

Chapter 10

3D Reconstruction of Confocal Image Data



Thomas C. Trusk

10.1 Introduction

The main advantage of the confocal microscope is often said to be the ability to produce serial optical sections of fluorescent samples, ultimately for the purpose of reconstructing microscopic objects in three dimensions (3D) (Carlsson and Aslund 1987). There are many ways, and reasons, to reconstruct confocal image data. As an example, consider the sample of embryonic mouse heart shown in Fig. 10.1 reconstructed using a variety of 3D techniques. This chapter will introduce these methods and discuss such topics as (a) why one might want to undertake this task, (b) some definitions of the representation of 3D space using images, (c) the different types of 3D representations that can be made, and (d) the necessary steps to making useful reconstructions. Along the way, the limitations and potential pitfalls that arise will be discussed.

10.2 Why Reconstruct?

Reconstructing image data into three-dimensional representations is a fairly recent activity made necessary by the invention of devices which virtually deconstruct their subjects by scanning serial planes at different depths. Positron-emission tomography scanners, computed axial tomography scanners, magnetic resonance imagers, ultrasonic imagers, and confocal microscopes are all examples of devices which can collect these 3D image datasets. Prior to the development of those tools, building accurate 3D models of microscopic biological specimens had always been

T. C. Trusk (✉)
Department of Regenerative Medicine and Cell Biology,
Medical University of South Carolina, Charleston, SC, USA
e-mail: TruskT@musc.edu

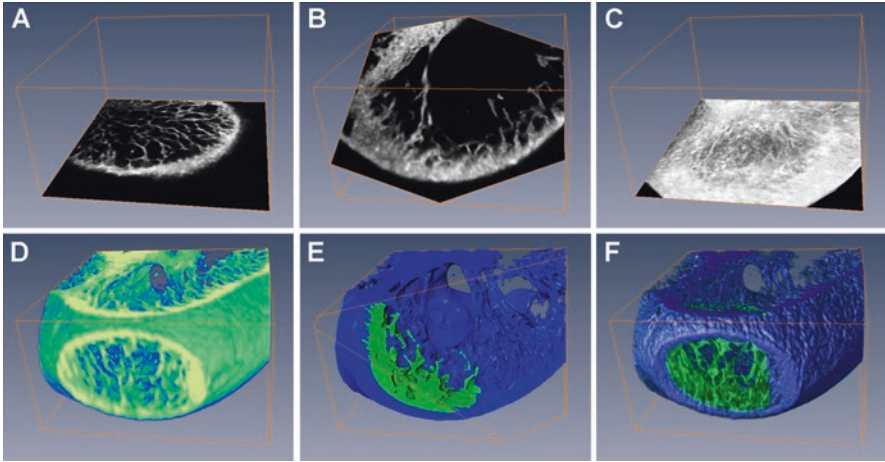


Fig. 10.1 Sample 3D reconstructions. The left ventricle of an embryonic mouse sampled in a confocal microscope. The 3D image dataset is contained inside the orange box where only the first confocal optical slice is shown (a). Methods which sample the 3D dataset include (b) an oblique slice, (c) a maximum projection through the z-axis, (d) a simple volume render, (e) a surface render manually segmented to highlight interior features, and (f) a volume render using complex lighting and segmentation information derived from (e)

a daunting task, that is, until powerful computers and sophisticated software were introduced. Instead of using scissors, cardboard, wax, modeling clay, or Styrofoam, it is now possible to sit in front of a computer workstation and construct, in a fraction of the time, a model that can be displayed in multiple forms and used to convey a wide variety of useful information.

Historically, the primary reason to reconstruct in 3D has been to display the morphology of the subject. In many cases, the 3D shape of the microscopic object may be completely unknown, or some experimental condition may influence its structure. It might also be helpful to reconstruct the various compartments of the subject and divide it into organs, tissues, cells, organelles, and even domains where some fluorescently labeled protein might be expressed. These arrangements often produce clues concerning function (e.g., see Savio-Galimberti et al. 2008). The value of a computerized 3D reconstruction is that the finished reconstruction can be viewed at different angles, including visualizations from perspectives impossible to achieve in the confocal microscope. Furthermore, most 3D software applications provide tools to produce digital movies, including interactive surface slicing, transparency, or subregion coloration in the models. These animations can be invaluable in conveying the structure of the subject.

3D reconstructions can be used to provide a substrate for the superimposition of other information (Hecksher-Sorensen and Sharpe 2001). For instance, the multiple channels used in confocal microscopy can be used to highlight the domain of proteins bound to fluorescent markers emitting different colors. Depending on the experimental question, these domains may be expected to be mutually exclusive or partially (or even completely) correlated within the structure of the subject. In another example, quantitative information (such as cell density or rate of

cell proliferation) may be overlaid as a coded color scheme onto a reconstructed object and provide a visualization of that property throughout the sample (Soufan et al. 2007).

Finally, once a fully segmented, surface-rendered reconstruction has been made, it is a simple matter to extract quantitative information from the sample. This includes counting objects, determining volumes, measuring surface areas, or quantifying the spatial arrangements of parts (such as measuring distances or angles between features).

10.3 Defining the 3D Space Using Images

The 3D Cartesian coordinate system (Fig. 10.2) provides a convenient space within which to reconstruct image data. The Cartesian system consists of three physical dimensions – width (x), length (y), and height (z) – which are perpendicular to each other (a relationship termed orthogonal) and are typically oriented with the z dimension pointing up. Individual images are considered orthogonal slices through the z dimension, and the pixel plane of each image is defined in the x and y dimensions. Figure 10.1a is an example of an orthogonal slice. Planes cut through the Cartesian space that are not perpendicular to a major axis are said to be oblique, as in the slice shown in Fig. 10.1b.

10.3.1 Perspective

When an observer views a three-dimensional scene consisting of objects of equal size, objects that are off in the distance will appear smaller than objects closer to the viewer. An artist (and a computer) will enhance the perception of depth in rendering

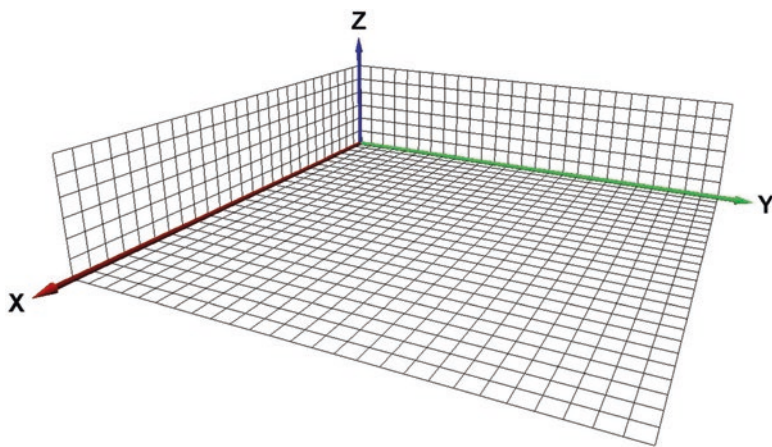


Fig. 10.2 The 3D Cartesian system. A mathematical model of three-dimensional space. Every position in the space is defined by its location along one of three axes (x , y , and z)

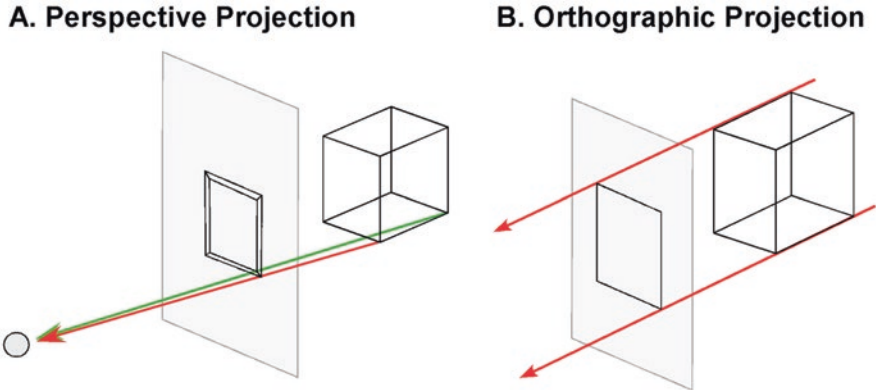


Fig. 10.3 Perspective vs orthographic projection. For objects drawn on a screen in perspective (a), all rays project from the object to the viewer. The more distant back face of the box appears smaller than the front face. In an orthographic projection (b), all rays projected from the object to the display screen remain parallel, and the perception of depth in the scene is lost

a scene by drawing objects “in perspective,” as if the drawing was traced on a window through which the viewer was looking. In Fig. 10.3a, rays exit the object and project directly to the viewer. The object is drawn on the display such that the back face of the box appears smaller than the front face. In an orthographic projection (Fig. 10.3b), all rays passing through the scene toward the display are parallel, and the object on the display does not appear in perspective. These types of renderings are sometimes useful, as in architecture, where object size comparisons are more important than the perception of depth. Most 3D software suites can draw objects on the screen with or without perspective, as the viewer chooses. The perception of depth in a scene is quite helpful to the viewer, especially when objects may be moving in dynamic displays or animations. Objects drawn orthographically and rotated often appear to suddenly switch rotation direction.

10.3.2 Voxels

A single confocal “optical slice” has a “thickness” in the z dimension that is defined by the microscope optics and the confocal pinhole. Recall that increasing the diameter of the pinhole allows extra light from above and below the focal plane to contribute to the light gathered at each pixel. On the other hand, a single “image” is infinitely thin along the z dimension (see Fig. 10.4). Confocal z-series images are typically collected at evenly spaced intervals, and when the images are restacked at the correct z distance, the empty space between the images is readily apparent.

