Chapter 13

Manipulation, Display, and Analysis of Three-Dimensional Biological Images

HANS CHEN, JOHN W. SEDAT AND DAVID A. AGARD

Department of Biochemistry and The Howard Hughes Medical Institute
University of California at San Francisco, San Francisco, Ca. 94143-0448

INTRODUCTION

Due to dramatic advances in optical microscope technology, it is now possible to examine biological specimens in three-dimensions with either electron microscopy or light microscopy. Although the algorithms used for generating the three-dimensional reconstructions are well known, much work remains on developing methods for the display and analysis of the resultant complicated three-dimensional data. After brief discussion of a rational file format designed for storing three-dimensional image data and its properties (e.g. pixel spacing, orientation, size etc), we will go on to discuss two major aspects of three-dimensional image data handling. The first covers image processing schemes for enhancing features in the data and some computational methods for manipulating three-dimensional data sets. In the second part, we will discuss the image display system in the context of software design and hardware requirements, which must be considered for convenient data visualization and measurement. Much effort has been spent on developing a generalized display system that can also be used for model building and analysis. By model building we mean the interactive tracking of features in three-dimensional volumetric images. Throughout this chapter we will make a distinction between display methods that directly utilize three-dimensional image data stored as a contiguous set of pixels (also called volumetric data, or voxel data) and those that convert the data into a set of polygon vertices that define a single-level contour surface. Although this latter approach contains much less information than volumetric approaches, it can make stunning pictures. Figure 1 shows a typical procedure for handling three-dimensional data involving image enhancement, data manipulation, data display and analysis.

STORAGE OF THREE-DIMENSIONAL IMAGE DATA

Three dimensional data can be obtained from a variety of microscopic sources. From light microscopy, three-dimensional data can be obtained as a series of two-dimensional images taken at different focus positions from either confocal microscopes or from conventional optical microscopes equipped with digitizing video cameras or cooled, scientific grade CCD detectors. Data from electron microscopy can be in the form of digitized images of physical thin sections or the result of tomographic analysis of thick sections. In all of these cases a three dimensional data set can contain an enormous amount of information. In our own work, we often have files of over 100 Mbytes in size for a single data set. There is thus a pressing need for a convenient way of managing this data. Several years ago at the MRC labs in Cambridge, England a unified file format for storing and manipulating two- and three-dimensional pixel data was developed for handling x-ray density maps and images. It is absolutely essential that there be a single header (Table 1) that describes the geometric and physical contents of the file that is a part of the same file as the data. Furthermore, it is imperative that each set of three-dimensional data be stored in a single file. Random access capability then allows the rapid access to any single image section or part of a section. In this way, all of the sections of a three-dimensional image can be managed as a single unit. The file design that was chosen consisted of a 1024 byte header immediately followed by the image data. A mode flag in the header indicated the format of the data. Each pixel could be represented numerically either by a single byte (allowing numbers from 0–255 to be stored), by a two-byte integer (values from −32768 to 32767), a four-byte real number (±10^+3)), or by pairs of either integer*2 or real data to represent the complex values from a 2D or 3D Fast Fourier Transform (FFT). In addition to containing information on the size of the image, the pixel spacings, mode, three-dimensional origin and orientation of the image, symmetry operators and unit cell parameters for the x-ray maps, the header also contains space for 10 text labels each 80 characters long. This is very convenient as each program that manipulates the

![Diagram](image-url)

**FIG. 1.** Procedure for handling 3D image data.
Manipulation, display, and analysis of three-dimensional biological images

