# 3D-Data Representation with ImageJ

Kai Uwe Barthel

International Media and Computing, FHTW (University of Applied Sciences) Berlin, Germany

*Abstract* — **After having developed three ImageJ plugins that use 3D display techniques of image data (*Color Inspector 3D, 3D Surface Plot, and Volume Viewer*), the idea came up that these plugins do share a lot of common algorithms that could also be used for other plugins which need a 3D display of (image) data.**

**The workshop concerning 3D-Data Representation with ImageJ that will be given at the "ImageJ User and Developer Conference" will consist of two parts. In the first part the usage of the plugins mentioned above using 3D-data display techniques will be described. Typical applications and image analysis features will be explained.**

**The second part of the workshop will explain how the 3D programming techniques shared by these plugins can be used by other plugins. At the time of the ImageJ conference there will be an "easy to use" JAVA framework, which will allow an uncomplicated construction of plugins using 3D display techniques.**

## I. Motivation

The main focus of ImageJ is the analysis, the measurement and the modification of (2D) images. ImageJ can also handle multiple spatially or temporally related images. These image sets are called stacks. A stack is displayed in a window with a scroll bar that provides the ability to move through the slices that make up the stack.

There are many other kinds of 2D and 3D representations of scientific data that do require appropriate visualization modes. Typical examples of such data types are volume data, confocal image data, elevation data, 3D histograms, and point data like XYZ triples. Since ImageJ has only few provisions for other display modes, several plugins (like "*VolumeJ*" from Michael Abràmoff [1] or "*OrtView*" from Paola Bonetto [2]) were developed to enable different display modes. The author has contributed three ImageJ plugins using an interactive display of 3D-data. The "*Color Inspector 3D*" plugin [3] can display 3D distributions of image pixel colors in different color spaces. "*3D Surface Plot*" [4] creates interactive surface plots, and the "*Volume Viewer*" plugin [5] can be used for stacks in order to generate volume renderings or views of arbitrarily oriented slices.

Common to all mentioned plugins is the problem of reusability of the particular programming techniques, which cannot easily be adapted for other plugins. This is due to the special design and optimization of each plugin for its special task and the particular kind of data to be displayed.

During the development of these three plugins using 3D display techniques of image data, it became obvious that they share common algorithms which could also be used by other plugins with a need for 3D display of (image) data. This paper will describe typical 3D visualization modes of the proposed 3D ImageJ framework.

The next section will briefly describe the developed plugins. Afterwards the basic idea for constructing plugins that allow an interactive display of 3D data will be explained.

## II. Realized 3D Plugins

### A. Color Inspector 3D

The Color Inspector 3D plugin displays the distribution of the colors of an image within a 3D color space (figure 1). The plugin supports the RGB, YUV, YCbCr, HSB, HSV, HSL, HMMD, Lab, Luv, xyY, and XYZ color spaces. By switching from one color space to another the relationship between different color spaces can be made visible (figure 2). In addition, the effect of typical image manipulations such as contrast, brightness, saturation change, and color rotation can be studied on the image and the corresponding color distribution. One major area of application for this plugin is the teaching of color science. In addition this plugin can also be used for color analysis and color segmentation.

The plugin supports five display modes for displaying color distributions: Colors can be shown independent or weighted
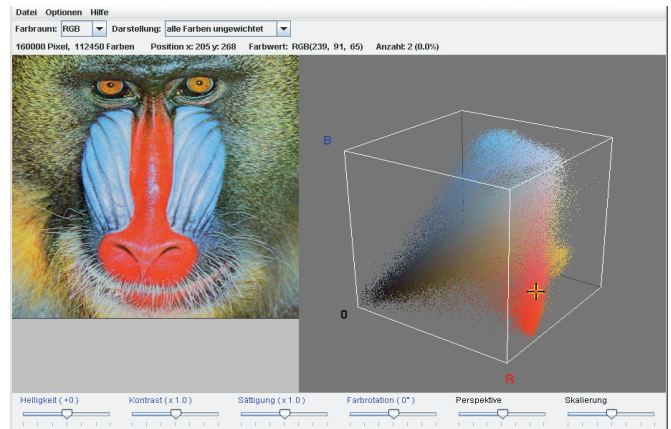