In order to fill the space between images, it is necessary to expand the concept of a two-dimensional image pixel into a three-dimensional volumetric voxel. This is demonstrated for a single voxel in Fig. 10.5. The grid of two-dimensional image

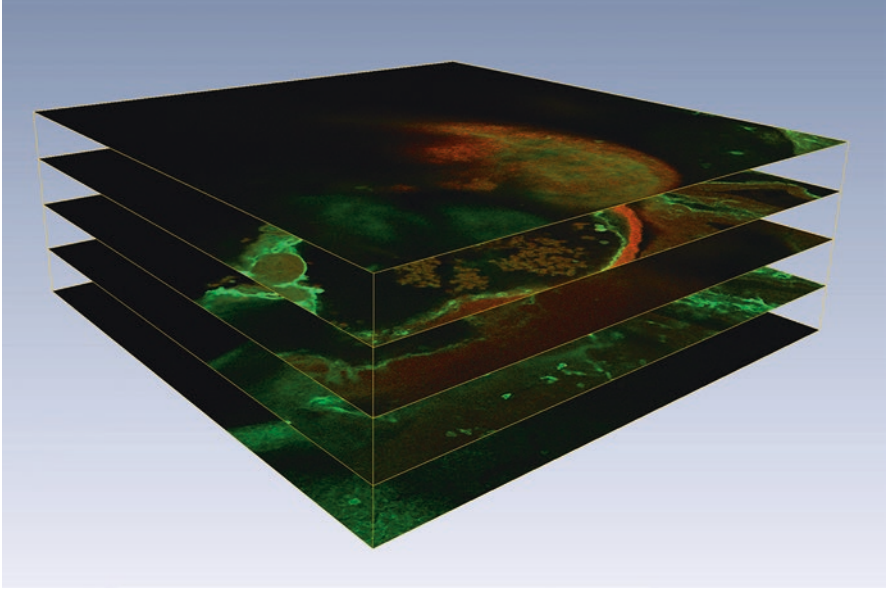


Fig. 10.4 A single image has no depth. Five evenly spaced confocal slices demonstrating the infinitely thin nature of image data in the depth (z) plane

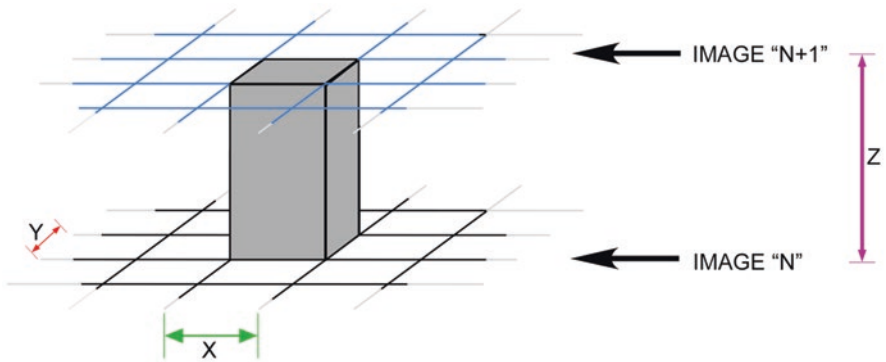


Fig. 10.5 Definition of a voxel. A portion of two consecutive images from a confocal series is shown. Image pixel dimensions were x units by y units, and images were collected every z units. A single voxel is the space defined by the pixel dimensions and the distance between consecutive images

pixels is stretched in the z dimension to produce three-dimensional voxels. Each voxel has the same dimensions of single image pixels in the x and y dimensions and the distance between consecutive images in the z dimension. Thus, the volume of each voxel is defined as the area of each pixel (x multiplied by y) multiplied by the distance between optical slices (z). This voxel volume is the smallest unit of volume that can be measured in that image dataset.

One can easily imagine that voxels are the “bricks” used to construct a 3D model. Each voxel not only has spatial dimensions, but each voxel also has a single unique brightness value inherited from its basis pixel. In an 8-bit sensitivity system, this value will range from 0 to 255 and is constant within the confines of a single voxel. Much like the bricks used to construct buildings, the voxels in a 3D image dataset are all the same shape and size, and each voxel has only one homogeneous color.

10.3.3 Resolution

The shape of individual voxels in an image dataset is determined by the resolution of the system. Some confocal microscope software controls will, by default, choose to optically slice a sample using a z distance that is as close as feasible to the pixel size. This would result in voxels that are nearly cubic, where x, y, and z dimensions are approximately equal. This is desirable for many reasons. First, cubic voxels more easily reconstruct 3D features correctly no matter what orientation they may have. For example, if you were to choose to use bricks to build a sphere, what shape brick would make the best representation? Figure 10.6 displays surface reconstructions of the same sphere rebuilt with different-shaped voxels and demonstrates the effect of increasing the voxel z dimension (or slice thickness) on the surface of reconstructed objects.

Most mathematical functions used in 3D reconstruction and visualization are much more efficient when the dataset uses isotropic or cube-shaped voxels. Using extremely anisotropic voxels can lead to inaccuracies in many measurements, especially surface area, and produce 3D visualizations that lose resolution when viewed from particular angles (see Fig. 10.7).

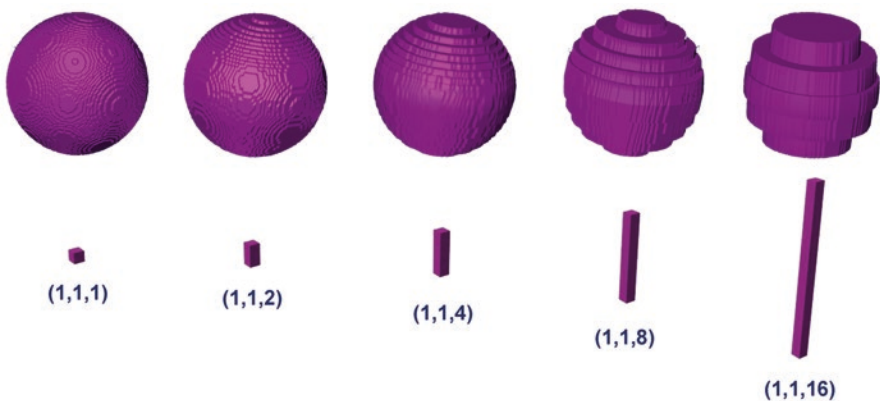


Fig. 10.6 Effect of voxel shape on surface reconstruction. The same spherical object is surface-reconstructed with voxels of increasing z dimension. The voxel shape used is shown (much magnified) below each rebuilt sphere with the voxel dimensions in x,y,z. The leftmost sphere was rebuilt using cubic voxels

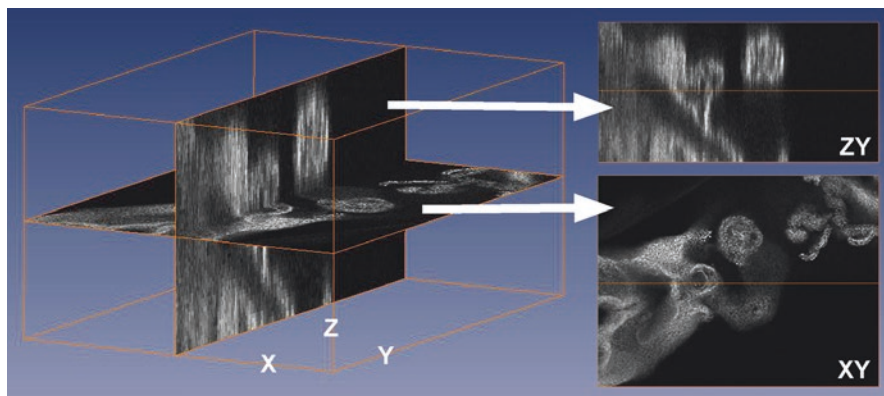


Fig. 10.7 Effect of voxel shape on 3D visualization. A confocal 3D image dataset is sliced in the XY plane (as originally collected by the confocal system) and in the ZY plane. Voxel x,y,z dimensions in micrometers are 2.4, 2.4, and 12, respectively

One solution offered in 3D reconstruction software is to resample the image data into near-cubic dimensions. That is, the original data will be interpolated to fit a cubic voxel model. However, this approach only helps in the calculation efficiency and does not improve the visualization of the sample. Resampling the data into larger voxels can be quite helpful if the computer memory is insufficient, but resampling into smaller voxel sizes will only make the volume take up more memory and not increase visible resolution at all. It is worth noting that resolution decisions are best made at the microscope when acquiring images, and this is demonstrated in Fig. 10.8. The same epithelial cell was captured using a 20 \times lens, a 40 \times lens, and a 40 \times lens with a digital zoom of 5 (for an effective 200 \times magnification). The actual collected pixel sizes were 0.73 μm , 0.37 μm , and 0.07 μm for the 20 \times , 40 \times , and 200 \times images, respectively. In each image, a white box identifies the same mitochondria, and the image data inside has been magnified to show individual image pixels. It should be easy to discern that the visual information obtained at 20 \times would be difficult to resample (or subdivide) into the higher resolution information seen at higher optical magnifications. This demonstration of resolution in the x and y dimensions also applies to the z dimension.

Resolution in the z dimension is set at the confocal microscope by choosing an appropriate distance between optical slices. For 3D reconstruction work, a good rule of thumb is to optically slice at a distance between 1 and 5 times the size of pixels in the XY images. Oversampling the object by slicing at smaller intervals is easier to correct later by resampling. If you are worried about datasets that are too large or take too long to collect, consider that the cost of data storage is essentially free. Returning the sample to the microscope later is always risky as fluorescence may have waned or the tissue may have deteriorated.

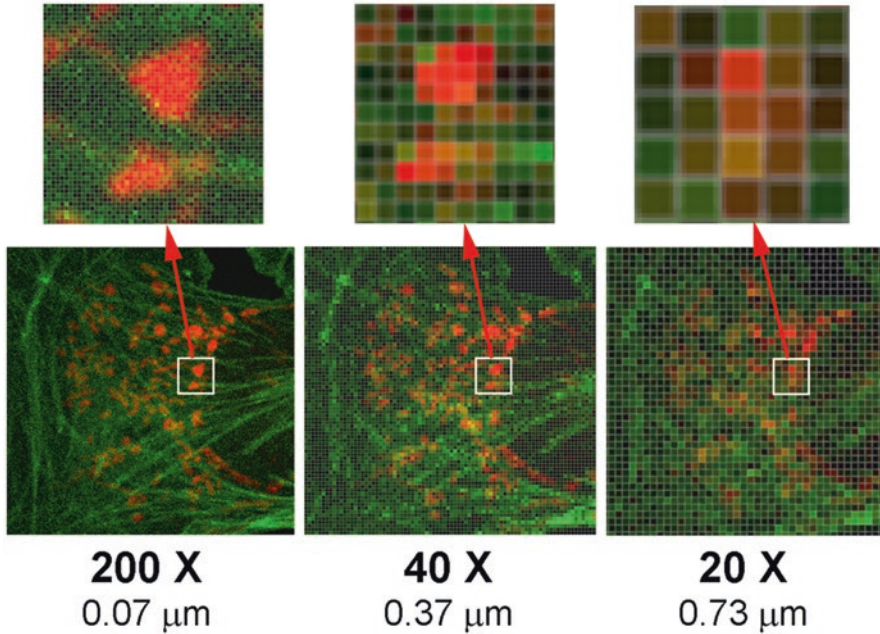


Fig. 10.8 Changing resolution. A bovine epithelial cell acquired on a confocal microscope using increasing objective power to improve resolution. The actual pixel dimension resolved is given for each image in micrometers. The white box region in each image has been magnified to reveal actual image pixels

10.4 Types of 3D Reconstruction

3D image datasets can be reconstructed and visualized in a variety of ways. These 3D models range in complexity from projections, a common function within most confocal system software suites, to surface reconstructions, which often require additional software. More recent confocal microscopes now include a sophisticated selection of reconstruction routines as part of the operating software, or possibly as optional additions. There are three basic methods used to present a 3D dataset: projections, volume renders, and surface reconstructions. The following will briefly describe how each type is calculated.