TABLE 1

<table>
<thead>
<tr>
<th>MRC</th>
<th>Image File</th>
<th>data structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-3</td>
<td>NXYZ</td>
<td># of columns, rows, sections</td>
</tr>
<tr>
<td>4</td>
<td>MODE</td>
<td>data type: [*1,<em>2, R</em>4, Complex]</td>
</tr>
<tr>
<td>5-7</td>
<td>NXYZSTART</td>
<td>number of first Column,Row,Section in map</td>
</tr>
<tr>
<td>8-10</td>
<td>MXYZ</td>
<td>number of intervals along X,Y,Z</td>
</tr>
<tr>
<td>11-13</td>
<td>XYZLENGTH</td>
<td>cell dimensions (cell/mxyz = pixel spacing)</td>
</tr>
<tr>
<td>14-16</td>
<td>ALP,BET,GAM</td>
<td>cell angle (Degrees)</td>
</tr>
<tr>
<td>17-19</td>
<td>MAPCRS</td>
<td>which axis corresponds to Columns,Rows,Sections</td>
</tr>
<tr>
<td>20</td>
<td>AMIN</td>
<td>minimum density value</td>
</tr>
<tr>
<td>21</td>
<td>AMAX</td>
<td>maximum density value</td>
</tr>
<tr>
<td>22</td>
<td>AMEAN</td>
<td>mean density value</td>
</tr>
<tr>
<td>23</td>
<td>ISPG</td>
<td>space group number</td>
</tr>
<tr>
<td>24</td>
<td>NSYMBT</td>
<td>number of bytes used for storing symmetry operators</td>
</tr>
<tr>
<td>25-40</td>
<td>EXTRA</td>
<td>user defined storage space</td>
</tr>
<tr>
<td>41</td>
<td>IDTYPE</td>
<td>type of data set, section, projection, etc</td>
</tr>
<tr>
<td>42</td>
<td>DV</td>
<td>data value 1, info related to data type</td>
</tr>
<tr>
<td>43</td>
<td>DV2</td>
<td>data value 2, more info for data type</td>
</tr>
<tr>
<td>44-49</td>
<td>Tilts</td>
<td>2 tilt sets (original and Tilts 2 tilt sets (original and current)</td>
</tr>
<tr>
<td>50-52</td>
<td>Waves</td>
<td>multiple wave length information</td>
</tr>
<tr>
<td>53-55</td>
<td>XYZORIGIN</td>
<td>x,y,z origin of image</td>
</tr>
<tr>
<td>56</td>
<td>NLABL</td>
<td>number of labels being used</td>
</tr>
<tr>
<td>57-256</td>
<td>LABEL(20,10)</td>
<td>ten 80 character text labels</td>
</tr>
</tbody>
</table>

file can add a label record allowing the processing history of the file to be clearly kept. A comprehensive set of subroutines has been developed to allow convenient manipulation of the image header as well as the image data itself. On disk, the data is stored in fixed length records to improve access and IO performance, however, all of this is completely transparent to the user who only needs to be concerned about images.

The convention chosen to specify the orientation of any image file is defined by the following equation.

\[
[XYZ] = [M]^{-1} [I_{XYZ} \times \Delta XYZ + OXYZ]
\]

Where matrix M is the same rotation matrix used for rotating the original data file and creating this new file. In simple section image files, OXYZ is the coordinate of the first data point in this file with respect to the rotated coordinate system, \(\Delta OXYZ\) is the pixel spacing. For a projection image file (see below), OXYZ is the coordinates of the center of the entire projecting volume. By making this information available to the image display system, it will be able to relate data from different types of image files.

**IMAGE ENHANCEMENT**

Once a three-dimensional data set has been obtained, it is often necessary to correct for systematic errors in the data. Such errors can be due to the lower resolution along the focus direction that occurs with either confocal data or conventional optical section data, or the missing data that occurs in electron microscopic tomography due to practical limitations on tilt angle. Further, confocal microscopy, especially laser scan methods, can suffer from scan irregularity and various geometric distortions occurring off the central axis. Three-dimensional data recorded on CCD cameras must be corrected for variations in gain and offset for each pixel (Hiraoka et al, 1987). All of these artifacts can and should be corrected before proceeding to visualization or quantitative analysis. We have previously described methods for image restoration that dramatically improve the three-dimensional resolution obtainable with conventional optical sectioning microscopy (Agard et al, 1989). These methods are just as applicable to confocal data.

After suitably corrected image data have been obtained, there are several image enhancement schemes that can be used to enhance the visual appearance of an image (e.g. edge structure). As we shall consider it here, image processing (as distinct from image restoration) does not try to add or improve the information content of the three-dimensional images, but only serves to improve the clarity of presentation. The field of **image processing** methods is well developed and the reader is directed towards such classic texts as Castelman (1979) or Pratt (1978). Very often data that represent the part of the structure which is of interest falls within a small portion of the overall dynamic range. When this type of image is displayed without enhancement, the data range of interest will be translated into only a few intensity levels in the display, suppressing desired detail in the resultant image. Contrast stretching and histogram normalization are two standard approaches. Here we will concentrate on other methods that we have found to be particularly useful for three-dimensional microscopic data. Most of these schemes work by remapping the image data according to a specific type of function. By choosing proper functions and parameters the desired enhancement can be accomplished. Because image quality is often a subjective criteria, it is generally necessary to try a few methods with a few sets of parameters.

