An Environment for Empirical Data Interpretation

Gudrun J. Klinker
Cambridge Research Lab, Digital Equipment Corporation
One Kendall Square, Cambridge, MA 02139

Jan 7, 1993

Abstract

While several visualization systems have recently been developed, many application programmers still prefer writing their own code. Why? Because these systems are ill-suited for applications based on empirical data, such as biomedical imaging, robotics, seismic data analysis, and wood classification. Current visualization systems focus on displaying data. But emprical data analysis also needs highly customizable data exploration and interpretation tools which can be adapted to the requirements of different applications.

This paper presents EDI, an environment for Empirical Data Interpretation. EDI provides the following capabilities essential to data exploration: (1) Users can probe the data, defining regions of interest with arbitrary shapes. (2) The selected data can be transformed and displayed in many different ways, e.g., several data sets can be displayed together – merged, mixed or overlaid – or they can be shown side-by-side with linked cursors. (3) Linked cursors can be established between several windows showing data sets with arbitrary relationships, e.g., between differently sized images or between images and scatter plots. (4) Data can be displayed on any screen across a computer network, allowing for tele-collaboration arrangements with linked cursors around the world. (5) EDI is user-extensible, allowing programmers to change any component of EDI while keeping the remaining functionality. We demonstrate how EDI can be used in several applications.

Contents

1 Introduction

Several systems for visualizing data have recently been developed with the intent to free users from the burden of graphics programming [10, 11, 17, 18, 19, 20, 21, 22, 24, 25, 26, 27, 30, 31]. Why do many application programmers still prefer writing their own visualization code over using such visualization systems? In part, this may be the case because applications in biomedical imaging, robotics, seismic data analysis, and wood classification require highly interactive user-customizable tools to explore and interpret empirical data [16].

Stand-alone systems, such as EXVIS [10], KBVision [31] and PVWave [18], have provided various data exploration tools. Yet, such systems become hard to adapt to new aspects of visualization research, due to their monolithic structure. To provide flexibility, other systems, like AVS [30], offer visual programming interfaces with which users can assemble their own networks of communicating programs (*modules*). When an output port of one module is connected with the input port of another module, data can flow between the modules. Large libraries of modules are available and can be extended with user-provided modules as needed.

Despite such flexibility, these new systems are not well suited for data exploration because they lack essential interactive capabilities. For example, the X-based [28] AVS module to display images encapsulates all handling of user interaction within the module. No feedback is provided to other modules. As a result, application programmers currently cannot exploit the full interactive power of X in their own AVS modules unless they write their own data display facilities. This missing feedback channel is crucial. Users often need to probe the data and define regions of interest. They also need to see the selected areas in relationship with other data. In a typical scenario, a radiologist in a hospital looks at series CT scans of a patient, all shown side-by-side on a screen. When the radiologist selects an area in one of the scans, the corresponding areas in the other images are highlighted. Such interactivity involving several windows cannot be achieved using the current programming style of AVS.

Current systems also dilute the distinction between two concepts, data representation (storage) and data presentation (display). Typically, systems display images as images, volumes as volumes, histograms as graphs, image sequences as movie loops, and geometry as shaded surfaces or contours. Users, however, often need to see the same empirical data in many different

views or combine multiple data sets into a single view. It is important that users be able to define mappings between the dimensions of a data set and a set of available display capabilities [4, 12, 17]. Systems must also be able to mix several data types in a single view, such as geometric data and array data.

This paper describes EDI, an environment for Empirical Data Interpretation designed to help users analyze empirical data interactively. EDI uses the network editor and the data flow model of AVS – thus benefitting from its visual programming capabilities. But EDI changes AVS in most other respects. It provides an open feedback channel from the user to any EDI module. Its new X-based *display data* module makes user interaction available through an output port to other modules.

EDI's concept of an open feedback channel to arbitrary modules has far-reaching consequences – both for system design and for system use. Modules which use the feedback channel need to be informed of all transformations that have been applied to the data. Because EDI applications consist of networks of independent modules, knowledge of such data transformations cannot be encapsulated in a single module. To ensure the correct interpretation of user input, EDI modules send a *logbook* record along with the data in which the modules record all coordinate transformations, as well as other descriptive information about the data. When a module receives a cursor positioning event from the *display data* module, it uses the *logbook* record to relate the window position to the correct pixel position in the original data set.

EDI's feedback channel can be used to attach arbitrary data exploration or data interpretation routines to cursor positioning events from a window. It can also be used to establish linked cursors between any number of windows. Furthermore, the mechanism lays the foundation for very flexible telecollaboration arrangements across computer networks, since EDI-windows with linked cursors can be sent to any accessible display. Colleagues can thus annotate the same data at both ends. The *display data* module adapts its display mechanism to the capabilities of the respective frame buffers such that users can either enjoy the capabilites of special-purpose, high-resolution, full-color frame buffers or view the data on the less sophisticated screen of their local workstation.

EDI also provides tools to mix, match and merge data from one or more data sets in various views. EDI divides the data presentation process into two stages, starting with tools to extract and merge dimensions from data

sets, followed by a stage which maps the selected dimensions onto display mechanisms. EDI's *display data* routine finally displays both array data and geometric data, overlaying geometric data on images. As a result, users can merge information from several sources, as well as break up information from a single entity into several partial views, mixing and matching data of different data types.

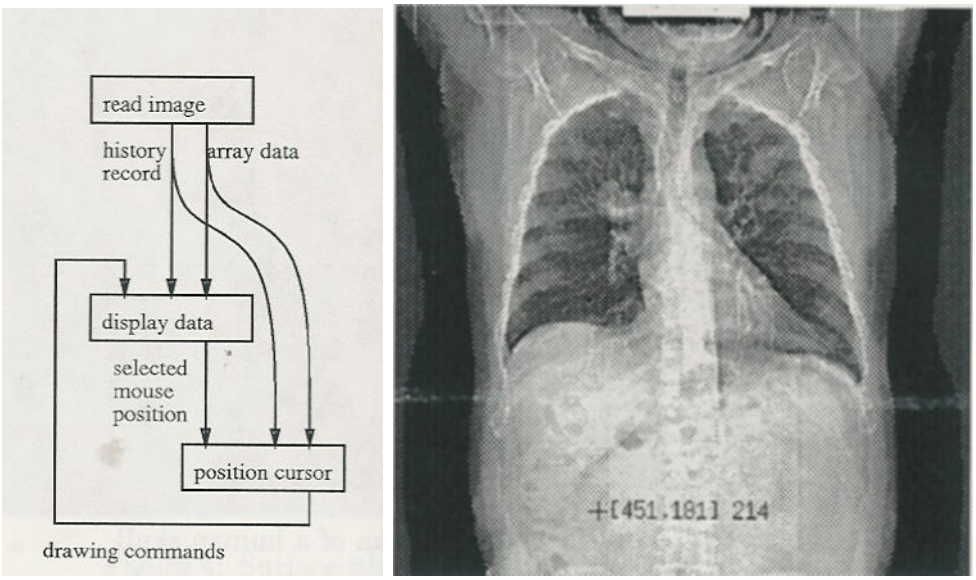

Figure 1: Basic example of an EDI network

Figure 1 shows a very basic EDI network which demonstrates how EDI can be used in data exploration. The network consists of three modules. After the first one has read an image, it sends it to the second module to display the data. When a user selects a pixel in the image with the mouse, the module sends the position to the third module which uses the position and the *logbook* record to retrieve the correct pixel from the original data set. The correct element is determined by inverting all geometric transformations that have been recorded in a homogeneous transformation matrix in the *logbook* record. The *position cursor* module then sends geometric drawing commands which describe the current cursor position back to the *display data* module. When the *display data* module receives the geometric data, it applies the same homogeneous transformations to the geometric data that were applied to the displayed image and thus overlays it at the correct position on the

image. The right half of Figure 1 shows how this network can be used to probe the density values in a CT image of a human chest.

The following sections follow the stages data sets will typically encounter as they flow through an EDI network. In Section 2, we start with techniques to extract and reconfigure data elements into suitable entities. Next, we introduce EDI's data presentation and display modules in Section 3. Section 4 shows how EDI can be used for data exploration tasks, demonstrating powerful cursor linking mechanisms. Section 5 presents tele-collaboration capabilities of EDI. We close with a discussion of the overall benefits and current limitations of EDI.

2 Data extraction

Many applications exist which need to interprete high-dimensional data sets in various – not necessarily geometric – formats [3]. Many applications also operate on more than one data set. Examples are multi-modal data (MR and CT data or surface seismic and well-to-well seismic data), and combinations of original and processed (filtered, segmented, transformed, histogrammed) data. It is important that users be able to define mappings between dimensions of data sets and sets of available display capabilities [12, 17]. Most current visualization systems have very rigid mapping mechanisms focusing on two-dimensional images, three-dimensional volumes, and geometric data [4]. This practice dilutes the distinction between two concepts, data storage and data presentation. EDI maintains a clear distinction between the concepts, separating data extraction and data presentation into separate stages. This section describes the data extraction stage in which users reconfigure the data space into one or more hyperspaces by extracting data along suitable dimensions. The next section will present various techniques for displaying the extracted dimensions.