10.4.1 Maximum Projections

A projection is a single two-dimensional image generated using data from the entire 3D image set. In the case of fluorescent images, this consists of finding the brightest, or maximum pixel value at each pixel address from each image in the entire dataset, and projecting it onto a result image (see an example in Fig. 10.1c). While the

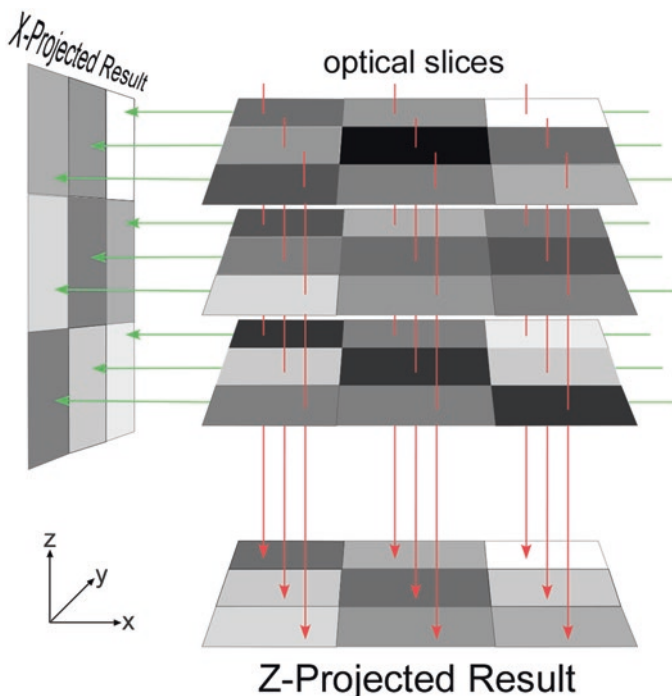


Fig. 10.9 Projected images. Three optical slices, each 3 × 3 pixels, are projected along the z-axis (red lines) and along the x-axis (green lines). The maximum (or brightest) pixel values along each projection line are placed in the resulting projected image

obtained image itself is not a 3D object, it conveys a visualization of the entire object from a particular viewpoint. Typically, the viewpoint used has been orthogonal, or along a principal axis (x , y , or z), which simplifies the mathematics of calculating a projected image.

Consider the simple 3D dataset in Fig. 10.9. Three optical slices, each a 3 by 3 pixel image, are projected along z-axis (red) and x-axis (green) projection lines. The brightest pixel encountered along each line is placed in the result image. Note that these projection lines are parallel to each other; thus they produce an orthographic rendering of the data, as previously discussed. As such, projected images tend to lack perceived depth.

An illusion of depth can be partially elicited by making a projection animation (or multi-frame movie) where the object appears to rotate in place, either in complete circles, or by alternately rocking the object clockwise and counterclockwise. These animations are generated by mathematically rotating the object a few degrees and recalculating the maximum pixels encountered along parallel rays projected toward the result image (see Fig. 10.10). When the frames are collected into a movie and played in sequence, the projected object appears to rotate. Recall that the resolution along the z-axis (as seen through the microscope) is always best and that resolution

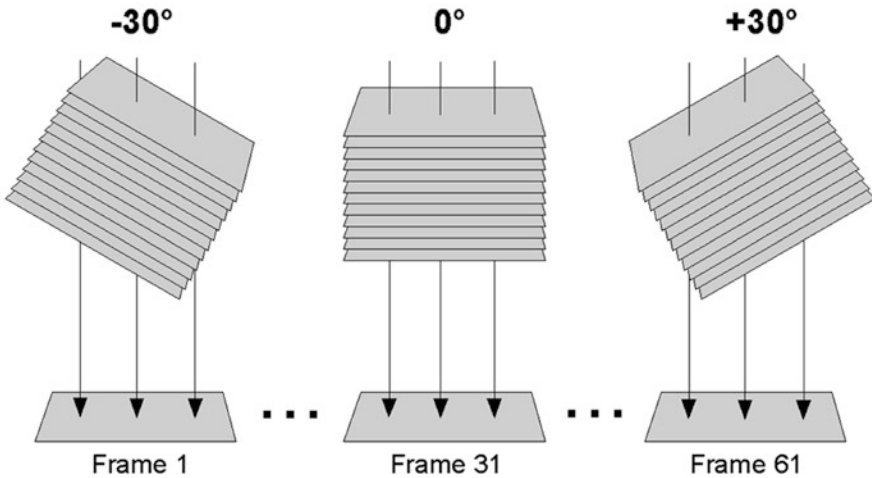


Fig. 10.10 Maximum projection animation. A 61-frame animation is made by rotating a 3D image dataset one degree before each maximum projection result frame is calculated. When the frames are collated into a movie, the projected image appears to rotate in place

is reduced when the data is viewed along the x- or y-axis. This is why projected images tend to appear to lose clarity as the object rotation exceeds $30\text{--}45^\circ$ away from perpendicular to the z-axis. For this reason, most animations are made to show the object shifting rotation between $\pm 30^\circ$ of tilt.

A common mistake is to slice the confocal sample at step sizes 10 or more times the size of individual pixels. This will produce a 3D dataset that will lose resolution too rapidly as the object is rotated. The illusion of depth is also lost when the number of slices is very small when compared to the pixel dimensions of each XY image. For example, 10 z-slices taken at 1024×1024 resolution will make disappointing rotation projection animations. The best projection animations use 3D datasets with z-slice intervals close to xy pixel size (i.e., near-cubic voxels) and small rotation factors between projection animation frames (about 1° of rotation for each frame).

Maximum projections are the most commonly seen 3D reconstructions, and all confocal software systems routinely generate orthogonal views at the press of a button. Procedures for making projection animations are also widely available. Projections work best as orthogonal visualizations of 3D fields with little depth and provide the “all-in-focus” micrographs difficult to obtain on widefield fluorescent microscopes. However, be warned that maximum projections can also be misleading. When the object is hollow with internal structure, like the heart ventricle shown in Fig. 10.1c, the external signal will mask the interesting internal features. More importantly, maximum projections of multichannel image sets will mix colors by addition. Red and green channel signals that lie along similar projection lines, but are not otherwise colocalized, will produce yellow objects in the result image. This could lead to misleading conclusions.

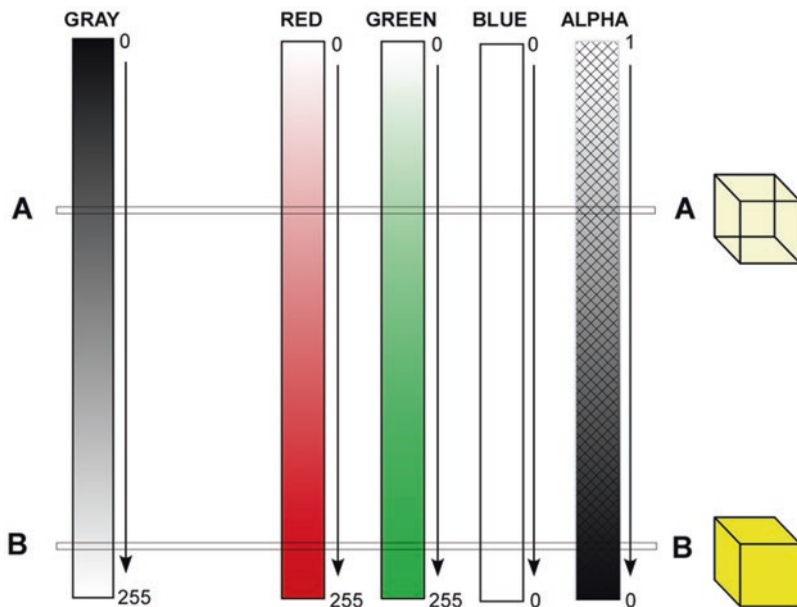


Fig. 10.11 RGBA lookup table. An 8-bit lookup table is used to convert gray values from a voxel into the corresponding red, green, blue, and alpha (transparency) values used for display. Red and green table values range from 0 to 255, blue is 0 across the range, and alpha ranges from 1 (transparent) to 0 (opaque). The gray values at A and B are converted in the mixed RGBA values and displayed in voxels at the right. The darker value at A mixes red and green to produce a transparent weak yellow voxel, while the brighter value at B produces a more saturated and opaque yellow voxel

10.4.2 Volume Rendering

As acquired in the confocal microscope, each voxel can be described with certain x,y,z spatial coordinates and a fluorescent intensity recorded in each channel. In order to visualize a 3D dataset in proper perspective, we need to render an image on the display in a manner which will allow internal structure to be correctly observed. One way to accomplish this is to arbitrarily assign each voxel an additional attribute of transparency. This attribute is termed alpha in imaging technology, and it has an inverse commonly used in 3D rendering called opacity, i.e., a voxel of 100% opacity has no alpha transparency. Alpha ranges from 0 to 1; thus opacity is $1-\text{alpha}$.

By now you should be familiar with the concept of pseudo-coloring images using color lookup tables. For 3D renderings, alpha becomes a fourth attribute, and voxel pseudo-color values are often referred to as RGBA tuples, referring to the four values assigned to each voxel. A typical RGBA lookup table is shown in Fig. 10.11 for two possible voxel values. In fluorescent images, the background is typically dark (not fluorescent). Since we wish to see through nonfluorescent regions,

we assign the most transparent alpha values to the darkest image pixels. In this particular example, we choose to make brighter image values to produce more saturated yellow voxels that also become less transparent (or more opaque) as intensity increases. This RGBA table will produce brightly stained fluorescent objects as yellow structures and leave the unstained background transparent.