**Linear Filters**

The use of various linear filters is a common and productive way to manipulate image data. Confocal microscope data is often quite noisy and it is useful to apply a smoothing filter. Enhancement can be achieved through the use of high-pass or bandpass filters. These can be applied to two- or three-dimensional data either as convolution operations in real space or as multiplication operations in Fourier space. In general, it is common to use filters that are circularly or spherically symmetric because there is little reason to distinguish particular directions within the image (although this can be simply done). We typically apply filters in Fourier space and prefer the use of gaussian filters to avoid the introduction of ripples and contrast inversions that can otherwise be generated. Low pass, smoothing filters are generated using a gaussian function \((\exp(-r^2/2\sigma^2))\) or for enhancement we often use a Fourier filter of the form \((1-\exp(-r^2/2\sigma^2))\). These can be combined, using different values of \(\sigma\) in each part, to give a bandpass filter.

**Median Filters**

These are a class of non-linear filters that have the property of smoothing data while maintaining edges. The general scheme is to choose an NxN or NxNxN box (N is odd) surrounding a particular pixel in either two- or three-dimensional space and to calculate the median intensity within the box. Recall, that the median intensity is that intensity value which is half way up the histogram. This can be quite distinct from the mean density within the box. Note that the median is rather insensitive to the absolute intensity of a point. The central value in
the box is then replaced by either the median value (Median filter) or by the actual value minus the median (Median enhancement). The position of the box is then incremented by one pixel and the process repeated. The median filter has the property that objects smaller than about one half of the box size tend to be removed. With the median enhancement methods, broad backgrounds are effectively removed, and small features remain. This is a very powerful method, but can be quite time consuming. Several authors have described fast algorithms for computing these filters (Hung, T.S., 1979; Narendra, P.M., 1981).

**Local Contrast Enhancement**

Surprisingly, the local contrast enhancement method can perform as well or better than the Median Enhancement method, above, and can generally be computed more rapidly. This scheme defines its mapping function based on the local mean and local contrast in the neighborhood of a particular pixel. As with the Median filter, the local neighborhood is defined in terms of an \( N \times N \) or \( N \times N \times N \) box (\( N \) is odd) surrounding a particular pixel. The local mean is defined as the average of all the data values within the box, while the local contrast is defined as the difference between the central data value and the local mean. Once the local mean and local contrast values are calculated, the contrast and mean are separately re-mapped according to a pair of lookup tables (LUTs) and then summed to give the final value which is placed in the central pixel. Again, the position of the box is incremented by one pixel, and the process is repeated. This approach enhances an image by reducing the overall dynamic range of the data and by boosting the local contrast in a potentially nonlinear fashion. This approach is especially useful for microscope data as it allows background to be suppressed without the artificiality and error caused by thresholding as in conventional contrast stretching. The use of separate LUTs to permit independent manipulation of the mean and contrast levels allows contrast in the important areas to be increased while maintaining a good overall sense of the image. The key to success with this method is the appropriate definition of the lookup tables. These can be designed to take into account the psychophysics of image perception. An example of this approach is shown in Figure 2. Clearly, significant detail is seen in image B (after 2D filtering of the final projection images) that is present, but not obvious, in A.

**Gradient Method**

This method is designed mainly for edge enhancement which is helpful in trying to determine connectedness in a complicated data structure or as a preparation for solid modeling (see below). This type of enhancement has been used extensively by the group at PIXAR (San Rafael, California) (Dreibin et al), and we find it to be extremely useful. The idea is to form a new three-dimensional image that is made by multiplying the value of every pixel by the two- or three-dimensional gradient in the neighborhood of that pixel. Although the gradients may be high in the background regions, the image values are small, thus noise seems to be effectively suppressed. Before calculating the gradient, it is often useful to remap the image by a sigmoidal function to further suppress background. The density mapping function used in this method is defined as below:

\[
X_{out} = X_{in} \times (K + (1 - K) \times \text{grad}(F(X_{in}))) \quad 0 < K < 1
\]

where grad is the modulus of the three dimensional gradient:

\[
\sqrt{g(x)^2 + g(y)^2 + g(z)^2}
\]

Where the function \( F \) can be either unity or an ArcTangent (a convenient way to generate a sigmoidal function). \( K \) controls

---

**Fig. 2.** (A) and (R) show the difference between before and after LCE. Data shown is Epon sections of metaphase-arrested Kc chromosomes.
the mix of image and image × gradient. Figure 3 shows both the original and an edge-enhanced image. A three-dimensional automatic data segmentation scheme is being developed at UCSF which uses both three dimensional gradient and data value information to optimally define structures.