2.1 What is the dimensionality of a data set?

Typically, images and volumes are considered to be two-dimensional and three-dimensional data sets because they are stored as arrays of such dimensionality. However, this is not quite correct. For a typical black-and-white image, every pixel is a triple (x, y, i) and an image thus has two geometric plus one spectral dimension. Similarly, volumes have three geometric dimen-

sions and one spectral dimension while vectors consist of one geometric and one spectral dimension. Color volumes, images, and vectors consist of tuples $(x, .., r, g, b)$, with one, two or three geometric dimensions plus three spectral dimensions. Time sequences, further spectral bands, multiple camera positions and changing sensor parameters (focal length, aperture) add more dimensions to the measurement space: $(t, \lambda_i, x_{cam}, y_{cam}, z_{cam}, f, a, x, y, z, r, g, b)$.

Aside from data sets with inherent geometric dimensions, there are many high-dimensional collections of non-geometric data, such as census data, financial data, and statistical measurements. The dimensionality of such data sets is defined by the number of variables per element (data record).

2.2 Dimensionality reduction

To ensure a flexible mapping between the dimensions of a data set and the data presentation capabilities of a display, we need mechanisms which extract user-defined hyperspaces from the original data set. Hyperspaces can be constructed in many different ways. EDI's goal is to provide tools to the user to extract data along certain dimensions, repackage the data, and then send it along to other EDI modules capable of presenting the data in the desired style. Quite a few different concepts for projecting data onto hyperspaces exist, such as:

- Data slicing along orthogonal or arbitrary directions: Elements are selected from a particular orthogonal or arbitrary slice through the data set. Non-orthogonal directions are linear combinations of several dimensions. Possible extensions may allow non-linear combinations of dimensions, e.g, to define spherical projections.

- Data accumulation with translucency: Elements are integrated along an arbitrary direction. If the data values are convolved with a translucency function, various structures in the data can be selectively emphasized. Possible extensions may allow multiplicative rather than additive accumulation schemes – which would be suitable for integrating probabilistic data sets. The data accumulation function may also take the position of a data element into account to provide depth cues.

- Data histogramming: The elements are counted rather than integrated along the projection ray. For example, the color histogram of a color

image is a projection from five dimensions (x, y, r, g, b) to four dimensions, $(r, g, b, count)$, along dimensions x and y.

- Data projection of extremal values: The element with the maximal or minimal value along the projection ray is selected.
- Data projection of iso-surfaces: The first element with a value within a specified range is selected.

EDI provides several data extractions mechanisms, and we plan to add more in the future.

2.3 Data mapping

When a dimension of a data set is mapped onto a specific display mechanism, it typically needs to be adapted to the limited capabilities of the hardware. In particular, unbounded or high-precision dimensions[1] of a data set need to be adjusted to the specified geometric and spectral ranges of the display device. Such mapping can take various forms: linear or non-linear scaling, subsampling or interpolation, cropping or thresholding with a specified maximum and minimum, wrapping modulo a specified value, substitutions according to a user-defined lookup table, random substitutions, permutations, and combinations of all of these techniques. Many such concepts are well-established colormapping techniques. Yet, the concepts can apply equally well to other dimensions of a data set, providing concepts, such as thresholding (cropping) and non-linear scaling (warping) to geometric dimensions or time.

3 Data presentation

Essential to data visualization are the display capabilities of the system. The capabilities can be exploited in various ways, depending on the visualization needs of the user, which in turn depend on the application and on personal preferences. They can vary over time as the data exploration and data interpretation focus changes.

This section presents a list of data presentation tools that EDI provides along different dimensions of display capabilities. Stand-alone systems, such

[1] such as the *count*-dimension of a histogram or a probabilistic value (x, y, p) which describes the error associated with a sensing device

Figure 2: Three-dimensional CT-scan of a human skull

as EXVIS [10], have explored various presentation styles like icons or sound generation in great depth. Yet, such systems become hard to adapt to new aspects of visualization research, due to their monolithic structure. In contrast, systems with a visual programming interface, like EDI and AVS, provide various data extraction and presentation tools as separate modular components which can be exchanged easily. In comparison to AVS, EDI's data presentation modules can all be mixed and merged such that users can generate detailed overlays which show several presentation forms, both geometric and intensity-based, together in a single view. Section 4.3 presents several examples.

- Intensity- and color-based displays: Up to four pieces of information per pixel, (x, y, z, i), can be shown in an intensity image, three geometric variables and one spectral variable. Up to six values are presented at each pixel (x, y, z, r, g, b) in a color image. Figures 1 through 3 show different examples of intensity and color-based displays: a human chest (x, y, i), a projection of a three-dimensional data volume (x, y, z, i) of a human skull [7, 13], and a color image (x, y, r, g, b) of a scene with several plastic objects [15].

- Graphs: Up to three values (x, y, i), can be plotted as unicolored lines

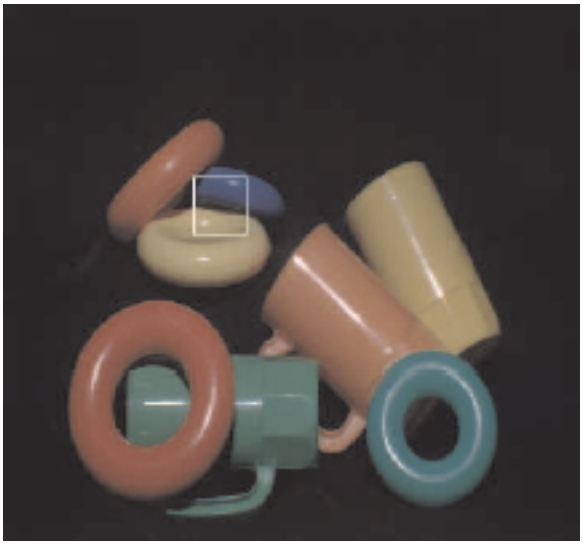

Figure 3: Color image of scene with plastic objecs

or surfaces over a one- or two-dimensional geometric geometric base, $i = f(x, y)$. More dimensions can be presented as families of graphs [2], in different colors or line drawing styles, or with labels. Some systems also map further information as colored texture onto individual surfaces [12, 18]. Figure 4 shows red, green and blue surface plots of the color pixels from the yellow and blue rings in Figure 3 (white rectangle). Two plateaus represent the shaded portions of the blue and yellow rings. The spikes correspond to highlights. Note the small ridge in the valley between the two plateaus. It is caused by interreflection between the two rings and is very hard to notice in the intensity-based presentation style of Figure 3. This surface-based rendering facilitates the detailed study of the color variation within the interreflection area.

- Icons can be arranged geometrically in one, two or three dimensions. Their shapes can represent many variables. Simple vectors present two-dimensional information. Higher-dimensional icons, including stick figures[10] and face drawings[9], have been designed and used successfully in several applications. The left picture in Figure 5 shows the yellow and blue rings from Figure 3 as an array of normalized color vectors. The vectors are oriented such that a pure red vector points straight upwards, green points 120^o to the right, and blue is at 120^o to

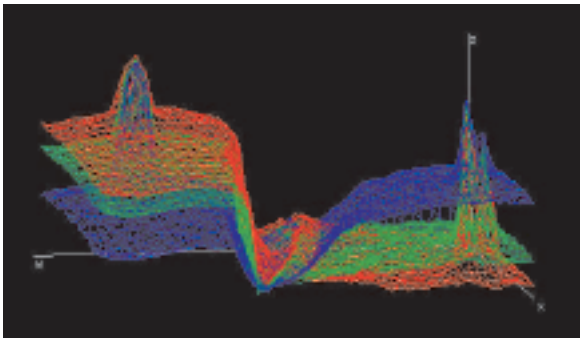

Figure 4: Surface plot of color variation in plastic scene

the left. White is represented by a vector pointing directly toward the viewer. The color changes in the interreflection area between the two objects are quite visible.

- Printed numbers: Printouts of high-dimensional data sets can be arranged as a long, sequential table, as two-dimensional arrangements of small blocks showing several data values per pixel, or as hybrids of the above. The right picture in Figure 5 shows the individual pixel values of a small subarea from the left picture, covering the interreflection area.