In the process of calculating a volume rendering, imaginary projection rays proceed from the display screen and completely penetrate the 3D dataset. If the display is to be orthographic, the rays remain parallel from the display through the object. For displays remaining in perspective, the rays appear to originate from the viewer and are not parallel. The number of rays possible can be quite high, but only enough are needed to fill each pixel on the display screen. As this limited number of rays pass through the object, the software has the choice of using all of the data, where calculations are made from every voxel the ray passes through, or the data can be interpolated by averaging across a select number of neighboring voxels to speed up the process for more dynamic displays.

The first steps in the process of calculating voxel values used along a projection ray are shown in Fig. 10.12. Consider a small sample, a $3 \times 3 \times 3$ volume. The volume is shown in the brick model in (A) and, in another conceptual format, the cell model in (B). In the cell model, voxels are viewed as the vertices of a three-dimensional grid. A single projection ray is shown in (C) as it passes through the volume and intersects the display pixel where the result will be seen. As the ray passes through the volume, it passes through several cells, each defined by the 8 voxels in the cell corners. The intensity value used for each cell is calculated using trilinear interpolation, as shown in (D). In the first approximation, four pairs of corner voxels are averaged, then those four values are averaged, and then finally the last pair is averaged. The final value is converted using the lookup table into an RGBA tuple. This process is repeated for all cells along the projection ray, and the final step is to composite all of these tuples into a single value to be displayed.

In compositing these values, calculations usually proceed from the back of the object space to the front, toward the viewer. As the ray first strikes the object, it has the color or intensity value of the background. The intensity passing through each cell in turn is modified according to the formula:

$$\text{Intensity}_{\text{out}} = \text{Intensity}_{\text{in}} \times (1 - \alpha_{\text{cell}}) + (\text{Intensity}_{\text{cell}} \times \alpha_{\text{cell}}),$$

and this process continues until the ray leaves the object, where the final intensity_{out} is placed in the display pixel.

Most modern computer video display adapters are now capable of performing much of these 3D rendering computations very rapidly. This makes it possible for most modern desktop computers to perform volume renders as quickly as expensive graphic workstations of the recent past. The addition of hardware accelerators dedicated to 3D texture mapping, mainly used in modern gaming software, has also made it possible to generate even more complex volume texture renderings. For these methods, each cell can be given attributes beyond color intensity and transparency, such as texture mapping, specular or diffuse reflection, shadowing, and other lighting effects (as shown in Figs. 10.1f and 10.20c).

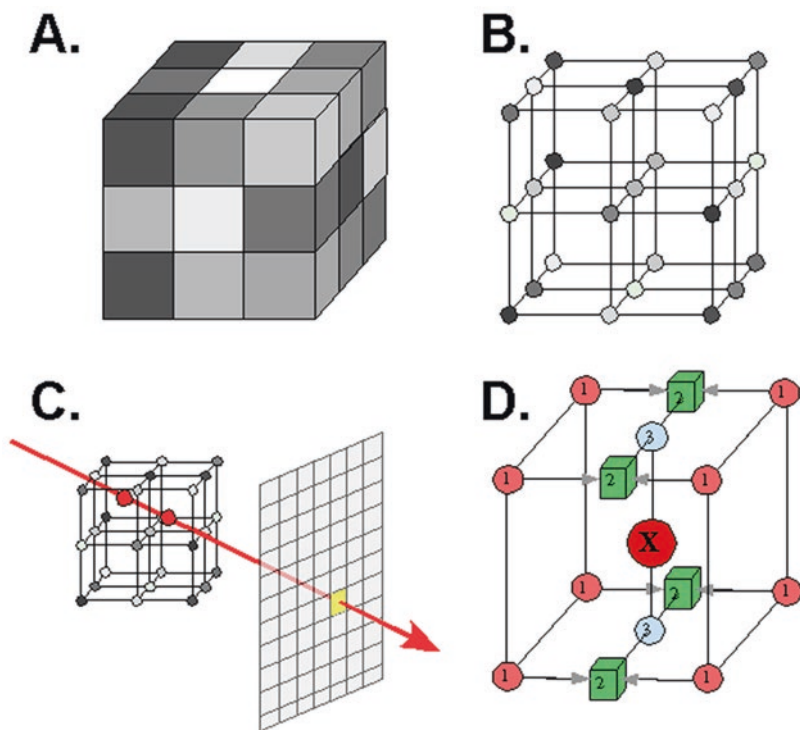


Fig. 10.12 Voxel averaging. A simple $3 \times 3 \times 3$ volume shown using the brick model (a) and the cell model (b). In the cell model, voxels are viewed as the vertices of a three-dimensional grid. A single projection ray is shown in (c) as it passes through the volume and intersects the display pixel where the result will be displayed. As the ray passes through the volume, it passes through cells, each defined by the eight voxels in the corners. The intensity value used for each cell is calculated using trilinear interpolation, as shown in (d). In the first approximation, four pairs of corner voxels are averaged, then those four values are averaged, and then finally the last pair are averaged

Volume render animations more faithfully display internal structure as compared to maximum projection animations and are less likely to generate false positive conclusions regarding co-localization. However, a single frame volume render of a multichannel 3D dataset can be just as misleading regarding co-localization as a maximum projection of the same dataset. Depending on the software, multichannel volume renders will often mix RGBA tuples for the final display image or may choose to display one channel as masking another. It is best not to rely on one visual aspect when ascertaining co-localization.

10.4.3 Surface Reconstruction

In image-based reconstruction, a surface is a three-dimensional polygonal boundary surrounding voxels of special interest. These voxels can be defined by setting a single intensity threshold value or by including a specific range of intensity values.

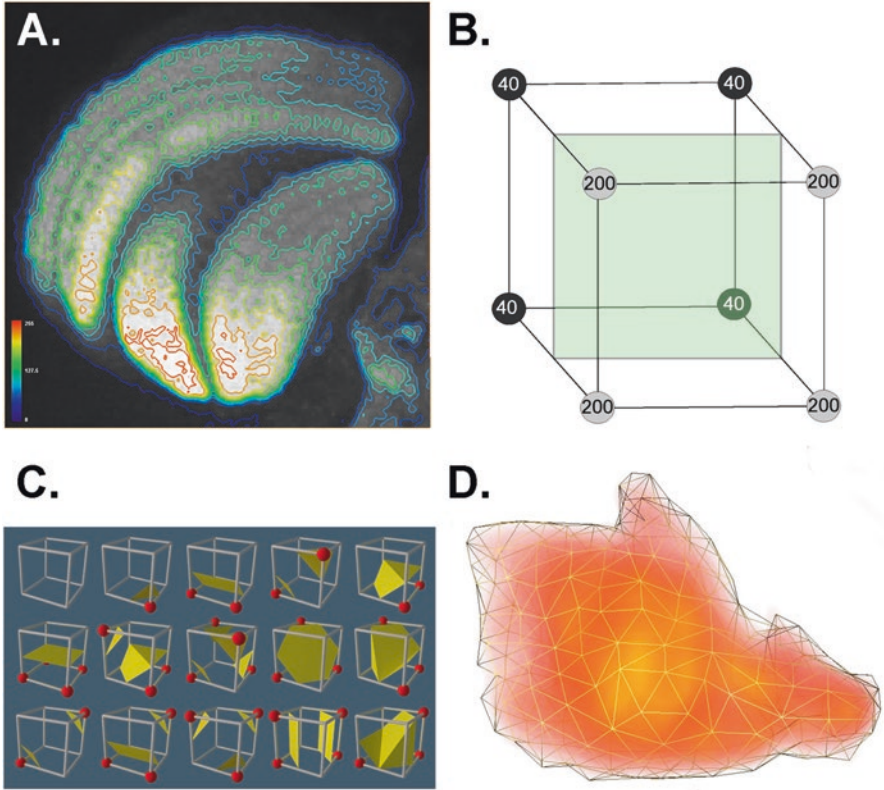


Fig. 10.13 Surface reconstruction. (a) Intensity isolines in a two-dimensional image. (b) A single 8-voxel cell is subdivided by a green surface boundary based on voxel intensities above 100. (c) The 15 possible solutions to the marching cube algorithm. (d) A 3D polygonal wireframe surface bounding volume-rendered voxels

Alternatively, individual voxels can be selected manually (in a process called segmentation) for inclusion into a surface boundary. A mathematical routine called the marching cube algorithm is typically used to define a surface in 3D space.

As an introduction to 3D surfaces, consider the simpler two-dimensional isosurfaces used to map intensity values in single images. We are used to seeing contour lines mapping altitude or isotherm lines showing temperature differences on weather maps. In Fig. 10.13a, the color-coded isolines surround pixels of a narrow range of intensity values. In 3D space, the isolines become surfaces which divide eight-voxel cells (as described above in volume rendering) to isolate voxels either by intensity or by segmentation. For example, given a threshold value of 100, the voxels of the single cell shown in Fig. 10.13b require the green surface to divide the 4 voxels of intensity 200 from the 4 voxels of intensity 40. This is how the marching cube algorithm functions, evaluating every cell in the volume so as to place boundaries sufficient to isolate voxels which either exceed a selected threshold or which have been

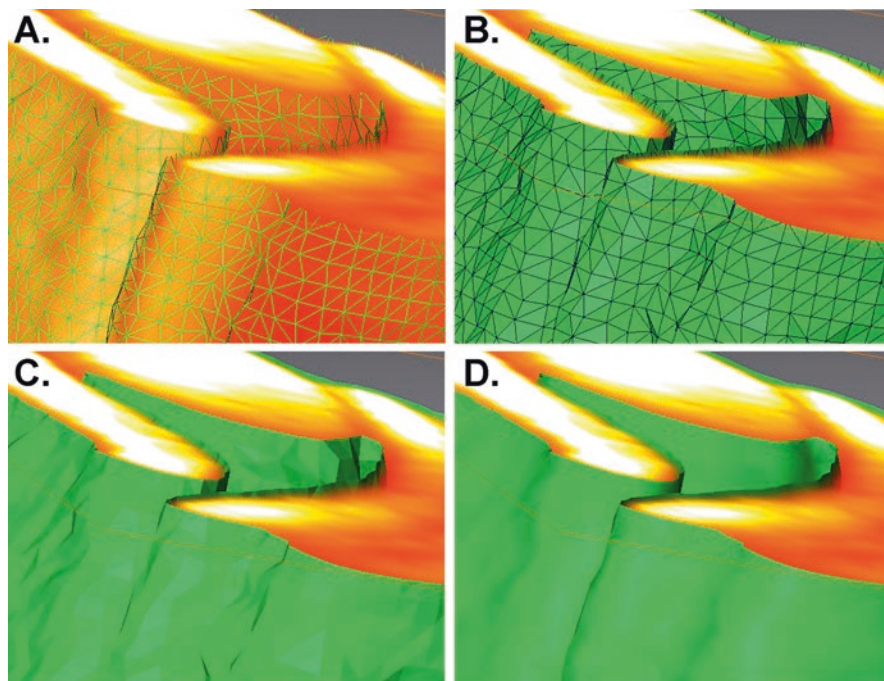


Fig. 10.14 Surface rendering. A portion of a folded surface reconstruction rendered as a wireframe (a), a wireframe filled with green (b), a green surface without the wireframe (c), and a green surface with reflection normals moved to the center of vertices (d)

manually selected. There are 256 possible solutions to an eight-voxel cell analysis, but given rotational symmetries, this number can be reduced to only 15 potential answers (Fig. 10.13c). After cleaning up some inconsistencies and imposing some rules governing maximal surface folding, the algorithm produces a set of vertices defining a mesh of triangles which enclose the selected voxels (D).