PROCESSING METHODS FOR DISPLAYING 3D DATA

Presenting three-dimensional data in a form where the vast amount of information present can be readily appreciated and analyzed is a challenging problem. Ideally, one would like to be able to wander around within a three-dimensional hologram under computer control to allow zooming in on particular regions or zooming out to see the entire data. Although one day this will undoubtedly be possible (there has been much progress in the computer generation of holograms), the present goals are to try to approach this immediacy of interaction with the data using considerably more mundane approaches. In order to visualize three-dimensional data, it must be converted in some manner to one or more two-dimensional images for display on a raster computer graphics system.

After 3D microscope data has been collected, reconstructed, processed, etc. it is generally thought of as a set of two-dimensional images. Each of which represents a slice through the data cube representing the three-dimensional specimen. A very simple way of displaying this data is to allow the user to examine each section individually, and when he is done to go on to another section. This can even be done in rapid succession, creating a movie. Unfortunately, although such approaches are ideal for examining the details of a single slice, they do little to provide a sense of the three dimensionality of the data as a whole because nothing has been done to integrate information into an image that appears to be distributed throughout three-dimen- sional space.

Three methods which we have found to be especially useful in visualizing the 3D aspects of these data sets are 1) simple display of stereo image pairs, 2) serial displays of mono or stereo projections of sequential rotations of the entire three-dimensional data cube, being displayed rapidly enough to achieve a movie effect and 3), simultaneous viewing of the same point in the data from multiple directions using two or more two-di-

Stereo Images

Stereo pairs can be calculated from a three-dimensional data set in either of two ways. The most accurate is to rotate the full three-dimensional data set by anywhere from ±3° to ±10° and then to project each of these rotate three-dimensional data sets onto a two dimensional plane (see also ROTPROJ, below). Another approach which is less accurate, but is less expensive computationally is the stacking method. Instead of actually rotating the three-dimensional image data, the required projection images are approximated by summing a series of section images with an appropriate pixel shift between the adjacent sections. The left eye part of the stereo pair is made by sequentially shifting each image to the left as one progresses from the farthest to the nearest plane. The number of the pixels to be shifted is dependent upon the actual pixel size and the distance between sections. In principle the angle resulting from the shift operation should fall between 3.0° and 7° for each image. To see the stereo effect on the computer monitor, the images can be displayed side by side and the viewer can wear simple 3D glasses which will fuse left and right images by properly adjusting angle between two reflecting mirrors in the glasses (very nice viewers are available from nu 3D vu Co, Eugene, Or.). A more convenient (and more expensive) system electronically alternates the display of the left eye and right eye right images on the monitor. The user can either wear a pair of mechanical or electro optical shutters, or a liquid crystal screen can be placed in front of the monitor that switches between left and right circular polarization in synchrony with the monitor. To assure that each image only reaches the appropriate eye, the user wears a pair of left-right circularly polarized “sunglasses” (available from Stereographics, San Rafael, Ca., or Tektronix, Beaverton, Or.). For more discussion see (Agard, et al 1989).

3D Rotations

With either display approach, computational methods are needed for generating new three-dimensional data sets in a different format or orientation. Two software programs have been
developed at the Biochemistry Department at UCSF for computationally rotating three-dimensional data sets and either projecting entire data set onto an assigned imaginary image plane or reslicing the data cube along the rotated axis. ROT3D is a FORTRAN program for rotating three-dimensional images. The approach taken in this program is following:

- Determine the size of the new rotated image data set
- Calculate the inverse rotation matrix
- Loop through every point in the output image and determine the coordinate of data point within the original image by pre-multiplying the new coordinate by the inverse matrix
- Determine the data value by interpolating between adjacent points in the original data file.