- Time: Many applications exploit the benefits of motion parallax by iterating through image sequences at interactive or real-time speeds. The image sequences are either computed on-the-fly or precomputed and recorded as movie loops. EDI provides a movie loop and a blink comparator [8] with which users can align consecutive images interactively. EDI's blink comparator consists of three modules (see Figure 6). One applies the transformations to one of the images, the second alternates rapidly between the two images, and the third is the *display data* module.

- Interleaved dimensions: If three geometric dimensions are not enough for laying out the information of the data set, more dimensions need to be interleaved. The data along a subset of dimensions can either be grouped in blocks and several such blocks are arranged side-by-side, or the data can be merged on a pixel by pixel basis, enlarging each pixel position into a small $n \times n$ region. Instead of laying the data at every

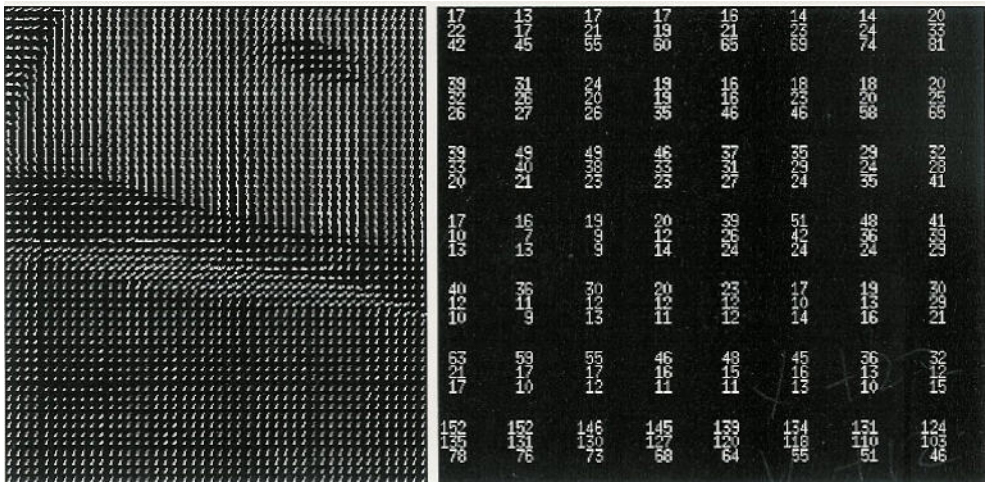

Figure 5: Vector icons and printed numbers showing color variation in plastic scene

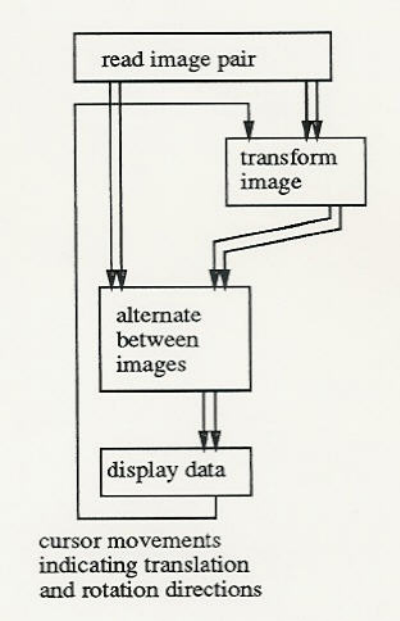

Figure 6: Network of a blink comparator

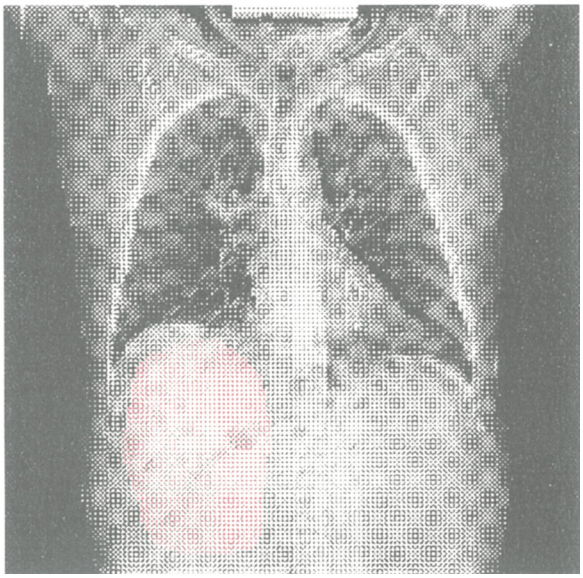

Figure 7: A black-and-white CT-scan of a human chest, interleaved with a red mask indicating a ficticious interesting region

pixel position out in an enlarged region, it can also be accumulated, averaged, maximized, masked, or alternated in a checker-board pattern. Interleaving can be performed along non-geometric dimensions, such as color (3 color bands) or time (movie loops). EDI has a module which merges up to three graytone images into a color image and a module which interleaves up to four images pixelwise in local 2×2 areas. Figure 7 shows how the black-and-white CT scan of a human chest can be interleaved with an image mask showing a ficticous interesting area.

- Audio: Although data visualization started out by considering visual display properties, accoustic data presentation has found its way into visualization environments [5, 14, 29]. Sonification of data provides the potential for presenting many more dimensions of information. Current studies towards including such capabilities into EDI are in progress[23].

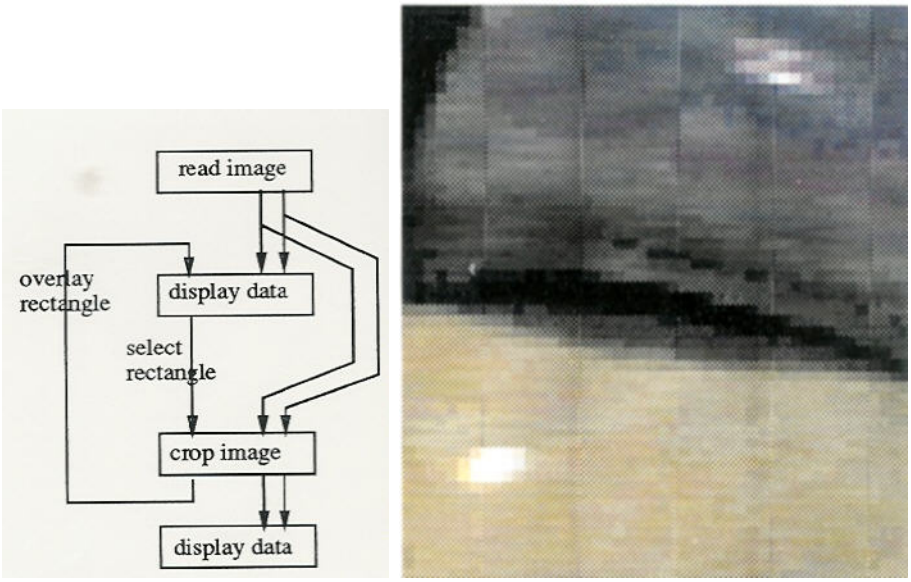

Figure 8: Network to select a region of interest and a selected, zoomed region from the plastic scene

4 Data exploration

For the analysis of empirical data sets, it is essential that the data be interactively explored rather than merely viewed. Among the necessary data exploration operations are mechanisms which allow interactive data probing, as well as arrangements to visualize data elements in context by establishing visual relationships between different parts of the data. This section describes such mechanisms in EDI.

4.1 Interactive data probing

Due to the open design of EDI's *display data* module and EDI's flexible data extraction and presentation concepts, users can interactively set up many different data probing arrangements.

Figure 1 in section 1 has already shown a very basic data probing arrangement: a network of three modules which read an image, display it, and overlay a software-cursor on top of the image. The left picture in Figure 8 extends this network, showing how the same mechanisms can be exploited to create a setup in which a zoomed copy of a small part of the image is shown