**Figure 1:** Color Inspector 3D, on the left the image to be analysed, on the right the 3D color distribution in the RGB color space
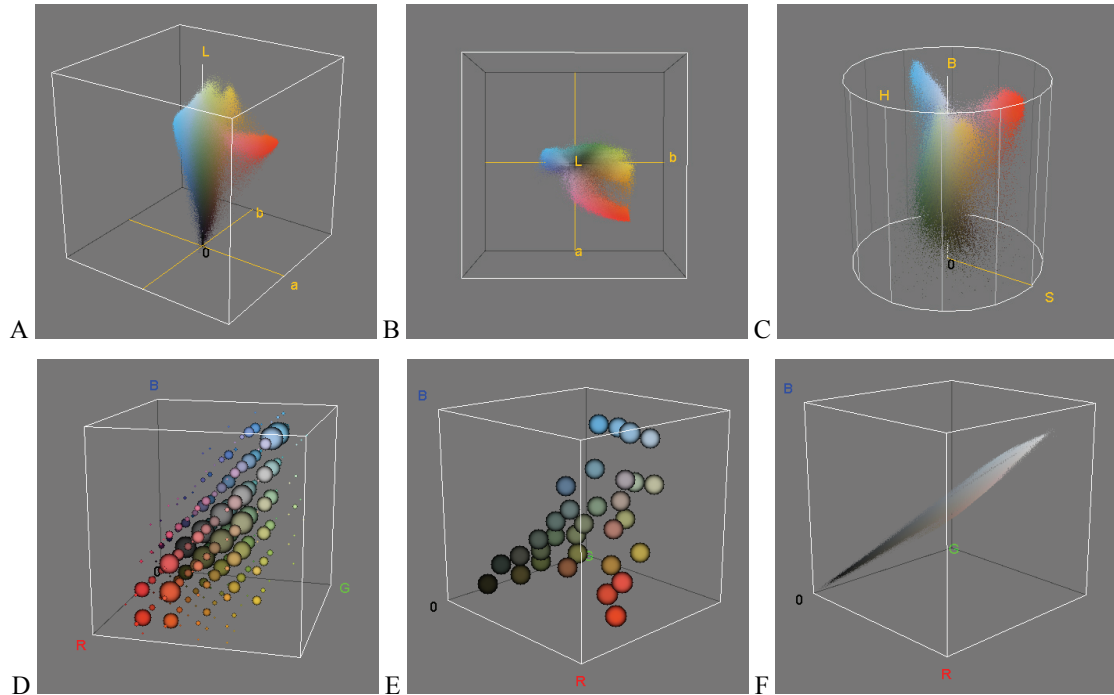
**Figure 2:** Different visualization modes of the Color Inspector 3D plugin: A) Lab color distribution; B) Lab color distribution (top view); C) HSB color space; D) RGB-3D color histogram; E) Color reduction to 32 colors; F) Image with reduced saturation

by their frequency. This plugin allows the generation of 3D color histograms by partitioning the color space into equally spaced cubes (color cells). Each cell is represented by a colored sphere with a volume proportional to its frequency. In addition two color reduction schemes can be applied. Color values and their absolute and relative frequencies are listed.

The 3D display of the color distribution can be rotated and translated. Scaling and perspective parameters are adjustable. The 3D visualization, image modifications and color segmentation results can be saved as new images.