Rotated Projections

The second program ROTPROJ is for calculating a series of projection images of any three-dimensional image data set calculated at different tilt angles. To speed up the process for calculating the rotated projection images, the rotation and projection operations are performed simultaneously; the result being a substantial saving in computer time (Agard et al 1989). Consider a two-dimensional slice of the three-dimensional image taken perpendicular to the desired rotation axis: for example the X-Z plane for tilts about the Y axis. Then the desired coordinate transformation is given by:

\[ x' = (x-x_c) \cos \theta - (z-z_c) \sin \theta + x_c \]
\[ z' = (x-x_c) \sin \theta + (z-z_c) \cos \theta + z_c \]

Where \( x_c, z_c \) are the centers of rotation and \( \theta \) is the rotation angle. Instead of rotating all the data and then projecting it down the \( z' \) axis, we loop through the desired \( x' \) range and the original \( z \) solving for \( x \) as we go.

\[ x = \frac{[(x'-x_c) + (z-z_c) \sin \theta]}{\cos \theta} + x_c \]

the value to be added into the projection vector at \( x' \) is then given by one-dimensional interpolation:

- loop on \( x' \) in the output projection vector
- loop on \( z \) in image
- calculate \( x \) in image that corresponds to the current \( x', z \)
- linearly interpolate in the input image to get value and add to value in projection vector at \( x' \)

By saving one loop, this approach can speed calculations by as much as 30 fold. For \( |\theta| > 45 \), because of the \( \cos \theta \) in the denominator, it is advisable to loop on \( x, x' \) and solve for \( z \). Rotation about other axes can likewise be efficiently accomplished.

Simultaneously with forming the rotated projections, one can either alter the contributions of density values arising from the back of the image or add in the property of opacity to the sample. Opacity is especially useful, as often simply projected images tend to have an transparent quality to them that makes them hard to interpret. As each point is summed into the projected image, it is first multiplied by a weight that is kept for each pixel in the projected image. The initial value for the weight is 1.0, as each new point is added in, the weight for each pixel is multiplied by \( \exp(-\alpha*val/\text{mean}) \). This acts to make the higher values more opaque. For speed, a lookup table containing exponential factors is precalculated. Varying \( \alpha \), varies how opaque the object seems.

Pixar Displays

It is often particularly useful to combine this ROTPROJ method with the PIXAR edge enhancement approach (Driebin et al, 1988). When opacity is added, the net effect is very similar to the surface rendering methods described below. Another variation also developed by PIXAR, is instead of combing the overall three-dimensional gradient information when multiplying by the image, is to separately rotate the \( x, y, z \) components of the image gradient data and to then take the dot product with respect to a light source direction. This final value is the summed with opacity to produce a very nice and extremely rapid surface rendering that is rather insensitive to choice of contour level.

Contour Surface Representation

In this approach, a contour surface is defined by a single threshold level in a three-dimensional data set. Displaying this surface is done by finding a set of three-dimensional triangles which approximate the surface. The face of each triangle is colored with a constant value that is derived from the dot product between the direction of an imaginary light source and the triangle normal. Alternatively, it can be shaded by smoothly interpolating between intensity values calculated for each triangle vertex. These intensity values are determined in a similar manner by taking the dot product between the light source and vertex normal. The triangle or vertex normals are chosen to point away from the structure. An example of this type of surface representation is shown in Figure 4. Since the structure geometry is assumed to be completely random, finding a scheme which will automatically search for all these triangles is difficult. One scheme developed at UCSF, defines the location of each triangle and its normal by considering all of the possible ways that a three-dimensional surface can be enclosed by a unit box. Depending on how the intersections occur, triangles can be defined easily to approximate the enclosed surface. By looping through all the unit boxes in the sampling space, we can cover the whole surface of the structure. The approach seems to work best on simple data such as would be expected from stained axons or for data that has been processed using the Gradient method, above.

GRAPHIC SYSTEM FOR 3D IMAGE DISPLAY AND ANALYSIS

Programs such as FRODO (T. A. Jones, 1982) which utilize advanced computer graphics techniques to display x-ray data, are widely used by scientists in the field of protein structure research. Unfortunately, the cage contour representation of the three-dimensional data adopted in such programs is not suitable for microscopy data. This is mainly due to the greater level of structure irregularity, higher noise and nonisotropic data resolution. At the Howard Hughes Structural Biology Unit at UCSF an interactive computer graphics display system PRISM has been designed and built to serve the purpose of three-dimensional microscopy data display and analysis. It displays data as a gray level image on an computer raster graphics CRT.
Details of PRISM's Design and Implementation

The main goal of PRISM is providing the capability of three-dimensional image data analysis, especially model building, through data visualization. The design began with building the foundation of a flexible and versatile display functionality and then adding data analysis functions such as path tracing, density analysis, distance measurements, and point marking. The following is a list of design principals of PRISM.