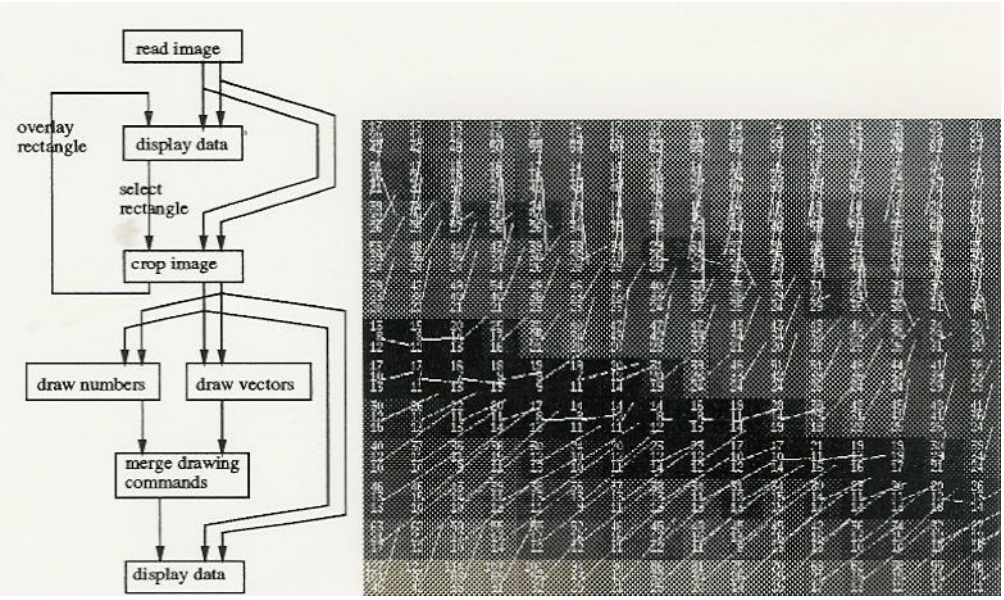

Figure 9: Network to overlay numbers and vectors on a zoomed image of a region of interest, as well as a picture showing a small region of interest of the plastic scene as an image with vectors and numbers

in a second window. The cursor position in the original window determines which part of the image is shown in the zoomed window. The cursor position is sent to a *crop image* module which uses it as the center point of a rectangular area of user-definable width and height. The cropped data is sent to a second *display data* module which generates the second window. The data then is zoomed in this window via interactive window resizing[2]. The right picture in Figure 8 shows the zoomed region of interest that was selected from the picture in Figure 3, as indicated by the white rectangle.

A small extension to the zooming and cropping network provides yet another level of functionality. The left picture of Figure 9, includes two new modules which print the color pixel values and draw normalized color vectors of each data element in the cropped image area, as described in sections 3 and 3. When the output of the two modules is merged and overlaid on the zoomed image data, the result is a rather sophisticated and detailed data presentation scheme. In EDI, users can thus change the complexity of the

[2]The *display data* module automatically zooms the data according to the window size

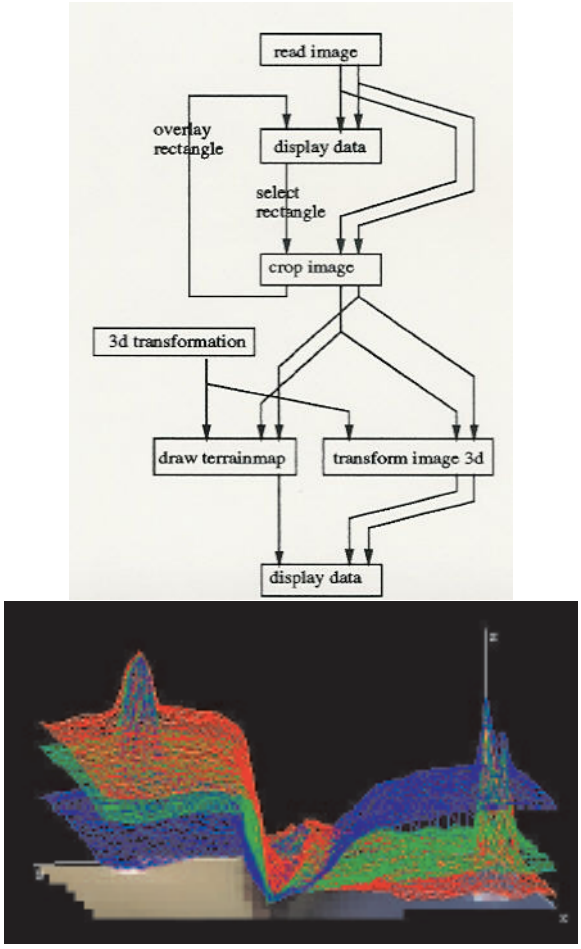

Figure 10: Network to overlay a surface graph on a slanted zoomed image of a region of interest, as well as the corresponding picture for the selected region of the plastic scene

data exploration arrangement by a stroke of the mouse. The right picture in Figure 9 shows the resulting display of a small region of the plastic scene, merging the data shown in Figures 5 and 8 into as single picture.

In addition to two-dimensional overlays on image data, users can also configure three-dimensional viewing arrangments. Figure 10 shows how the zoomed image can be viewed in combination with the surface plots of Figure 4. In this case, the network contains a module to specify the three-dimensional viewing direction, one to draw draw the surface plot (*draw terrainmap*) and one to map the image data into the (x, y)-plane of the three-dimensional coordinate system.

4.2 Cursor linking

Beyond data probing tools within a single image, users also need mechanisms that help them understand relationships between different parts of their data set, such as the relationship between MR and CT data. Furthermore, users may also be interested to visualize the same data several times in different presentation styles side-by-side, for example as an image and as a terrain map. If visualization involves data display in several, separate windows, users need mechanisms which link the information from all windows together. In EDI, a system of linked cursors can be configured and overlaid on the windows to indicate related data. The cursor-linking concept allows users to experiment with linking arrangments that can accomodate the needs of their application.

In a typical cursor-linking arrangement in EDI, different parts of a data set are shown in several windows side-by-side. Each window has its own *software cursor* to indicate the current position of interest in the window. The cursor in any window can be repositioned with the mouse. Cursor movement in one window affects the cursors in all windows. Such *live links* between several windows can be established interactively in the AVS network editor. Each displayed window corresponds to a different *display data* module in the network. When one of them receives a new, interactive mouse positioning event, the event is cross-fed to all display modules. The *logbook* mechanism ensures correct cursor positioning even if different windows show the data at different scale factors.

Figure 11 shows two registered pre- and post-contrast MR images of a human head before and after a contrast solution had been injected. A third image shows the pixel-wise differences. The bright pixels indicate areas which

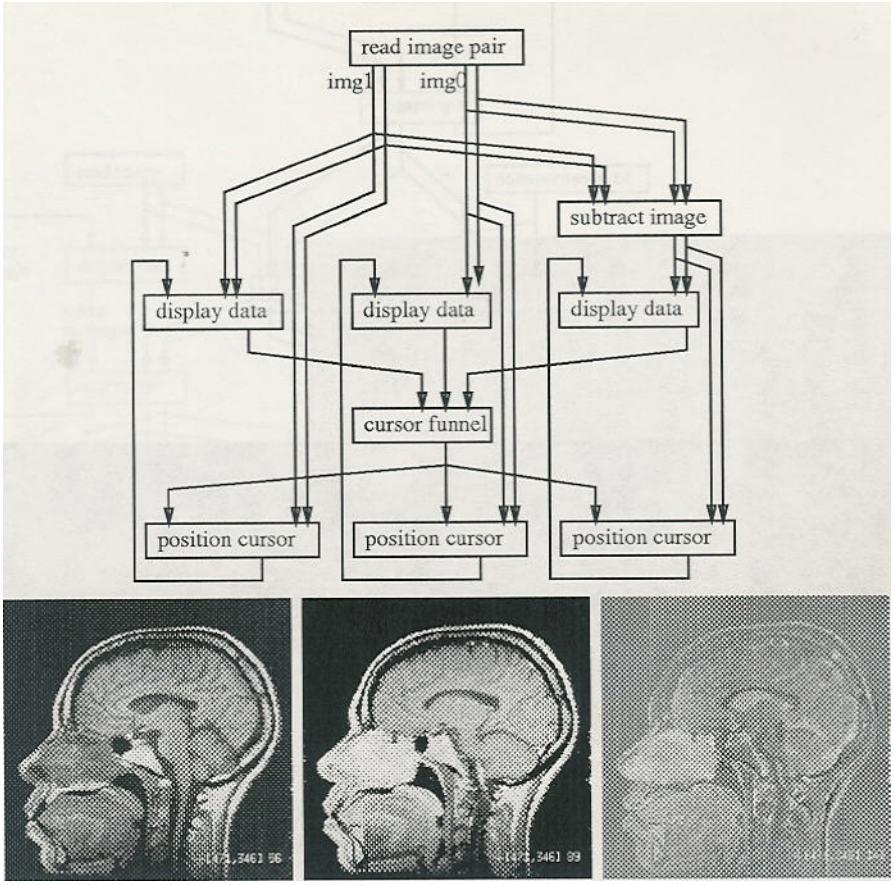

Figure 11: Network to view pre- and post-contrast MR-images and their difference image with linked cursors

Figure 12: Three levels of an image pyramid with linked cursors

the solution has penetrated. On the screen, the images are shown in three windows with linked cursors: When a user identifies an interesting position in one of the windows, the corresponding pixel positions are marked in all three windows. The network in Figure 11 shows that this functionality is achieved by multiplexing the cursor positioning output from all three *display data* modules in the network through a *cursor funnel* and then sending it on to three *position cursor* modules. Each *position cursor* module creates geometric drawing commands which are then sent back to the respective *display data* modules.

