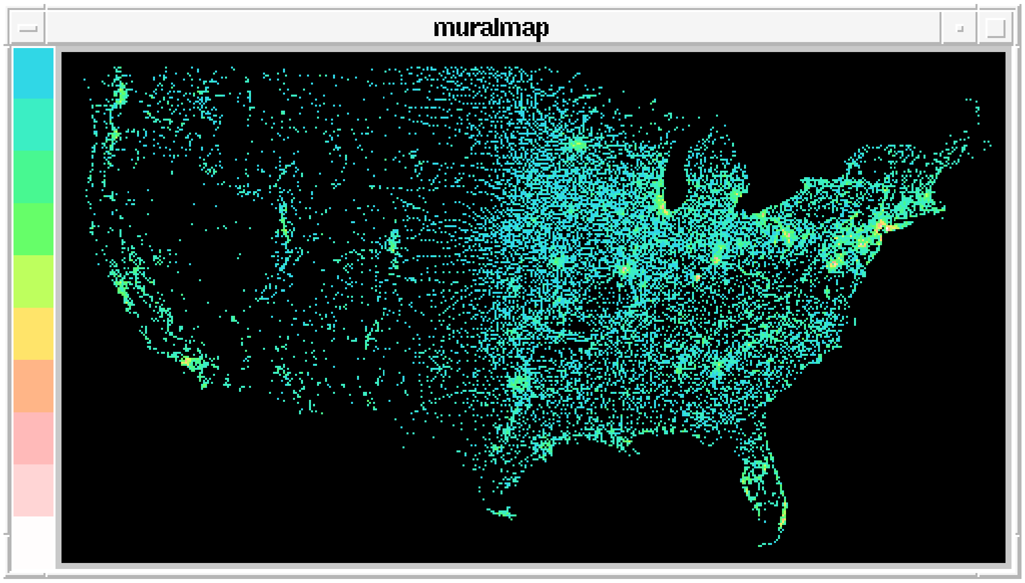
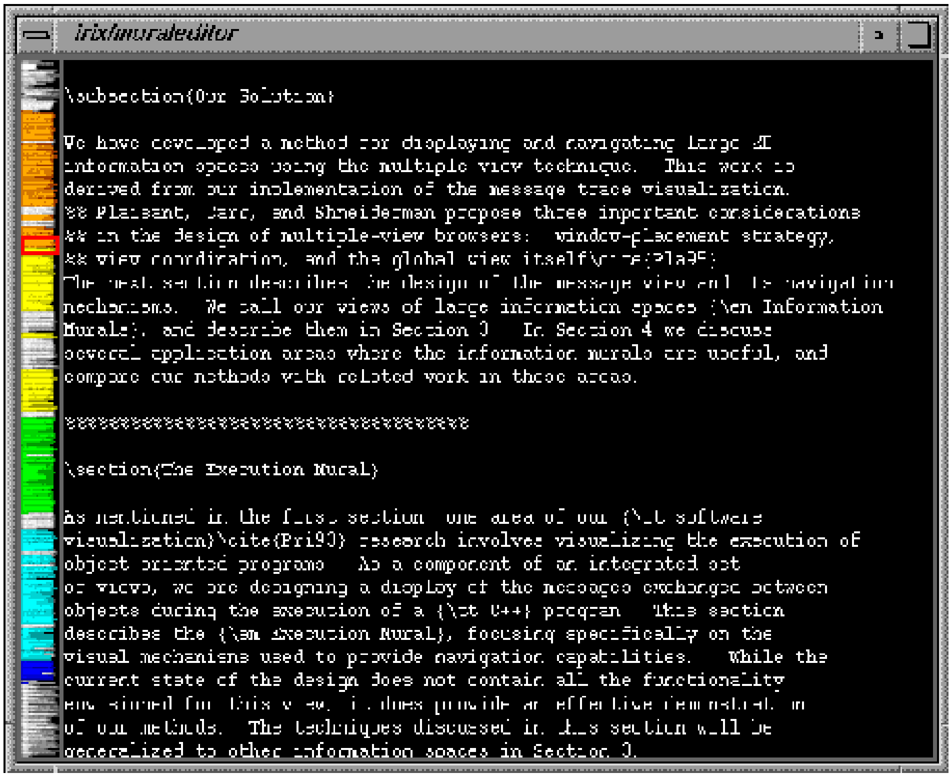
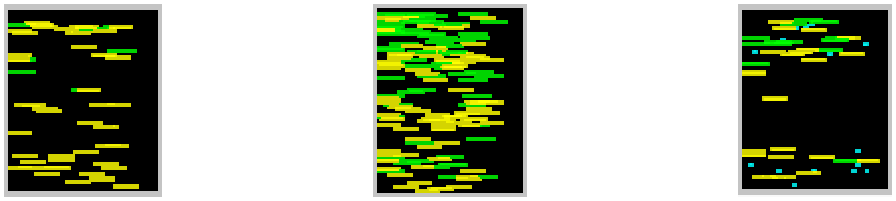


Figure 13: Mural of population density distribution, using data from the 1990 census.



```
\subsection{Our Solution}

We have developed a method for displaying and navigating large 2D
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 those 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 Latex document. Mural of the entire file is shown in the background of the scrollbar, with text colored according to section.



(a) (b) (c)

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

formation 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[8]. 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 which 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[3].

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 applications 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 which correspond to a thematic breakdown of a document to visually display relevance information to a keyword search[6]. 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 Conclusion

An Information Mural is a 2D, graphical representation of a large information space which fits entirely within a display window or screen. The miniature representation is drawn using anti-aliasing compression techniques and intensity shading or varying pixel size, 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 always useful to start with a global overview of the information. Information Murals 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. Information Murals increase the visual bandwidth available to visualization applications. 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.

References

[1] D. V. Beard and J. Q. W. II. Navigational techniques to improve the display of large two-dimensional spaces. *Behaviour and Information Technology*, 9(6):451–466, 1990.

[2] S. K. Card, G. G. Robertson, and J. Mackinlay. The Information Visualizer, an information workspace. In *Proceedings of the ACM SIGCHI '91 Conference on Human Factors in Computing Systems*, pages 181–188, New Orleans, LA, May 1991.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] J. Mackinlay, G. G. Robertson, and S. K. Card. The Perspective Wall: Detail and context smoothely integrated. In *Proceedings of the ACM SIGCHI '91 Conference on Human Factors in Computing Systems*, pages 173–180, New Orleans, LA, May 1991.

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

[19] 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.

[20] 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.

[21] S. P. Reiss. PECAN: Program development systems that support multiple views. *IEEE Transactions on Software Engineering*, SE-11(3):276–85, March 1985.

[22] S. P. Reiss. A framework for abstract 3D visualization. In *Proceedings of the 1993 IEEE Symposium on Visual Languages*, pages 108–115, Bergen, Norway, Aug. 1993.

[23] G. G. Robertson and J. D. Mackinlay. The Document Lens. In *Proceedings of the 1993 ACM Symposium on User Interface Software and Technology*, pages 101–108, Atlanta, GA, Nov. 1993.

[24] 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.

[25] R. Spence and M. Apperley. Data base navigation: an office environment for the professional. *Behaviour and Information Technology*, 1(1):43–54, 1982.

[26] 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.

[27] 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.

[28] 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.