\begin{enumerate}
\item[i)] Being able to handle multiple image displays and allowing each display to be manipulated (e.g., zoom, pan, update etc) independently. Comparison between several images is probably the most common practice in data analysis. Especially when working with three-dimensional data sets, it is often necessary to simultaneously examine a region of the data from multiple angles of view in order to resolve and interpret complicated structures. Thus a windowing capability (allowing multiple, independent images to be simultaneously displayed and manipulated) is a definite must.
\item[ii)] The capability to rapidly and sequentially display a set of images. As discussed in the previous section, one of the best ways to perceive a 3D structure is by looking at a set of sequentially rotated projection images displayed in movie mode.
\item[iii)] Accommodate images up to 1024 by 1024 pixels. Most digital microscopy data is no larger than 1024 by 1024 in one section. The system should be able to display the whole data section when desired.
\item[iv)] Providing an on-screen cursor for interacting with data displayed on the screen. Since PRISM is the system for both data visualization and analysis, the ability to interact with the scene on the screen is very important. A mouse-driven cursor can give the user a convenient way to either select data on the display screen or input program commands.
\item[v)] Functionality for tracing and building a model of a complicated biological structure from 3D microscope data. This is also the ultimate goal for PRISM.
\item[vi)] Model display. It can be difficult to appreciate the spatial aspects of the structure by looking at gray level images. Being able to display a simplified model in stereo with the capability to rotate the display gives the user an easy way to visualize the important aspects of a complex structure.
\end{enumerate}

The Window System

The software solution for the independent manipulation of multiple images on the screen is known as windows. A window is a viewport in which an image can be displayed and manipulated. Every image on the screen has a window structure associated with it. Displaying an image requires that the image be copied from a storage area in the display memory (the image buffer) to a pre-defined location on the area of memory that can be displayed (the viewport buffer). To create a new display on the screen requires adding a new window to the window system. Since PRISM is designed for both data display and
analysis, the functionality of this window system not only de- 

fines the mapping relationship between the image buffer and 

viewport buffer, but it also creates a geometrical link between 

a two-dimensional screen coordinate system and a real space 

three-dimensional coordinate system. Therefore, it is possible 

to determine the location in real space for data displayed on 

the screen. This is essential for doing three-dimensional model 

building and data analysis. This link is accomplished by track-

ing the source data file of every image loaded to the window 
system, its location in the data file, the screen coordinate of each window, and the geometrical information stored in the  

image data file header. (As discussed in the previous section, in  
doing three-dimensional analysis it is often necessary to ex-

amine the data stack from various angles. Since most computer  
systems are not capable of doing on-line three-dimensional vol-

umetric rotations, these are often pre-computed and stored in  
some image data files. Each one represents some geometric transformation of the data. It is necessary that all of the relevant  

orientation information be stored in the file header, so that 

images from different files can unambiguously be related to one  

another.) Every window has its own set of attributes, which  

completely determines the on-screen display contents. The fol-

lowing is a list of window attributes defined in PRISM:

- Image source. This is the data file associated with this par-


ticular window. Different windows can link to the same file, 

but each window can have only one associated file. Every  

image file has a header attached to it, which contains the  

relevant information such as size, pixel spacing, data type  

(e.g., projection image vs section image), orientation and  

origin in three-space.

- Allocation of space in the image buffer for preloading images 

from the associated image file. As mentioned earlier, images  

need to be pre-loaded into the image buffer to allow rapid  

updates. Every window has an array which contains the lo-

cation of each image stored in the image buffer.

- Region of the image that has been loaded into the image bu-

uffer.

- Window size and its location in the viewport buffer.

- Zoom Factor

- Display priority. The window system maintains a priority stack to determine the display order for overlapping win-


dows. In the current system there are up to 8 windows. The  

window on top of the stack is called the active window. Only  

its content will be affected by the user input commands.  

Selecting the active window is done by raising the target  

window’s display priority to the top of the stack and pushing 

the priority of all the other windows one level down. In the  

case where none of the existing windows are set to be on top  

(done by setting top entry to be 0), all the windows become  

active. This means that the same command will apply to all  

the windows. For instance if the command is ZOOMUP  
every window’s zoom factor attribute will be incremented and 

the whole screen will be repainted with the new zoom factor.  

This gives users the ability to perform parallel operations among the windows. Using two windows to generate a  

to a stereo movie is one example of this. Images are manipu-

lated on the screen by changing the value in the correspond-


ing attribute and repainting the screen.