Figure 12 shows how the same cursor-linking mechanism is applied to three levels of an *image pyramid*. The lowest level of the pyramid shows the image at full resolution, higher levels of the pyramid show the data at increasingly lower sampling rates.[3] It is essential to the use of image pyramids that pixel positions can be visualized across all levels, using the appropriate coordinate transformations.

The cursor-linking mechanism in EDI can also be used to establish relationships between windows with very different types of dimensions, such as to

[3]Such image pyramids are frequently used in computer vision. Low-resolution images help algorithms to focus on global image features while the images at higher resolution provide the detailed features.

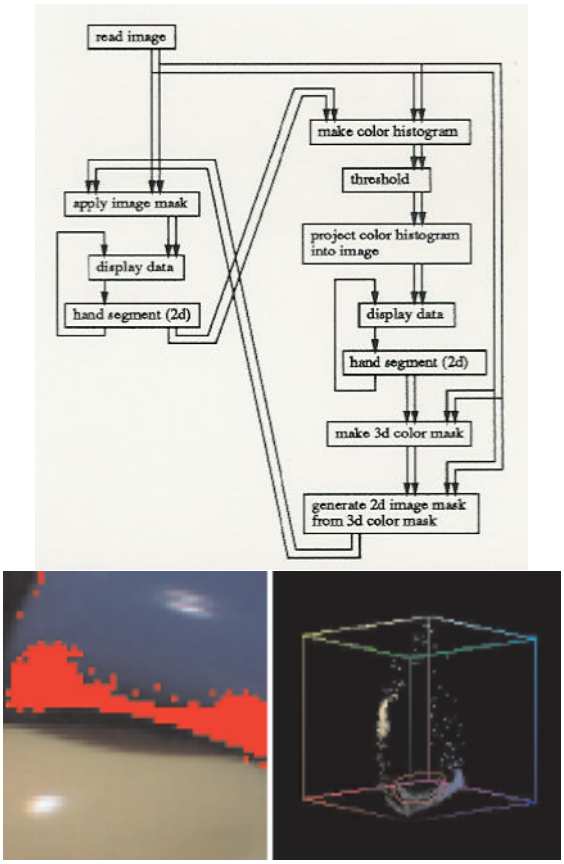

Figure 13: Network to view a color image and its histogram with linked cursors

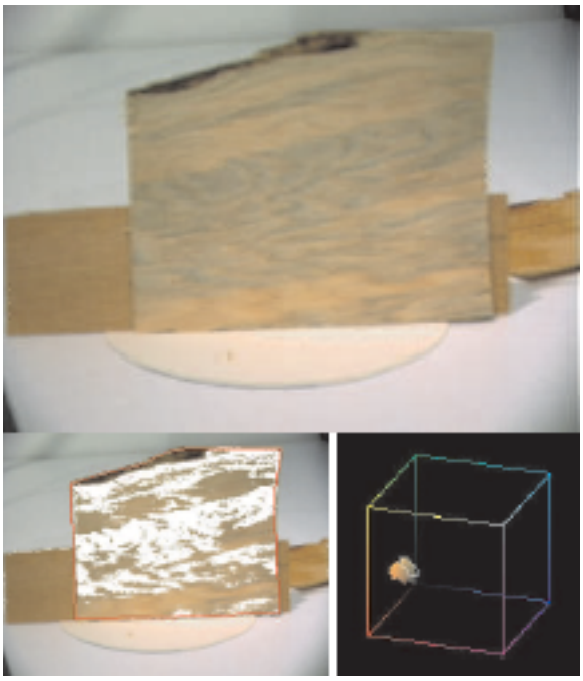

Figure 14: Color histogram analysis for wood inspection research

link an image to its histogram. In this case, a selected mouse position in the image does not translate directly into a position in the histogram. Instead, the data value at the pixel position is used as the index into the histogram. This is very easy in EDI: the mouse positioning information from the image window just has to flow through a translation module which performs the necessary pixel lookup.

Figure 13 shows a network which relates a color image to a color histogram (a three-dimensional scatter plot). A color histogram consists of four dimensions, $(r, g, b, count)$. It is stored as a volume, with each voxel (r, g, b) indicating how many pixels in the original color image have this particular color value. EDI presents the color histogram, using a z-buffering algorithm and ignoring all voxels with *counts* below a threshold. The voxels are displayed either as intensity data according to their *count*-value, or as color triples (r, g, b) according to their position in the histogram. In Figure 13, the color image of the blue and yellow rings from Figure 8 and its color-encoded histogram are shown side-by-side. The left branch of the network shows the modules responsible for processing and displaying the color image. The modules on the right deal with the histogram. Mouse interaction with the original image creates a local, hand-segmented image mask which is then passed on to the color histogramming column of the network where a new histogram is created, using only the image pixels under mask. Mouse interaction on the color histogram, on the other hand, creates a hand-segmented mask in the color histogram which is translated into a color image mask, indicating all pixels in the color image with colors in the selected part of the histogram. In Figure 13, the small area which is outlined in red has been selected in the color histogram. The resulting red image mask is overlaid on the color image, indicating the interreflection area.

Figure 14 shows how the same network can be used to design automatic wood inspection algorithms [6]. For such tasks, various impurities, such as knots and stains, need to be detected and classified. They typically cause color shifts in the wood, yet they are hard to recognize automatically, due to the presence of specularities and other phenomena. Figure 14 shows a wood sample with a blueish wood stain. The color histogram in the lower right picture has been oriented interactively such that blueish colors can be hand-selected. The lower left figure shows the corresponding pixels in the image, identifying a significant portion of the stained area. This interactive viewing arrangement in EDI can help the development of color-based automatic wood-inspection algorithms, since researchers can investigate the

characteristic color clustering properties for different wood samples.

The semantics of any relationship between data sets can be encoded in transformation modules which can then be used to visualize the data sets together. Extensions to higher-dimensional statistical data sets, such as data bases of census data or financial data, can be created.

Live links for visualizing the relationship between images and their histograms are built into some current visualization systems [18, 31]. Yet, those are closed systems without a visual programming interface. They provide only limited capabilities which cannot be extended easily by the user. In EDI, on the other hand, users can configure any cursor-linking arrangement they need.

4.3 Flexible exploration arrangements

The previous sections have presented and discussed many options for presenting and exploring data. Many more presentation styles and mixes of capabilities can be generated, and system developers often face the question from the user: "Why did you chose this style? Are you convinced that another style would not have suited the problem better?" EDI adheres to a concept of medium-scale modularity such that flexibility in providing and rearranging data presentation and exploration styles can be achieved. The modules can then be assembled into networks of modules which AVS can package into macro modules.

Figure 15 demonstrates the importance and power of EDI's medium-scale modularity. It shows a network which operates on two sequential sections of an embryo heart. The slices need to be registered with respect to one another. Carlbom et al. [8] have developed a stand-alone, interactive blink comparator with which users can interactively move and rotate one image while the images are rapidly alternated (*blinked*) in the window, helping the user can visualize the relative position of features in both images and minimize their dislocation. A similar, yet much more adaptible, capability can be provided in EDI, as shown in Figures 6. Figure 15 demonstrates how the functionality can easily be extended in EDI, e.g. by adding a *color-merging* module to the network which folds up to three grayscale images into a color image. The bottom left picture shows two intentionally misaligned slices of an embryo heart which were viewed under a light microscope. The first image is folded into the red and green color bands; the second image becomes the blue color band. The misalignment between the two slices thus