The number of triangle faces produced by the marching cube algorithm depends on the voxel resolution. An ideal solution contains just enough faces to efficiently enclose the 3D region of interest. However, the speed of rendering a surface is directly related to the number of faces to draw. Most efficient surfaces have less than 40,000 faces, yet the first approximation of the marching cube algorithm will often yield over a million faces. The face count can be reduced by either averaging across two or more voxels before executing the marching cube algorithm or by using a surface simplification routine on the final surface.

A variety of choices are available when rendering a surface to the display (Fig. 10.14). Drawing the mesh of triangles which connect the vertices defining the surface produces a wireframe (Fig. 10.14a) which is transparent. Triangles can be filled with a chosen solid color (Fig. 10.14b), and the wireframe triangles can be hidden (Fig. 10.14c). This reveals the faceted reflections of the light bouncing off the triangles. This facet effect is caused by the 3D lighting calculations of the display

system, which depend upon a surface normal that has been generated for each surface triangle. The normal is a spatial vector pointing away from the center of each triangle, and this vector is orthogonal to the plane of each triangle. The normal guides light in the scene directly to the triangle face, where it is reflected, producing a faceted appearance. This reflection can be removed by forcing the rendering engine to use normals centered on the triangle vertices rather than the center of triangle faces, which produces reflections of a smoother, more natural surface (Fig. 10.14d).

Surface reconstructions produce virtual representations of objects in 3D. While they appear less like the original object compared to projections and volume renderings, the segmentation process for surface reconstructions is a necessary first step to quantifying anything in the dataset. The process of segmenting dataset voxels based on intensity (or some other specific criteria) provides a framework for calculating volumes (the number of voxels multiplied by the volume of a single voxel) and counting objects (the number of independent regions with directly connected voxels). Surface areas are easily determined by adding together the areas of triangles enclosing virtual objects of interest.

10.5 The Steps to Reconstruction

Three possible reconstruction pathways are shown in Fig. 10.15. Each path is based on the main types of reconstruction that can be produced, and the flow chart points out the main steps that will be required to achieve the desired result. Note that results obtained in more complicated reconstructions may contribute to other types of visualizations. The steps to reconstruction can be reduced to seven basic processes that must be followed in order, but it should be clear that not every step is required to produce a 3D visualization.

Most 3D reconstruction software systems perform similar functions (Clendenon et al. 2006; Rueden and Eliceiri 2007), but it is beyond the scope of this chapter to produce a guide to reconstruction for each system. As is generally true with software, the more you pay, the more power you get. Each system should have a user guide to the functions, and with luck, some tutorials designed to reduce the learning curve for novice users. This section will present a more philosophic approach to 3D reconstruction and attempt to describe the generic steps to performing this complicated task.

10.5.1 Planning

When building (or rebuilding) any structure, it is always helpful to have a plan. In 3D reconstruction, this means gathering as much information about the sample as is feasible and to have some idea of what the final product will include. For example, what is the approximate real size of the largest and smallest features to be

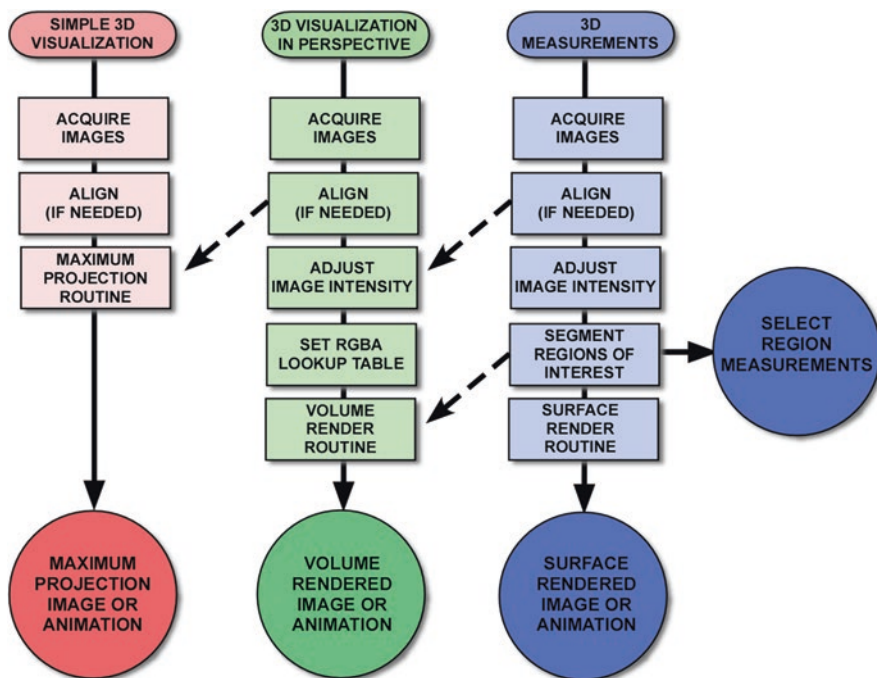


Fig. 10.15 The reconstruction process flowchart. The steps required to reconstruct will depend upon the type of result desired. (Red) Simple 3D visualizations generated using maximum projection methods only require a confocal z-series. (Green) Volumes rendered in perspective may need some intensity adjustment and the assignment of a suitable RGBA lookup table. (Blue) Surface reconstructions and any 3D measurements will require segmentation of the volume into regions of interest. The dashed arrows between pathways indicate that results from one path can contribute to other types of reconstruction. For example, the segmentation output of a surface reconstruction can be used to assign textures to multiple regions of a volume render (instead of using an RGBA lookup table)

reconstructed? That information will guide the microscopist to select an objective with enough resolving power and sufficient field of view. In some cases, there is no objective that will work well for both resolution and view field. This typically is true when the entire object is too large to fit in the view of an objective needed to resolve the smallest features of interest. In those situations, it's best to use the higher power objective for the best resolution of the smaller features. The microscopist should then plan to collect overlapping fields of view in acquisition and then realign all the parts into one complete dataset.

There should be specific goals in place. Ask yourself which products will result from the reconstruction? Are you attempting to visualize the sample from unique 3D perspectives? Will you need to subdivide the structure into components to study how they fit together? Do you wish to quantify some spatial characteristics of your sample? The answers to these questions will determine which tasks you must perform and will lead you to an efficient path to your goals.

10.5.2 Acquisition

Most of the content of this book is aimed at improving the readers' ability to acquire quality confocal images, but it is worth repeating that collecting the best dataset possible will greatly influence the reconstruction process. All of the issues previously discussed apply, especially maximizing contrast and resolution in the images. The basic rule is to enhance the visualization of the regions of interest as much as possible.

Your raw image data will be collected using your specific model confocal microscope's software system. Every manufacturer's software suite saves z-series image stacks in a different format, so it is important to understand how your system stores these images. Furthermore, every 3D reconstruction program reads these image stacks in a different manner, so you must also be prepared for the possibility that you might need to reformat your original data in order to move it into the 3D software. Hopefully your 3D software can read the raw data format of your confocal microscope, but if not, the best solution is to export z-series into TIFF format images using filenames that are in the correct order when alphabetically sorted. If you must export TIFF images, you must also keep different confocal channel images in separate, appropriately named folders. As explained elsewhere in this book, it is best to avoid using JPEG format when moving 3D datasets.

Another consequence of exporting TIFF images is that spatial information may not be read with the data into the 3D software. If you plan to produce quantitative information using physical units, be prepared to know the pixel size and z-slicing distance. Most 3D systems ask for the size of a single voxel as an x,y,z input. Be sure to report all of these values in the same units (e.g., micrometers).

10.5.2.1 Deconvolution

Deconvolution of confocal image data may seem unnecessary, but this process can actually remove a noticeably large proportion of the background noise, especially in the z-plane, and greatly assist in later reconstruction steps. Deconvolution should really be considered an adjustment method, but because its implementation has more stringent image acquisition requirements, it is best to mention it here. Any discussion of the process of deconvolution would be beyond the scope of this chapter, so the reader is greatly encouraged to consider recent papers for a general introduction (Biggs 2004, 2010; Feng et al. 2007; McNally et al. 1999).

10.5.3 Alignment

One of the more difficult and tedious tasks in reconstruction is aligning the serial slices of the 3D image set. When serial tissue slices from a microtome are mounted, stained, and photographed, the slices must be returned to their original positions by

transforming the images (moving in x and y directions and rotating). Often, the process of cutting the tissue will change the shape of serial slices as the sample is dragged across a sharpened knife. For confocal data, this step is completely unnecessary. The sample remains intact, the serial slices are obtained optically, and every image is collected in perfect alignment to the entire sample.

However, circumstances do arise when it becomes necessary to align multiple confocal datasets into one large dataset. Consider the case where larger features of interest cannot fit into the field of view of the objective required to resolve the smallest regions of interest. In these situations the entire range of features can be obtained by tiling 3D image sets together. Many modern confocal microscopes come equipped with automated x-y stages and software which will stitch together neighboring z-series into large 3D image datasets.

10.5.4 Adjustment

Typically, if you were carefully setting the gain and offset during acquisition, your confocal images should need little adjustment beyond minor changes in brightness or contrast. Please keep in mind the guidelines for scientific data as applied to images, such as those defined by the Microscopy Society of America (Mackenzie et al. 2006). However, it might be necessary to adjust the intensity values in the images to enhance the differentiation of regions of interest for thresholding or manual segmentation procedures to follow. Always remember that it is important to keep a record of these procedures and to report them in any scientific reports so that your results are repeatable.

Most 3D software suites provide image adjustment functions that will alter the brightness data in the images either in individual slices, across all slices, or even using 3D kernels as image filters. These functions are strictly to be used to help differentiate regions of interest when necessary. Some “automatic” segmentation routines work best when regions of interest have sharp boundaries based on intensity, and these functions can often help the efficiency of those routines. When manual segmentation procedures will follow, these adjustment routines greatly assist the observer in finding regional boundaries.