## B. 3D Surface Plot

This plugin creates interactive surface plots from all kinds of image types. The luminance of each pixel in the image is interpreted as the height for the plot. Selections, also non-rectangular, are supported. The plot can be displayed in different color schemes: original colors, grayscale, and different LUT schemes. Scale, rotation, perspective and position can be adapted. Using a smoothing option reduces noise. An adjustment of the lightning condition improves the visibility of small differences. The surface plot supports four different drawing modes: dots, lines, meshes or a filled surface.
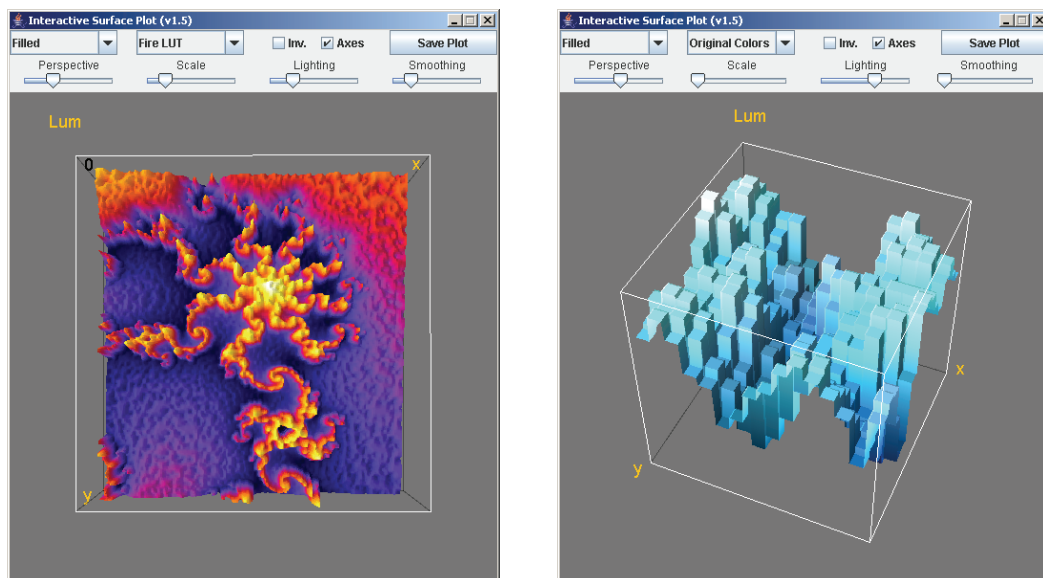


**Figure 3:** Examples of different visualization modes of the "3D Surface Plot" plugin
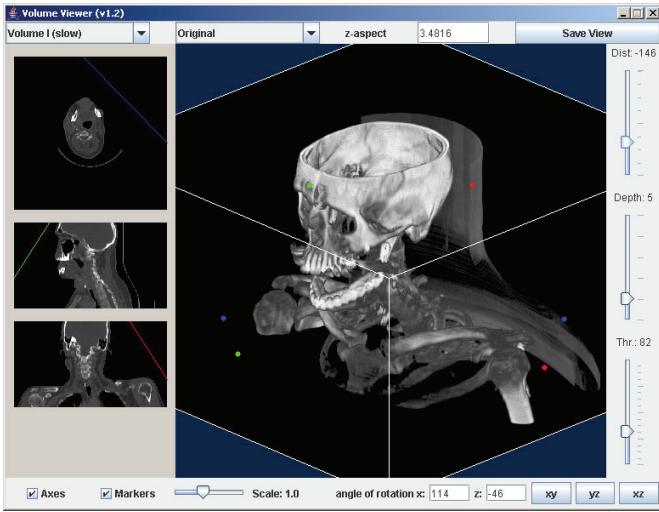
**Figure 4:** Volume Viewer plugin: Rendering of a series of 83 CT images downloaded from homepage.mac.com/rossetantoine/osirix/Index2.html.

### C. Volume Viewer

The volume viewer plugin [5] shows stacks as volume visualizations within a 3D XYZ-space. This plugin can be used with 8, 16 and 32 bit stacks. Six display modes allow displaying slices and volume visualization using different interpolation and rendering techniques. If brightness or contrast settings of the original stack are adjusted, the data is reread and redisplayed. The viewing position and the orientation and position of slices or volumes can be arbitrarily chosen.

### III. 3D-GRAPHICS VS. VISUALIZATION OF 3D-DATA

Modern programmable computer graphic hardware supports sophisticated 3D display features. For several reasons however, these techniques cannot easily be used for a platform independent display of 3D data. One problem is the large variety of different graphic hardware and their dependency of particular drivers and programming requirements which cannot be handled easily with a pure standard Java programming approach.

The next problem results from the difference between 3D objects typically approximated by 3D primitives and 3D measurement data.