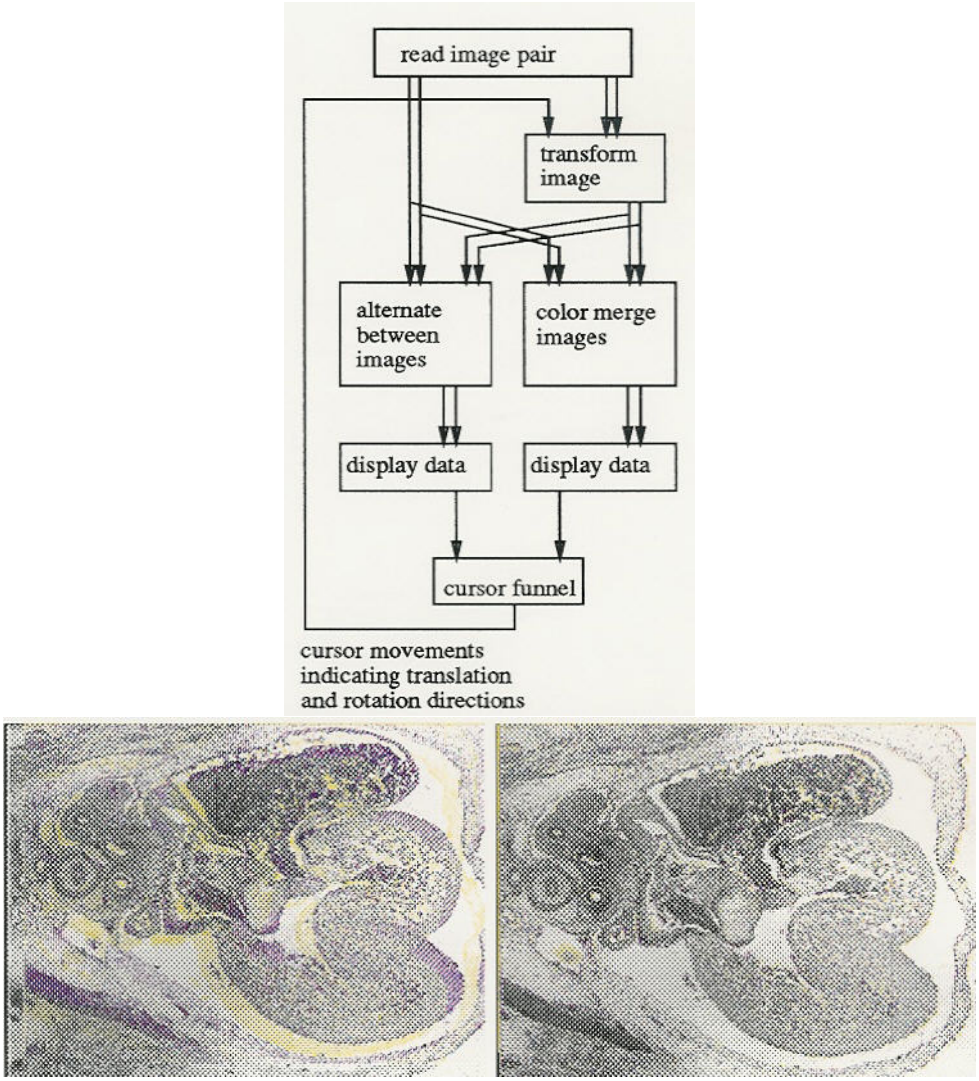

Figure 15: Network which combines a blink comparator with a color-merging module, used to align consecutive slices of an embryo heart

results in yellow or blue areas along image contours, whereas aligned areas have similar red, green and blue components and thus look gray. The bottom right picture shows the same slices after alignment.

Similarly, the viewing arrangement of the pre- and post contrast MR-images of a human head in Figure 11 can be changed with a few mouse strokes. In Figure 11, the images were shown side-by-side with linked cursors. In Figure 16, the pre- and post-contrast images are merged into a color image, as shown in the lower left picture. The pre-contrast image is represented by yellow colors, the post-contrast image by blue colors. The small region indicated by a white rectangle is displayed in the lower right picture as the base plane of a three-dimensional cube, with the difference image between the pre- and post contrast images overlaid as a red surface map. In Figure 17, the two images are overlaid as yellow and blue surface maps on an intensity-based display of the difference image.

5 Tele-collaboration

Many data exploration tasks in real applications are group efforts rather than problems solved by an individual. For example, surgeons seek the advice of radiologists when they use CT and MR data before and during an operation. Currently, they need to schedule a meeting or the radiologist has to rush from his office to the surgery room, where he has to scrub before being able to consult. Because of such inconveniences, consultations will only occur when absolutely necessary or may be impossible in emergencies, if the radiologist is too far away. Moreover, advice from experts across the continent or ocean cannot be sought spontaneously. Similar tele-collaboration needs exist in many other application areas, such as sharing test results of pharmaceuticals between a research lab and the manufacturer, or using new seismic measurements in the discussion between higher management of an oil company and drilling sites when determining the position of new bore holes.

Quite a few tele-communication systems are currently being developed to address some of these needs. So far, they focus on the problems involved in high-bandwidth data transfer over extended and slow networks. For example, *teleradiology* systems typically provide capabilities such that radiologists can access images remotely, e.g, from their personal computers at home when they are *on call*. The radiologist typically first transfers the data to the local computer and then displays or processes it there, giving a diagnosis

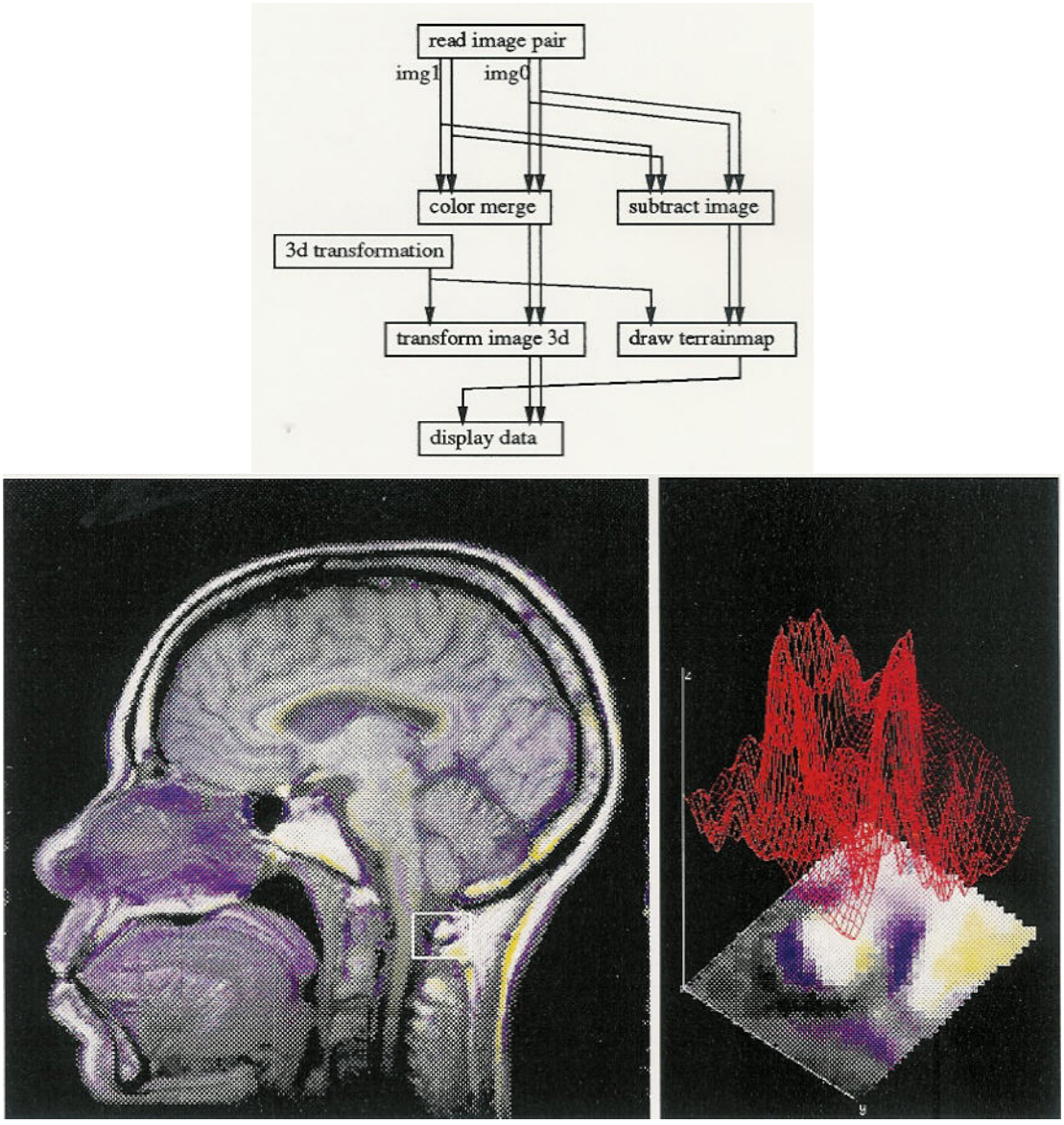

Figure 16: Network which shows a color-merged rendition of two MR images and overlays their difference image as a surface map