As an example, consider the application of the histogram equalization function. Histogram equalization is often used to increase contrast in an image by “equalizing” the frequency of pixel intensities across the available range. The effect of typical unrestrained histogram equalization is shown in Fig. 10.16a and b. For confocal images, this type of equalization tends to emphasize the out-of-focus epifluorescence, which tends to defeat the purpose of optical slicing. A better solution is to apply the histogram equalization function repeatedly over smaller, more contextual regions of the image in a procedure called local adaptive histogram equalization. This greatly maximizes contrast, making hidden regions more visible, but also greatly enhances the background noise and produces shadows at image intensity edges (Fig. 10.16c).

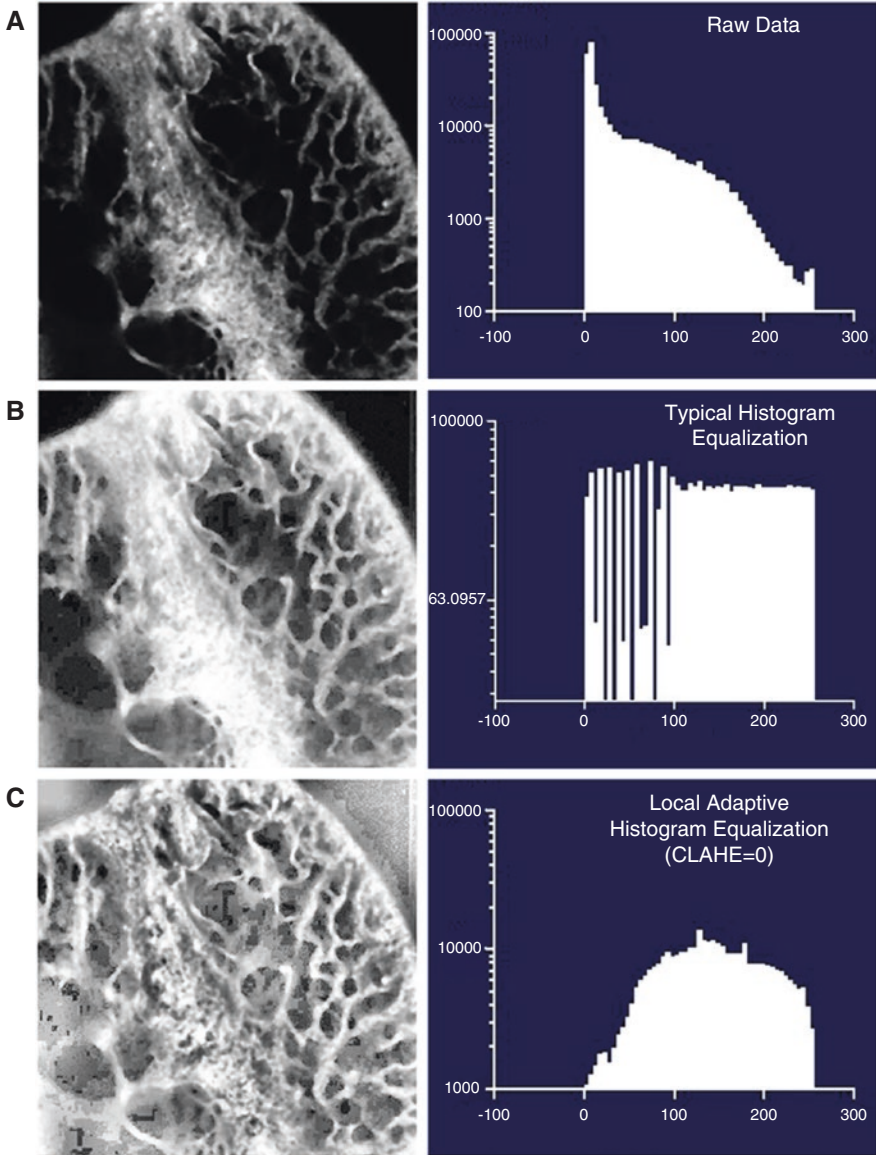


Fig. 10.16 Image adjustment. Histogram equalization. (a) A raw image and the corresponding histogram. (b) The result of a typical unrestrained histogram equalization. (c) Local adaptive histogram equalization without contrast limit. The effect of increasing the contrast limit in adaptive histogram equalization to 5 (d), 10 (e), and the maximal 15 (f)

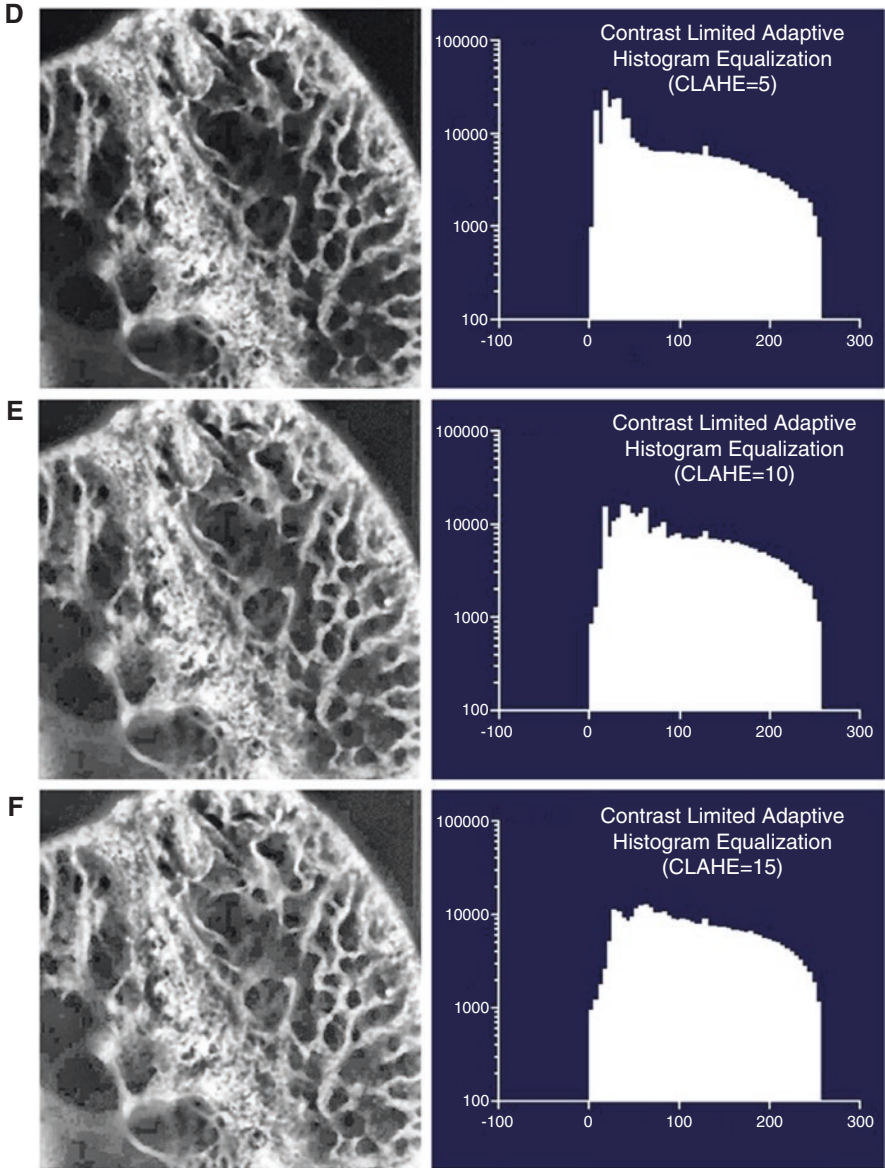


Fig. 10.16 (continued)

10.5.4.1 Contrast-Limited Adaptive Histogram Equalization (CLAHE)

This is a procedure originally developed for medical imaging and can be quite helpful in enhancing low-contrast images without amplifying noise or shadowing edges. Briefly, CLAHE works by limiting the amount of contrast stretching allowed in small local areas based on the contrast already present. So in regions where intensity is uniform, the amount of equalization is reduced, and the noise is not emphasized. The CLAHE effect is typically set by variably adjusting the degree of contrast stretching to be applied in local areas, and this is demonstrated in Fig. 10.16d–f. For the purpose of regional segmentation, the adjusted image in Fig. 10.16d would be far easier to differentiate than the original for both automatic and manual procedures.

10.5.4.2 Z-Drop

The sample itself will often interfere with both excitation light on the way in and emission light on the way out (Guan et al. 2008; Lee and Bajcsy 2006; Sun et al. 2004). The effect of this is to produce an image set where the average intensity of each optical slice appears to be reduced in deeper z-sections (a phenomenon termed z-drop, see Fig. 10.17). Most automated segmentation routines used threshold-based rules based on voxel intensity; thus it would be desirable that voxel intensity within regions of interest be uniform throughout the sample. Many 3D reconstruction systems offer routines to correct for z-drop as an adjustment procedure.

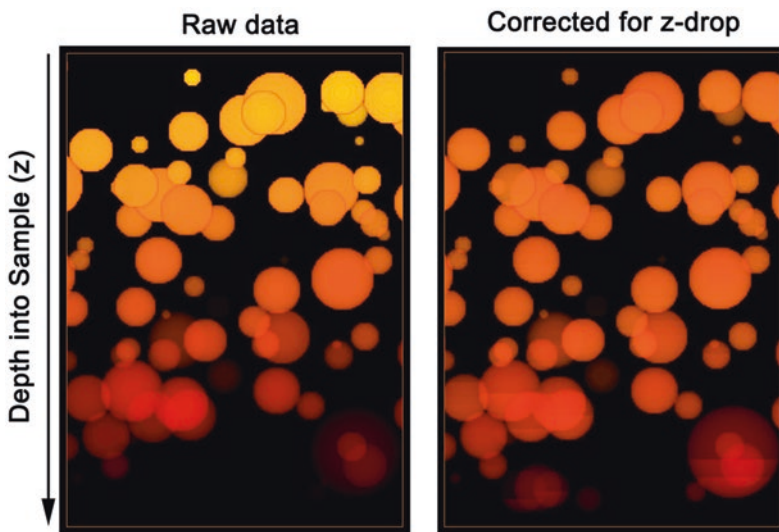


Fig. 10.17 Correcting for z-drop in confocal data. Volume rendering (XZ view) of confocal dataset showing the effect of z-drop through the depth of the sample (right). Following correction (left), voxel intensities have been adjusted so that average intensity is more nearly constant with depth

It is often possible to correct z-drop in the acquisition process by systematically adjusting laser intensity or detector gain (or both) as deeper optical slices are collected. Many confocal microscope control systems include a routine which will adjust these settings with optical depth, usually based on linear models of the z-drop effect.