The normal process of creating a 3D computer graphics scene can be sequentially divided into three basic phases: modeling, layout of the scene, and rendering. Typically 3D objects are approximated by polygonal meshes onto which texture images are mapped. In a final step the resulting 2D view of this 3D scene is rendered taking into account the viewing position, projection parameters, lightning conditions, etc.

For the purpose of a 3D visualization of three dimensional (measurement) data this process is not appropriate. Instead of having real 3D objects that can be described by polygonal meshes or other approximation forms, the measured 3D data typically consists of individual voxels, pixels or xyz-triples.

Java 3D is a scene graph-based 3D application programming interface (API) for the Java platform, that runs on top of either OpenGL or Direct3D. A scene is constructed using a scene graph that is a representation of the objects that have to be shown. Again there is no easy way to use Java 3D for the visualization of 3D measurement data.

The proposed programming framework *JRenderer3D* for displaying 3D data in ImageJ does not use Java 3D. It is only based on the Java standard edition J2SE.

### IV. BASIC VISUALIZATION ELEMENTS

The analysis of the different display types of the plugins described in the previous sections confirmed that all visualization modes can be generated with a common rendering technique. Only a few typical very simple basic 3D elements are sufficient to manage a large variety of different visualization modes.

Table 1 lists basic 3D elements, their required data (like coordinates or intensities), their optional data (like a color lookup table), and the visualization modes that can be generated using the proposed programming *JRenderer3D* framework.

**Table 1:** Basic 3D elements and their JRenderer3D visualization modes

| 3D elements | required data | optional, additional data | JRenderer3D visualization modes |
|---|---|---|---|
| xyz-triples | x, y, z | size, color | **points / dots spheres** |
| lines | $x_1, y_1, z_1$ $x_2, y_2, z_2$ | attributes like color and line width | **3D lines** |
| text | x, y, z, (text position) text string | attributes like color, font and size | **2D text** positioned in a 3D space |
| images (2D data) | x*, y*, pixel data | color LUTs, lighting conditions | **surface plots** points / dots lines, meshes filled surfaces |
| stacks (3D volume data) | x*,y*,z*, intensity/ alpha | z-aspect-ratio, color LUTs | **volume renderings slices projections** |

*) this data can be omitted if the storage is organized in arrays with consecutive data elements

General visualization parameters for the 3D visualization are the 3D transform parameters (describing scale, orientation, and perspective), the size of the drawing canvas, and the background color.
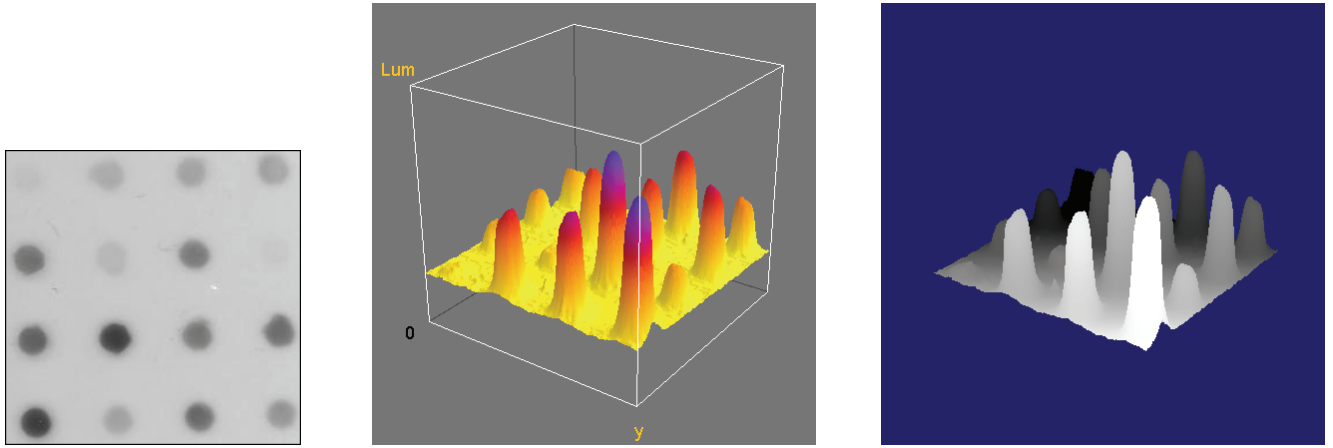
**Figure 5:** Example image, surface plot (inverted height), and corresponding Z-buffer image