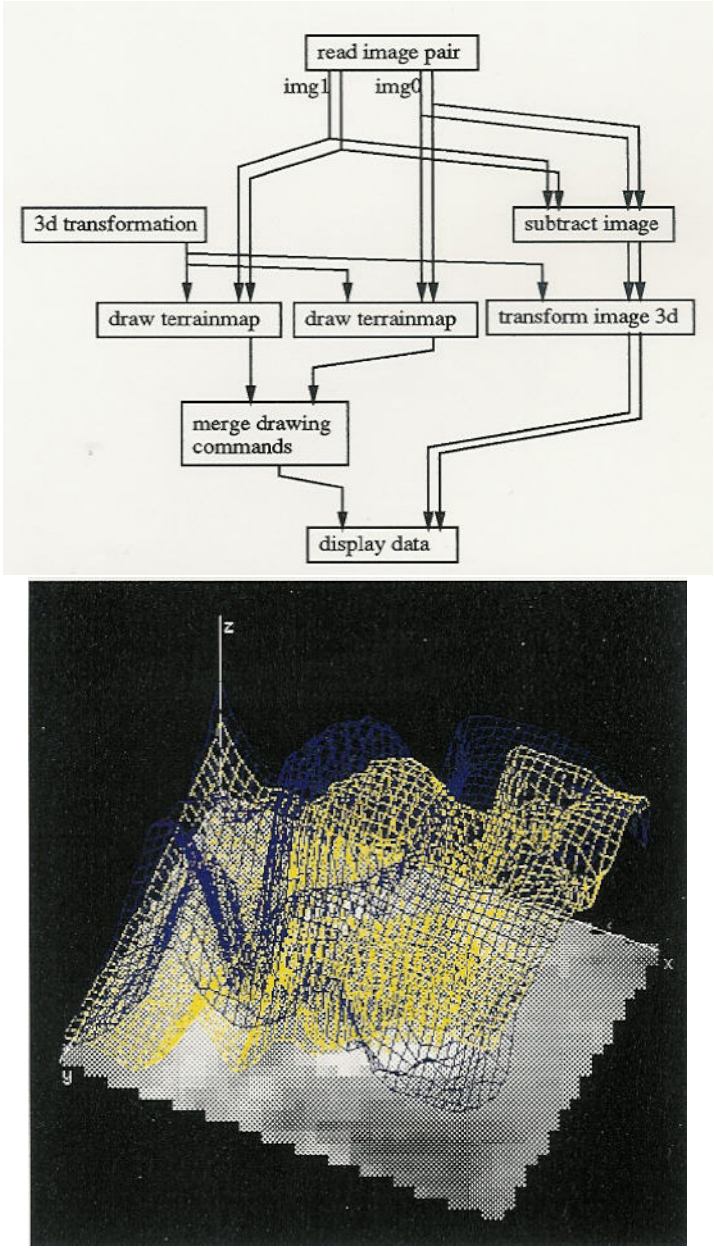

Figure 17: Network which shows two MR images as surface maps in different colors and their differences in an intensity image

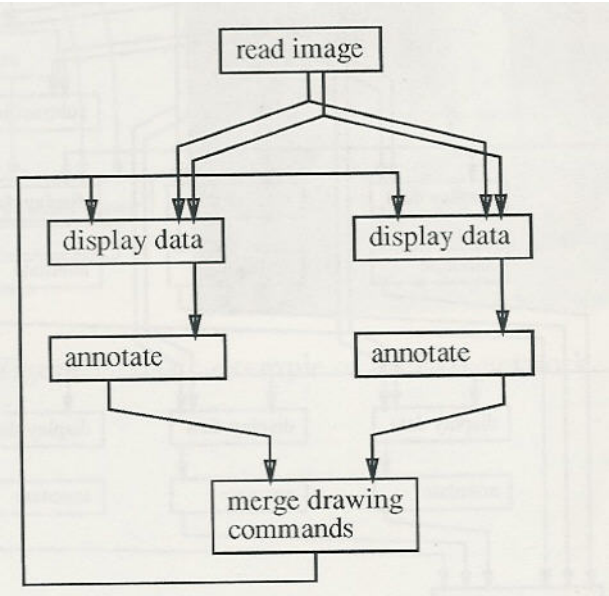

Figure 18: Basic tele-collaboration network

over the telephone. In some cases, colleagues at both ends can actually view and annotate the data simultaneously across the network, using tools such as shX[1]. Yet, such systems do not yet allow colleagues to adjust their viewing arrangement independently, according to differing display hardware capabilities or different viewing preferences. Data visualization and exploration concepts need to be included in the design for systems to develop from *tele-communication* arrangements towards true *tele-collaboration* systems.

EDI opens the way for collaborative data exploration, due to its open *display data* module and its window migration capability. Linked cursors can be established across long-distance computer networks just as easily as between neighboring windows on a single screen. The *display data* module allows users to view their data on any display which is reachable over the network, simply by specifying a *display name* parameter according to the naming conventions in X. With this mechanism, users can let windows migrate between arbitrary displays on the network. We have successfully demonstrated EDI's tele-collaboration capability between Chicago and Boston as part of the Innovation Showcase at SIGGRAPH 92, as well as between Boston and Sweden.

Figure 18 presents the basic network necessary for tele-communication.

An image is read in and displayed in two separate windows, potentially on different displays across the country. The mouse events from either window are sent to annotating modules, which translate mouse movements into sequences of differently colored line drawing commands. The drawing commands from both windows are then merged and sent back to both display modules so that the lines are overlaid on the images in both windows. The result is a system in which annotations made in either window are shown in both windows.

This arrangement can be used in the medical scenario discussed above. The radiologist and the surgeon can jointly annotate the image, while viewing it as suitable, e.g, one might see it enlarged, covering the entire screen, while the other has placed it into a small area of the screen such that it doesn't overlay with other important windows. Furthermore, since EDI's display module adapts to the frame buffer quality at the different sites, the same image can be viewed at different quality levels: on a highly specialized, high resolustion device in the operating room and at much lower resolution on the home computer of the radiologist on call.

Figure 19 shows a more complicated tele-communiation network. In this case, the pre- and post-contrast MR images and their difference image from Figure 11 are displayed and explored at two different sites, using linked cursors between six different windows. This figure emphasizes that EDI combines all of its data presentation and exploration capabilities with a very flexible data communication scheme. Whereas other tele-communication systems typically provide an identical arrangement of shared windows, users of EDI are free to assemble any window arrangement they need and share all or parts of it with as many colleagues as necessary.

6 Discussion

Applications which use empirical data impose special requirements on visualization environments. They need a very high degree of interactivity which has been ignored by current visualization systems. EDI is designed to aid with the analysis of empirical data. Built on top of AVS, it benefits from the visual programming paradigm that AVS provides. By making feedback from the user directly available to all modules and by introducing a *logbook* scheme between modules, EDI is able to evolve AVS from a pure visualization system into a data exploration system. In this new system, users are able to inter-

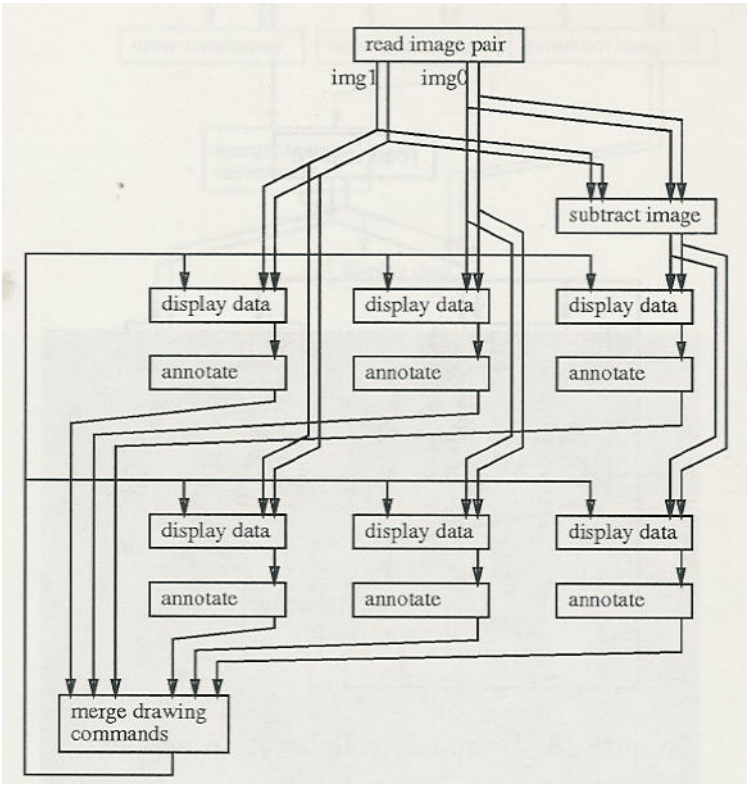

Figure 19: More complicated tele-collaboration network

actively select pixels in windows, perform arbitrary operations on them, and redisplay them in many different ways. Selected pixels can be cross-linked between several windows with different data sets and sent around the world for tele-collaboration arrangements between remote colleagues. Furthermore, EDI provides many different techniques for displaying data, and for mixing and merging data from several data sets into a single view.