10.5.5 Segmentation

Segmentation is the process of dividing the image dataset into regions of interest. This is often the most time-consuming task and typically requires a great deal of expertise. It is worth noting here that segmentation is required in order for any measurements to be collected. In the simplest case, a confocal dataset collected in a single fluorescent channel can be divided into two regions: fluorescent signal and background. Segmenting this channel using a fast, automated method is a matter of choosing a threshold intensity value that discriminates between these regions. However, the selection of a single threshold value that will apply across every optical slice is often thwarted by effects such as noise or z-drop. When such problems cannot be otherwise corrected in acquisition or by adjustment, manual segmentation methods are usually necessary. In this simple case, that may involve selecting a different threshold appropriate for each optical slice (to subvert conditions like z-drop) or manually editing the results of an automatic segmentation to remove artifacts.

An example of a segmentation editor is shown in Fig. 10.18. This routine is from the 3D software Amira (www.visageimaging.com) and offers a slice-by-slice view of the data from any perspective in addition to a selection of automated and manual selection tools. In the sample shown, a threshold has been set in the Display and Masking section to select all voxels (in the entire confocal dataset) with an intensity greater or equal to 64. Those voxels were then placed in a material named “myocardium” and outlined in green. This material data is saved in a separate label data file associated with the image data.

Once the nonfluorescent background has been isolated from the fluorescent signal, further refinements are possible. For the sample data shown in Fig. 10.18, it was decided to further segment the fluorescent signal into two histologically relevant regions: trabecular myocardium and compact myocardium. Obviously these regions cannot be isolated based on voxel intensity, so manual segmentation by an experienced observer is required. In Fig. 10.19, the “myocardium” material was renamed “compact myocardium,” and a second material “trabecular myocardium” has been defined and color-coded red. The background material “exterior” was locked (to prevent any changes), and a large paintbrush was used on each optical slice to select voxels from the green regions that belong in the red material. For 170 optical slices and a trained observer, this procedure took about 2 hours.

More sophisticated approaches to automatically segmenting confocal data have been reported (Lin et al. 2003; Losavio et al. 2008; Rodriguez et al. 2003;

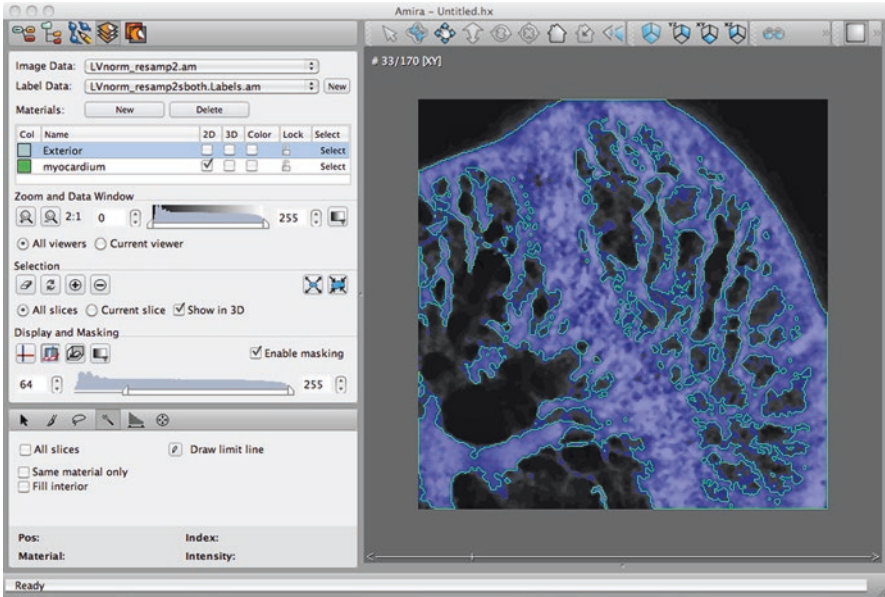


Fig. 10.18 Segmenting confocal data based on threshold. The segmentation editor in Amira (www.visageimaging.com) was used to locate voxels with intensity values between 64 and 255 (see the Display and Masking section on the right). The selected voxels are assigned to the material “myocardium” and outlined in green. Although only one slice is shown, the selection was completed on all 170 optical slices simultaneously

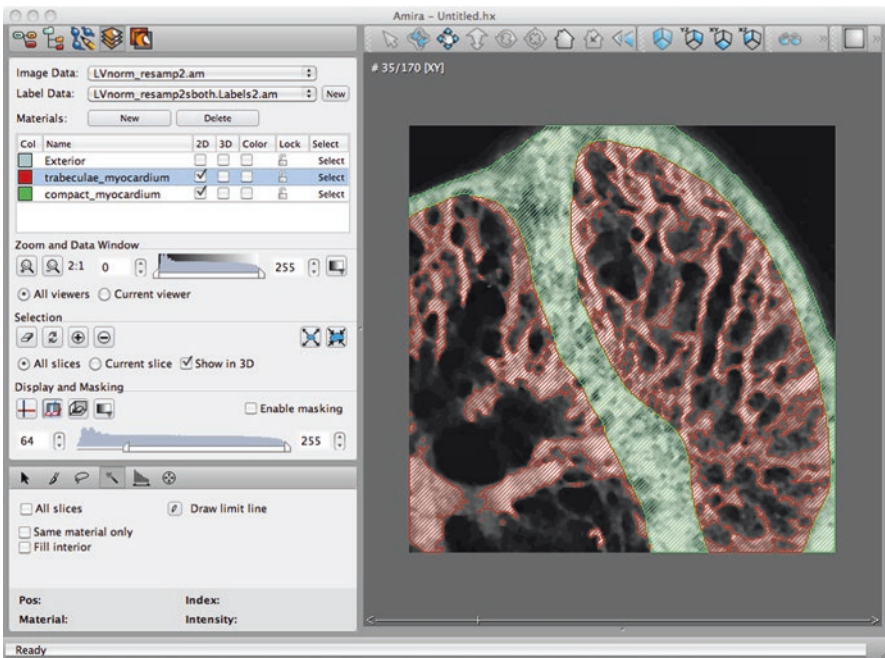


Fig. 10.19 Segmenting confocal data based on manual selection. The automatically segmented data in Fig. 10.18 was further separated into trabecular (red) and compact (green) myocardium on each optical slice by manual selection using the drawing tools

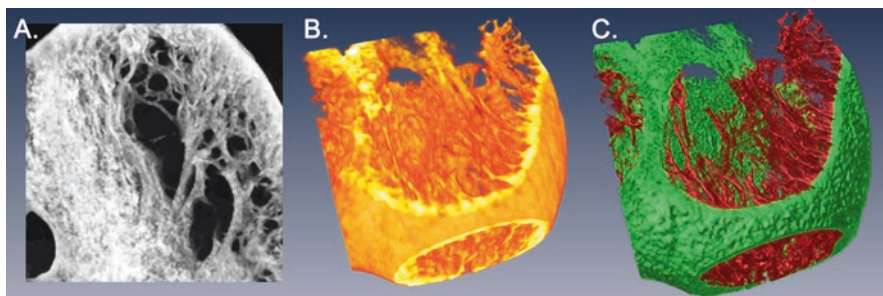


Fig. 10.20 Reconstruction models. (a) Projected data from the middle 50 slices (of 170). (b) Volume rendering of all 170 optical slices. (c) Volume rendering using manual segmentation to highlight trabecular (red) and compact myocardium (green)

Yi and Coppolino 2006), and some have even been made available in some specialized 3D software programs. These methods are usually based on finding the edges of regions based on voxel intensity, and some use filters based on the shape and size of the regions they expect to locate. As such, these routines require the image data to have the best contrast available (the data uses most of the available brightness spectrum), and the resolution should be optimal (and probably even deconvolved). While these routines can save time, the results will still contain errors and require some manual editing.

10.5.6 Modeling and Visualization

Most modern computers can now render 3D models practically in real time; thus visualized results are often rendered repeatedly until desired results are obtained. These computational speeds also make it easier to render animation frames, where the subject changes position or rotates in the field of view in a series of predetermined steps. Nearly all 3D software programs offer methods of generating animations in AVI or MPEG format or can generate a folder full of single frames suitable for import into digital movie applications such as Adobe Premiere.

The amount of information conveyed in a 3D visualization can be augmented with just a little extra work. Consider the data from the confocal dataset from the previous segmentation example now portrayed in Fig. 10.20. In (A), the middle 50 optical slices are shown as a maximum projection image generated immediately after acquisition. If all 170 optical slices had been included in the projection, the interior detail of the trabecular myocardium inside the ventricle would have been obscured (see Fig. 10.1c). In Fig. 10.20b, all of the data is visualized in a volume render using an RGBA lookup table which renders the background noise as transparent. The more intense voxels are rendered more opaque. The 3D nature of the sample is more apparent, and the trabecular myocardium can now almost be differentiated from the compact myocardium. In (C), the volume render has made use of the segmentation data to color code the two histological types of myocardium, and the discrimination of trabeculae inside the ventricle wall composed of compact

myocardium is clear. Note that (C) is not a surface reconstruction but a special volume render utilizing the segmentation data to color-code voxels and adding diffused lighting and specular effects.

While single images of 3D models are easy to publish, often it is of value to share a model which can be viewed from multiple angles. This has typically involved supplying a supplementary animation file with a guided tour of the object as directed by the filmmaker. It is now possible to include 3D objects entirely within the Adobe Acrobat PDF file format (www.adobe.com) such that the data can be interactively manipulated by anyone with the PDF file (Ruthensteiner and Hess 2008). Note that only surface reconstruction data (not volume renders) can be imported into PDF files as of the writing of this chapter.

10.5.7 Measurement

Measurements obtained from 3D datasets are based on the spatial dimensions of single voxels in the data. For example, if the size of a single voxel is $1\text{ mm} \times 1\text{ mm} \times 1\text{ mm}$, then the smallest volume that can be measured is a single voxel, or 1 mm^3 . To determine the volume of a fluorescent object in an image dataset, it is necessary to identify (segment) and then count the voxels in that object. The total volume will then be the voxel count multiplied by the volume of a single voxel.

The calculation of surface area will require a surface reconstruction. In this case, the individual triangles within the surface will each have unique areas determined by the x,y,z coordinates of the three vertices of each triangle. The planar areas of each triangle are calculated and summed to produce a total surface area. (One can appreciate why surface calculations proceed more slowly when there are millions of triangles.)