## V. 3D VISUALIZATION TECHNIQUE

This section describes the basic visualization process of the proposed 3D rendering scheme *JRenderer3D* (see figure 6). The 3D visualization consists of the following steps. In a first step the elements to be visualized have to be acquired. Possible 3D elements are listed in the first column of table 1. This list of 3D data elements is passed to the *JRenderer3D*. For each 3D element the desired visualization mode is passed as well.

The *JRenderer3D* will perform the rendering of the resulting 2D image of the 3D "scene" that is described by the list of 3D elements. All elements are rendered according to their desired visualization mode.

The coordinates of each 3D element from the render list are transformed and drawn into the 2D image to be rendered. Taking into account the general visualization parameters and the actual transform parameters that determine the viewing angle, scale, perspective, and lighting condition the resulting 2D projection is calculated.

The actual rendering done by *JRenderer3D* is using the Z-buffer algorithm. The Z-buffer algorithm is an easy to implement algorithm for rendering images properly according to depth. A Z-buffer image containing the closest depth at each pixel location is created parallel to the buffer for the rendered image. Each location in this depth buffer is initialized to a large negative value. Now for each point that is to be rendered, the depth of this point is checked against the depth of the point at the desired pixel location. If the point depth is greater than the depth at the current pixel, the pixel is colored with the new color and the depth buffer is updated. Otherwise, the point is behind another object and therefore will not be rendered. Figure 5 shows an example of a surface plot and the according Z-buffer image.

Detailed methods how to generate 3D data representation with ImageJ will be given at the workshop of the ImageJ conference. Source code and the API documentation will be available at the ImageJ website.

### REFERENCES

[1] M. D. Abràmoff, *VolumeJ - Volume Rendering in Java* bij.isi.uu.nl/vr.htm, 2003
[2] P. Bonetto, *OrtView.* 2002, www.disi.unige.it/person/BerteroM/MedNuc/OrtView/OrtView.html
[3] K. U. Barthel, *"Color Inspector 3D".* 2004. http://rsb.info.nih.gov/ij/plugins/color-inspector.html
[4] K. U. Barthel, *"3D Surface Plot".* 2004. http://rsb.info.nih.gov/ij/plugins/surface-plot-3d.html
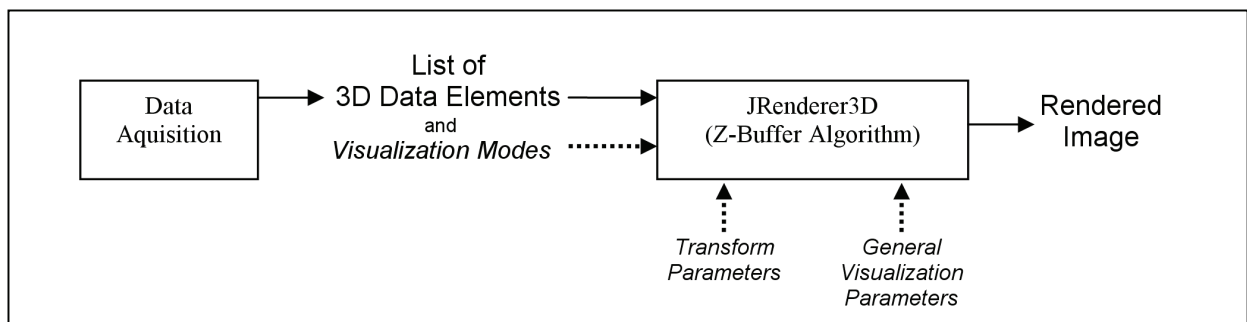[5] K. U. Barthel, "Volume Viewer*".* 2005. http://rsb.info.nih.gov/ij/plugins/volume-viewer.html

**Figure 6:** Visualization process of the proposed 3D rendering scheme