Since each window has its own set of attributes, images can 

be manipulated in a completely independent fashion. One as-

pect that should be emphasized here is zooming. The ability 

to have different zoom factors on a single screen gives great flex-

ibility in data visualization. The traditional zoom is performed  

display hardware and applies a single zoom factor through-

out the whole screen. Whenever the zoom factor changes the  

size of the viewport area also changes. Because of this, it is  
difficult to maintain image layout on the screen. Furthermore,  

menu items that are displayed on the screen will also be zoomed  

up and down. This can be very inconvenient. For these reasons  

we have adopted the use of software zooms instead of hardware  

zooms. This is most easily done by pixel replication while copy-

ing the image from the image buffer to the viewport buffer. The  

bigger the zoom factor, the smaller the area of data that will be  
mapped onto display window. Zooming is set up to take place  

with respect to the image center instead of a screen corner.

Digital Movies

The hardware must be capable of accomplishing both the  

desired movie mode and the windowing functionality. To  

achieve the movie effect, it is necessary that images to be up-

dated at the rate of 6 to 10 frames per second. Image display  
generally involves reading data in from disk, scaling between 0  

and 255, and loading the result into the viewport buffer of the  

graphic imaging device. To obtain the desired speeds, extremely  

high speed, parallel transfer disks, and a fast host CPU with a  

high speed data bus between system memory and its graphic  

subsystem are required. In practice, extra CPU speed is needed  

for rotating the three-dimensional data set and calculating pro-

jection image. Except for a very few high-end graphics work  

stations, most of the computer systems that are readily available  
to the biologist do not have the hardware to meet this require- 

ment. An alternative is to precalculate images and to load the  

images onto an image buffer in the imaging device itself, pro-

vided it has enough memory for all the images. Then, updating 

an image is done by sequentially copying the new image from  

image buffer onto the same location in the viewport buffer. Most  
inexpensive imaging devices take less than one tenth of a second  
to copy an image, providing the necessary speed. For this rea-

son, having a large image buffer in which to store the pre-com-

puted views becomes an important criterion in selecting graphic  

and image display hardware.

Choice of Display Hardware

Based on the above discussions, there are two major criteria  

for selecting display hardware. The first is that the display device  

must have a large image buffer for storing images. The second  
is that it must be able to perform an image copy operation with  

pixel replication. This function must be fast enough to allow  

smooth transitions in the movie mode (at least 5–10 frames/  

sec). After a careful study of most of the imaging graphic de-

vices commercially available, we picked the Parallax 1280 (Sun-

nyvale, CA) as the display device for the following reasons:

- It uses a high resolution, 1280 by 1024 pixel, 60 Hz, non- 

interlaced display. This provides sufficient space for dis-

playing 1024 × 1024 images with room on the right side for  

menus. The 60 Hz non-interlaced ability results in flicker-

free viewing even under fluorescent lights.

- It has 12M bytes of display memory which can be used either  
as three sets of eight bit pseudo color images or as a twenty
four bit true color system. The memory is organized as 2048 × 2048 with only 1280 × 1024 visible at any one time. Although pseudo color is more than enough for displaying gray scale images the true color mode is very useful for transparently overlapping images without losing any intensity information. This is obviously very important for examining multiply-labeled data sets.

- The Parallax 1280 processor is able to perform pixel replication and BLIT (Block Image Transfer) simultaneously at the speed of 12Mpixel/sec which is fast enough to support the movie function on the entire 1024 by 1024 display area.
- It supports a specialized macro instruction set. By loading the proper macro program, the Parallax is able to track the movement of a pointer device (such as a mouse) and represent it as an on-screen cursor completely independent of the host computer. The mouse can be programmed so that when the status of any button changes, a host interrupt will be generated, and the current screen position and button status will then be sent to the host.

Developments in hardware of this type are constantly being made, however, and interesting systems are currently offered by Indec Systems (Sunnyvale, Ca), Vital Images (Fairfield, Ia), ISG Technologies Inc. (Toronto, Canada) and others.

Model Building in PRISM

A model is defined as a stick figure representation of the path of a structure in three-dimensional space. It can be considered as a collection of separate objects. Each object is defined as a set of connected nodes which represent a single branch in the structure. A tree data structure is used to store the connectivity of all these nodes, and a position array is used to store the coordinates of every node in real space. To increase the generality of this data structure, the tree structure does not impose a limitation on the depth of the tree or on the number of entries at any level. The only limitation derives from total available memory space. For model display, each object has its own set of display attributes which PRISM uses to determine its visibility and color. On the node level, PRISM allows users to attach any of six predefined marks to a node. Thus when it is displayed, the attached mark will be shown at the same location and in the same color as the object to which it belongs. While building the model, PRISM maintains a data pointer within the tree structure. It determines the location in the tree structure where model command action takes place and the node pointed to by the mouse is called the active node. There are four basic commands for updating a model. They are ADD, DELETE, BRANCH and MODIFY. ADD will add a node right after the active node, and make the new node active. DELETE will remove the active node from the structure, and make the previous node active. BRANCH will create a new branch substructure at the active node, and set the first node in the new branch as the active node. MODIFY will not affect the tree structure, it simply alters the coordinates of the current active node. Besides these basic building commands, PRISM also provides a set of
commands for moving pointers within the tree structure so that any part of model can be retrieved easily.