The segmentation data of Fig. 10.19 was used to make the surface reconstructions shown in Fig. 10.21. Figure 10.21a is a surface model of the voxels contained in both segments. For segmented data and surface reconstructions, it is possible to model the segments together (Fig. 10.21b) or independently (Fig. 10.21c–d). The volume and surface area measurements for the defined segments are shown in Table 10.1.

Most 3D software systems with measurement capabilities will further subdivide each material into regions based on voxel connectivity. That is, an isolated island of contiguous voxels is considered a region separate from other voxel regions. Material statistics can be gathered for materials based on regions, which are useful for counting cells, or nuclei, or any other objects that exist as isolated fluorescent entities.

Figure 10.22a shows a field of fluorescently stained nuclei that were segmented using an automatic threshold and then surface rendered (Fig. 10.22b). Material statistics generated on this segmentation yields the data in Table 10.2, where the nuclear material has been analyzed based on region (the table is abbreviated).

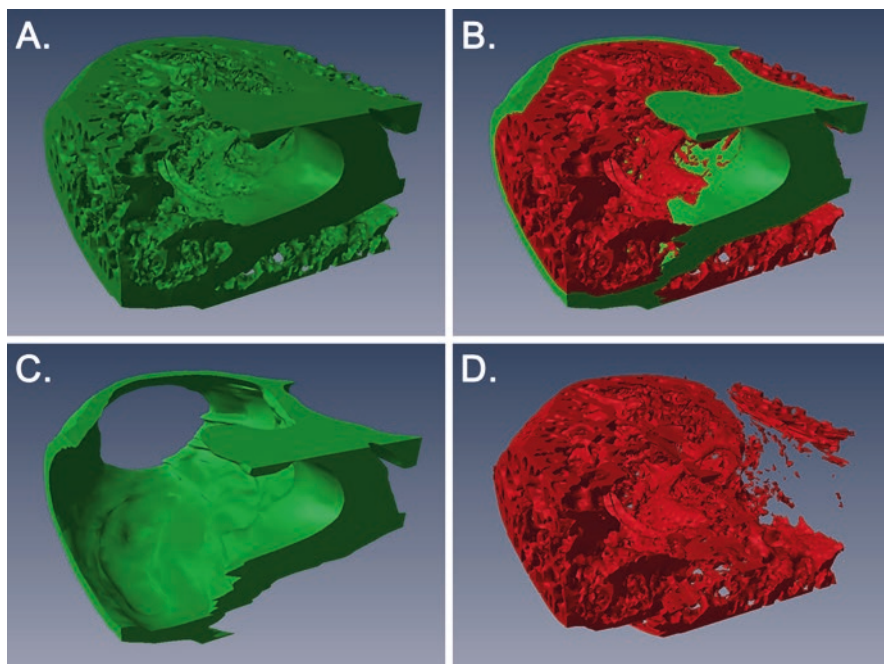


Fig. 10.21 Surface reconstructions from segmented confocal data. (a) Surface from threshold-based automatic segmentation. (b) Surface from data in (a) manually segmented into trabecular (red) and compact (green) myocardium. Segmented regions of interest allow separate rendering (c, d) and measurement

Table 10.1 Measurements from 3D reconstructed confocal data

Material	Voxel count	Volume (m ³)	Surface area (m ²)
Exterior	7,083,350	0.760	
Trabecular myocardium	1,838,729	0.197	21.4
Compact myocardium	2,219,041	0.238	7.3
Total	11,141,120	1.196	

Voxel size = $5.08 \times 5.08 \times 4.14 \mu\text{m}$

Total dataset is $256 \times 256 \times 170$ voxels

The analysis provides a total count of the regions revealing that 139 isolated regions were identified. The volume; voxel count; x,y,z center coordinates; and mean voxel intensity within each region are provided. The data can be exported for more complex analyses. Note that this data from Amira is sorted by volume. It is a task left to the experimenter to decide if the smallest regions (nuclei with volumes of $1 \mu\text{m}^3$) are noise

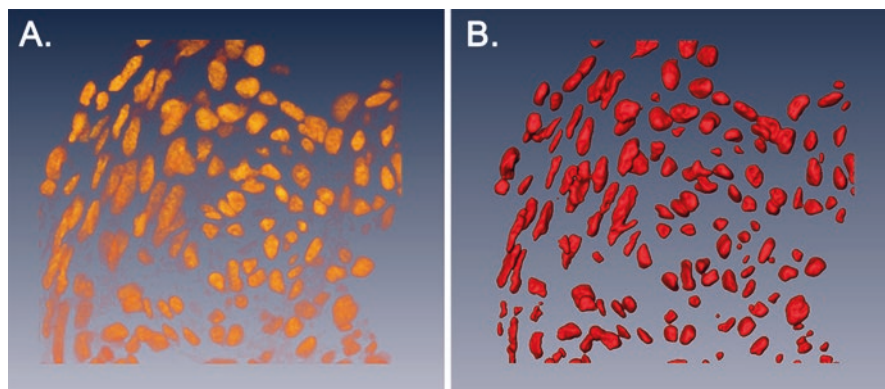


Fig. 10.22 Counting nuclei in segmented confocal data. (a) Voxel rendering of a field of stained nuclei. (b) The nuclei were segmented using an automated threshold for a surface reconstruction. Material statistics for the regions in this segmentation report volume, spatial location, and fluorescence intensity data for each nucleus (see Table 10.2)

Table 10.2 Regional measurements

Nr	Region	Voxel count	Volume	CenterX	CenterY	CenterZ	Mean
1	Exterior 1	2,196,590	116823.1	93.6	119.4	5.3	13.0
2	Nuclei 139	13,248	704.5	44.7	133.3	6.2	78.6
3	Nuclei 138	13,229	703.5	122.8	96.4	5.5	81.3
4	Nuclei 137	9868	524.8	104.0	111.3	5.6	87.6
5	Nuclei 136	9134	485.7	93.1	92.6	6.2	89.2
6	Nuclei 135	8511	452.6	62.9	114.3	5.5	71.9
7	Nuclei 134	8500	452.0	80.5	105.9	5.1	79.1
8	Nuclei 133	8015	426.2	72.0	83.7	5.7	82.0
9	Nuclei 132	7334	390.0	78.5	92.4	5.6	84.9
10	Nuclei 131	7048	374.8	77.6	123.0	5.6	73.6
...
138	Nuclei 3	19	1.0	41.4	149.6	6.83	59
139	Nuclei 2	19	1.0	93.7	72.6	2.9	57
140	Nuclei 1	19	1.0	82.3	91.4	8.8	62

The measurement routines in most 3D software systems will provide the basic data described here. Some provide more sophisticated tools, but these systems tend to be more expensive and will have highly specialized functions of little use to everyone. Thus, the last advice this chapter can offer is that the user should have some knowledge of the type of quantitative measures needed and to carefully select software that will fulfill the research requirements.

Literature Cited

- Biggs DS (2010) 3D deconvolution microscopy. *Curr Protoc Cytom.* Chapter 12: p. Unit 12 19 1–20
- Biggs DSC (2004) Clearing up deconvolution. *Biophoton Int* (February):32–37
- Carlsson K, Aslund N (1987) Confocal imaging for 3-D digital microscopy. *Appl Opt* 26(16):3232–3238
- Clendenon JL, Byars JM, Hyink DP (2006) Image processing software for 3D light microscopy. *Nephron Exp Nephrol* 103(2):e50–e54
- Feng D, Marshburn D, Jen D, Weinberg RJ, Taylor RM II, Burette A (2007) Stepping into the third dimension. *J Neurosci* 27(47):12757–12760
- Guan YQ, Cai YY, Zhang X, Lee YT, Opas M (2008) Adaptive correction technique for 3D reconstruction of fluorescence microscopy images. *Microsc Res Tech* 71(2):146–157
- Hecksher-Sorensen J, Sharpe J (2001) 3D confocal reconstruction of gene expression in mouse. *Mech Dev* 100(1):59–63
- Lee SC, Bajcsy P (2006) Intensity correction of fluorescent confocal laser scanning microscope images by mean-weight filtering. *J Microsc* 221(Pt 2):122–136
- Lin G, Adiga U, Olson K, Guzowski JF, Barnes CA, Roysam B (2003) A hybrid 3D watershed algorithm incorporating gradient cues and object models for automatic segmentation of nuclei in confocal image stacks. *Cytometry A* 56(1):23–36
- Losavio BE, Liang Y, Santamaría-Pang A, Kakadiaris IA, Colbert CM, Saggau P (2008) Live neuron morphology automatically reconstructed from multiphoton and confocal imaging data. *J Neurophysiol* 100(4):2422–2429
- Mackenzie JM, Burke MG, Carvalho T, Eades A (2006) Ethics and digital imaging. *Microscopy Today* 14(1):40–41
- McNally JG, Karpova TS, Cooper JA, Conchello J-A (1999) Three-dimensional imaging by deconvolution microscopy. *Methods* 19(3):373–385
- Rodriguez A, Ehlenberger D, Kelliher K, Einstein M, Henderson SC, Morrison JH, Hof PR, Wearne SL (2003) Automated reconstruction of three-dimensional neuronal morphology from laser scanning microscopy images. *Methods* 30(1):94–105
- Rueden CT, Eliceiri KW (2007) Visualization approaches for multidimensional biological image data. *BioTechniques* 43(1 Suppl):31 33-6
- Ruthensteiner B, Hess M (2008) Embedding 3D models of biological specimens in PDF publications. *Microsc Res Tech* 71(11):778–786
- Savio-Galimberti E, Frank J, Inoue M, Goldhaber JI, Cannell MB, Bridge JHB, Sachse FB (2008) Novel features of the rabbit transverse tubular system revealed by quantitative analysis of three-dimensional reconstructions from confocal images. *Biophys J* 95(4):2053–2062
- Soufan AT, van den Berg G, Moerland PD, Massink MMG, van den Hoff MJB, Moorman AFM, Ruijter JM (2007) Three-dimensional measurement and visualization of morphogenesis applied to cardiac embryology. *J Microsc* 225(Pt 3):269–274
- Sun Y, Rajwa B, Robinson JP (2004) Adaptive image-processing technique and effective visualization of confocal microscopy images. *Microsc Res Tech* 64(2):156–163
- Yi Q, Coppolino MG (2006) Automated classification and quantification of F-actin-containing ruffles in confocal micrographs. *BioTechniques* 40(6):745–746, 748, 750 passim