These capabilities are essential to the analysis of empirical data. We thus expect EDI to open up scientific visualization environments to new classes of applications. But the question is: Will application programmers now stop writing their own visualization code? Probably not. Quite a few limitations still exist which need to be addressed. Among them are needs to connect visual programming environments with a static data depository, such as a data base or a knowledge base. Data exploration is an iterative process in which the user and the computer together incrementally extract the information from the data set. Data cannot just flow dynamically; it needs to be retained during the exploration session so that partial results of various avenues of investigation remain available for later comparison. Another limitation concerns real-time processing of time-critical events. If a module creates sound, generates a movieloop, or receives video sequences from a TV-camera, users need tools to ensure that the modules will be scheduled with guaranteed real-time response. Current data-flow-oriented schedulers for visual programming environments need to be redesigned to provide the appropriate mechanisms to users – potentially providing control flow capabilities as well as data flow capabilities. Furthermore, more elaborate graphical user interfaces will be necessary to provide full-scale tele-collaboration capabilities. Travelling data windows need to be accompanied by parameter windows which contain a suitable subset of the parameters from all modules which are involved in the creation and modification of the data that is shown in the travelling window. Most likely, all communicating parties also want access to the network editor so that everybody can customize their own viewing arrangement. We plan to address such issues in the future.

EDI has already proven itself useful for data exploration. It has been used to analyze color variation on wood samples, and to investigate reflection properties on plastic objects for computer vision research. It is an integral part of ongoing biomedical research at our research lab, and we have demonstrated its tele-collaboration capabilities on several occasions. Due to its visual programming interface and its interactive feedback channel, EDI has great potential for data exploration in many other areas. We plan to use

it in several more applications, such as earth sciences and statistical studies, in the future.

Acknowledgements

The development of EDI has been influenced by many people. I am especially grateful to the members of the Visualization Group at CRL, Ingrid Carlbom, William Hsu, Richard Szeliski, Demetri Terzopoulos, and Keith Waters, for their help and for spending many hours discussing aspects of EDI and other visualization environments with me. Several people have helped me understand the data exploration needs in various application areas, such as wafer board inspection (Yossi Glass), biomedical imaging (Ingrid Carlbom and Ika Rogowska), wood inspection (Alberto Maristany and Jim Funck), and earth sciences (Peter Kochevar and Ingrid Carlbom). Michal Altenhofen has shown me the current state of the art in sharing X between several displays. Ika Rogowska and Keith Batchelder have introduced me to the current usage of teleradiology. Steve Franklin has spent time discussing characteristics of AVS with me. He has also been a source of encouragement on several occasions.

The data shown in this paper has been collected from various sources. The plastic scene comes from the Calibrated Imaging Lab of the Robotics Institute at Carnegie Mellon University. Richard Szeliski has provided the tea tin image. The CT scan of a human chest and the pre- and post-contrast MRI scans of a head are from the Mallinckrodt Institute. The embryo heart images belong to the Visible Embryo Project at the University of Illinois. The CT data volume of the human head has been generated at North Carolina Memorial Hospital. The particular view shown in this paper was rendered by William Hsu. The wood sample is from Department of Forest Products at Oregon State University.

References

[1] Michael Altenhofen, Burkhard Neidecker-Lutz, and Paul Tallett. Upgrading a window system for tutoring functions. In *European X Window System Conference and Exhibition (EX'90)*, November 1990.

[2] D.F. Andrews. Plots of high-dimensional data. *Biometrics*, 28:125–136, March 1972.

[3] R.A. Becker and W.S. Cleveland. Take a broader view of scientific visualization. *PIXEL*, 2(2):42–44, 1991.

[4] R.A. Becker and W.S. Cleveland. Viewing multivariate scattered data. *PIXEL*, 2(2):36–41, 1991.

[5] S. Bly. *Sound and Computer Information Presentation*. PhD thesis, Computing Science Group, University of California, Davis, Lawrence Livermore National Laboratory, March 1982.

[6] C.C. Brunner, G.B. Shaw, D.A. Butler, and J.W. Funck. Using color in machine vision systems for wood processing. *Wood and Fiber Science*, 22(4), 1990.

[7] I. Carlbom, W.M Hsu, G. Klinker, R. Szeliski, K. Waters, M. Doyle, J. Gettys, K.M. Harris, T.M. Levergood, R. Palmer, L. Palmer, M. Picart, D. Terzopoulos, D. Tonnesen, M. Vannier, and G. Wallace. Modeling and analysis of empirical data in collaborative environments. *Communications of the ACM (CACM)*, 35(6):74–84, June 1992.

[8] I. Carlbom, D. Terzopoulos, and K.M. Harris. Reconstructing and visualizing models of neuronal dendrites. In *Proc. of CG International '91: Visualization of Physical Phenomena, Boston, MA*, Tokyo, Japan, June 24-28 1991. Springer-Verlag.

[9] H. Chernoff. The use of faces to represent points in k-dimensional space graphically. *Journal of the American Statistical Association*, 68(342):361, 1973.

[10] G. Grinstein, R.M. Pickett, and M.G.Williams. EXVIS: An explanatory visualization environment. In *Graphics Interface*, pages 254–261, London, Ontario, Canada, June 1989.

[11] A.J. Hanson and L.H. Quam. Overview of the SRI Cartographic Modeling Environment. In L.S. Bauman, editor, *DARPA-Image Understanding (IUS) workshop*, pages 576–582. Morgan Kaufmann, April 1988.

[12] W. Hibbard, C.R. Dyer, and B. Paul. Display of scientific data structures for algorithm visualization. In *Proc. of Visualization '92*, pages 139–146, Boston, MA, October 1992. IEEE Computer Society Press.

[13] W.M Hsu. A massively parallel volume renderer. Private communication, 1992.

[14] B. Kaplan. Sonification in AVS. In *AVS '93*, Walt Disney World, Lake Buena Vista, FL, May 24-26 1993.

[15] G.J. Klinker. *A Physical Approach to Color Image Understanding*. PhD thesis, Computer Science Department, Carnegie-Mellon University, May 1988. Available as technical report CMU-CS-88-161.

[16] G.J. Klinker. We need interactive data interpretation rather than interactive data visualization. Workshop on Scientific Visualization, Visualization '91, San Diego, October, 1991.

[17] G.J. Klinker. VDI – a Visual Debugging Interface for image interpretation and other applications. In F.H. Post and A.J.S. Hin, editors, *Advances in Scientific Visualization*, pages 165–195. Springer Verlag, Berlin, Heidelberg, New York, 1992. Presented at the Second Eurographics Workshop on Visualization in Scientific Computing, Delft, Netherlands, 22-24 April, 1991. Also available as technical report CRL 91/2 from the Cambridge Research Lab, Cambridge, MA 02139.

[18] R.D. Kriz. PV-Wave point and click. *PIXEL*, 2(2):28–30, 1991.

[19] B. Lucas, G.D. Abram, D.A. Epstein, D.L Gresh, and K.P. McAuliffe. An architecture for a scientific visualization system. In *Proc. of Visualization '92*, pages 107–114, Boston, MA, October 1992. IEEE Computer Society Press.

[20] C.C. McConnell and D.T. Lawton. IU software environments. In L.S. Bauman, editor, *DARPA-Image Understanding (IUS) workshop*, pages 666–677. Morgan Kaufmann, April 1988.

[21] P.J. Mercurio. The data visualizer. *PIXEL*, 2(2):31–35, 1991.

[22] P.J. Mercurio. Khoros. *PIXEL*, 3(1):28–33, 1992.

[23] J. Morse. Using a audio module within the EDI framework. Private communication, 1992.

[24] J. Mundy, T. Binford, T. Boult, A. Hanson, R. Beveridge, R. Haralick, V. Ramesh, C. Kohl, D. Lawton, D. Morgan, K. Price, and T. Strat. The image understanding environment program. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'92)*, pages 406–416, 1992.

[25] Department of Computer and Electrical Engineering of the University of New Mexico. Xvision 3.0. Also known as the Visualization Workbench from Paragon Imaging, 1989.

[26] L. Quam. The Image Calc vision system. Technical report, Stanford Research Institute, Menlo Park, CA, 1984.

[27] K. Riley and C. McConnell. Powervision. Technical report, Advanced Decision Systems, Mountain View, CA, March 1988.

[28] R.W. Scheifler and J. Gettys. The X window system. *ACM Trans. Graphics*, 5(2):79–109, April 1986.

[29] S. Smith and M.G. Williams. The use of sound in an exploratory visualization environment. Technical Report R-89-002, Department of Computer Science, University of Lowell, Lowell, MA 01854, May 1989.

[30] C. Upson, T. Faulhaber Jr., D. Kamins, D. Laidlaw, D. Schlegel, J. Vroom, R. Gurwitz, and A. van Dam. The Application Visualization System: A computational environment for scientific visualization. *IEEE Computer Graphics and Applications*, 9(4):30–42, July 1989.

[31] T.D. Williams. Image understanding tools. In *IEEE 10th International Conference on Pattern Recognition (ICPR'90)*, pages 606–610, Atlantic City, NJ, June 1990. IEEE.