Model Building

Building a model is accomplished by determining the locations of all the nodes. When displaying section images, a data point on the screen will correspond to a unique location in the data sampling space, as each image represents a slice in the sample. For such data, model building is accomplished by picking data points displayed on the screen. Stepping the section forward or backwards (which occurs rapidly because they have been preloaded) provides the third coordinate. As described in the previous section, the Parallax mouse will generate an interrupt whenever a mouse button is pushed, making it easy to send the screen coordinates to the host computer. Selecting nodes is also done by simply pointing to the data location on the screen with the cursor and pressing one of the mouse buttons.

When dealing with complicated objects, it is often necessary to be able to examine the data stack from multiple viewing angles. As described above, the program ROT3D can computationally reslice the data stack in any orientation. It is then possible in PRISM to open several windows on the screen and have each of them display images of the same data, but sliced in a different orientation while building the model. PRISM will use the header information to keep track of the geometry of each window. Thus, in the model building function, PRISM will choose what (usually orthogonal) section to display in each window so that they all contain the same spot in three-dimensional space as defined by the cursor and the section number in the active window. This approach to window management and model building allows the PRISM user to simultaneously construct a model from multiple viewing angles (Figure 5). The models can only be updated when the cursor is in the active window which displays the section image. Modeling within multiple projections images (such as stereo pairs) can be performed but requires the use of a stereo cursor which has not yet been fully implemented. Nevertheless, a projection can be a very good reference window to check any mistakes made in the model as shown in Figure 6. In this regard, it is possible to allow the model to be corrected in the plane of the projection image in the active window. To aid in the identification of individual substructures, a computational method has been developed which extracts data within a defined radius perpendicular to the 3D line model and then linearizes the model and the surrounding extracted data as shown in Fig 6 (left).
Superimposing the Model on a Background Image

This is necessary for both model building and display. There are two types of background images: projection and section images. For projection images, the procedures for superimposing a model on the active window are:

- Transform all the model coordinates to the same orientation as the background image
- Project the 3D model onto the same imaginary projection plane as the background image
- Transform the projected 2D model to screen coordinates
- Clip the 2D model within the window boundary and display the model overlaid on an appropriate section or projection from the original 3D data set.

For section images, the procedures are the same except for the second step. Instead of doing the projection, the model will be clipped according to the boundary defined by the appropriate slice in the data stack. All the geometrical transformations are made possible because of the information stored in the image file header and the design of the window system. For the non-active windows, the procedures are the same except that in section windows, the background images need to be updated so that they contain the data point which is designated as the active node before overlaying the model.

Future Development and Discussion

The major drawback of PRISM is the need for precomputing all the images which are used for 3D data analysis. This is because the host microVAX computer is too slow to provide on-line calculations for generating these rotate-projected images. With today's high-end workstations which drastically increase the performance of both the CPU and its graphic subsystem, the existing problem can be eliminated. At UCSF we are currently designing a second generation image display system which will greatly improve both three dimensional image data display and analysis function in PRISM. The new system will be implemented on the Titan workstation (Ardent, Sunnyvale, California) using it as both a computing engine and graphics display device.

With this system we will be able to provide a display functionality which allows real time, random location and orientation of a 128 by 128 by 30 subregion of a larger three-dimensional data set. Manipulating the entire data set will be possible, but at slightly slower than real time. This will, for the first time, give the viewer complete freedom in visualizing complex three-dimensional images of biological specimens.

A similar software system that permits rapid display of three-dimensional images without window or model building has been developed by Argiro (Van Zandt and Arigo, 1989) to run on Silicon Graphics workstations.

ACKNOWLEDGEMENTS

This work has been supported by funding from the Howard Hughes Medical Institute and by grants from the National Institute of Health to J.W.S. (GM-25101) and D.A.A. (GM-31627). D.A.A. is a National Science Foundation Presidential Young Investigator.

REFERENCES