A Simple Implementation of a 3D Data Cube Viewer

Honglin Ye

National Radio Astronomy Observatory

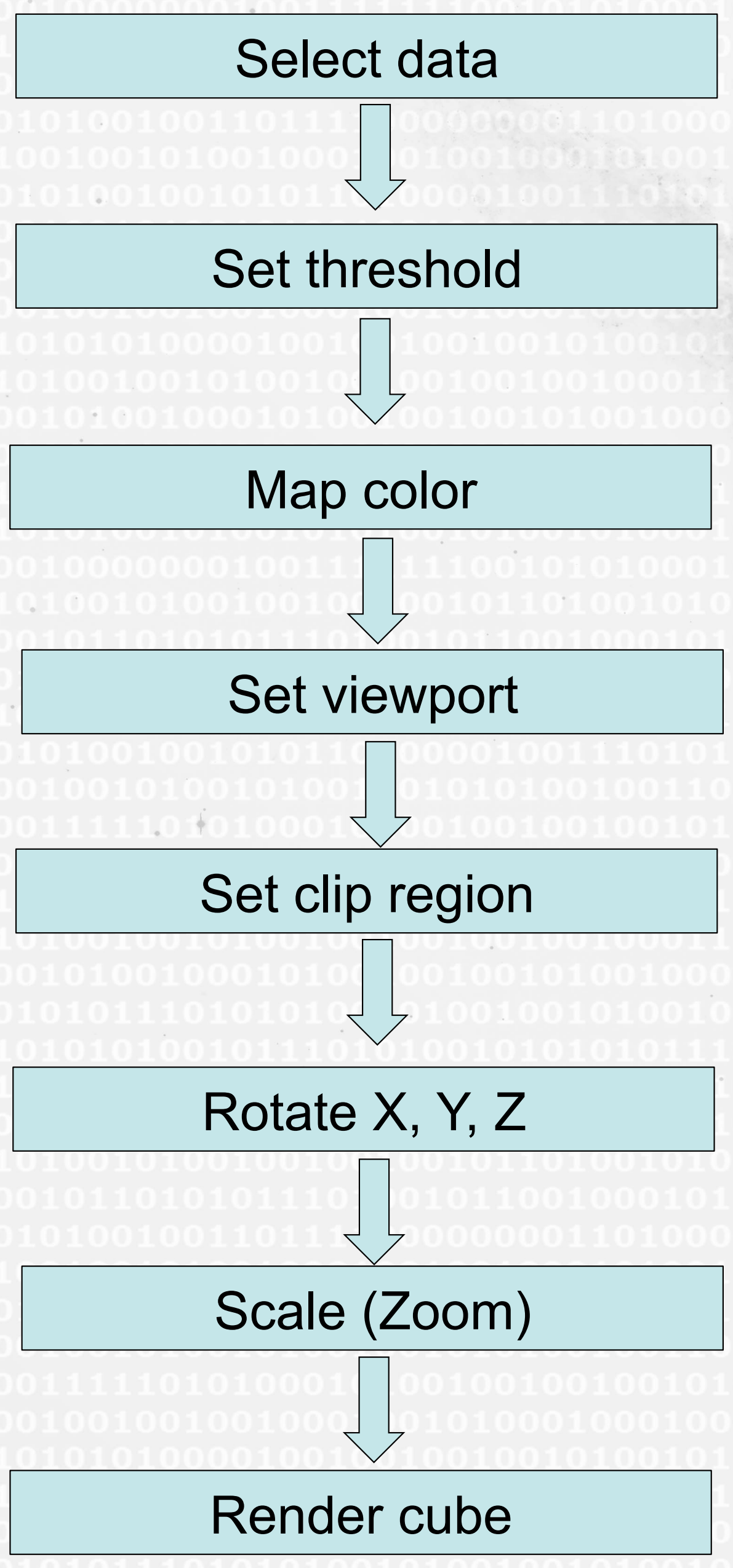
3D View Capability Needed

One of the most important end products from a spectral line observation is the brightness $\mathcal{B}$ as a function of right ascension α, declination δ and frequency ω. The $\mathcal{B}(\alpha, \delta, \omega)$ is commonly referred to as a data cube or an image cube. In practice, brightness is obtained, for each frequency channel separately, as images. While rendering a stack of such images in various ways can illuminate the important aspects of a data cube, the process of finding relations between channels can be difficult and time consuming. New instruments, such as ALMA and EVLA, which can observe over a broad wavelength range and up to thousands of channels, will present enormous challenges for the visualization and analysis of large numbers of paged images. It is often desirable to render the data cube as a whole to examine from arbitrary viewing angle and for any clip region of interest. To meet some of these needs in CASA, we developed a GUI based on the Qt OpenGL module for viewing 3D data.

Qt OpenGL Module

OpenGL (Open Graphics Library) is a standard specification defining a cross-language, cross-platform API for 3D graphics. OpenGL provides a complete feature set for graphics operations in a pipelined hardware accelerated architecture for triangle and polygon rendering. OpenGL provides little support for GUI programming issues. The Qt OpenGL module is a platform-independent Qt/C++ wrapper that opens an OpenGL display buffer where one can use the OpenGL API to render the contents. Qt OpenGL module is easy to use in Qt/C++ applications.

State Machine


Select data
Set threshold
Map color
Set viewport
Set clip region
Rotate X, Y, Z
Scale (Zoom)
Render cube

This application uses right hand display coordinate system (X, Y, Z), where X is horizontal pointing to right. Without rotation, the world coordinates (α, δ, ω) of a cube are parallel to (X, Y, Z). A cut-off brightness value can be selected such that pixels with lower $\mathcal{B}$ become transparent. The pixels of $\mathcal{B}$ values above the threshold are assigned to colored vertices. The cube is to rotate about its center

$$((\alpha-\alpha_0)/2,(\delta-\delta_0)/2,(\omega-\omega_0)/2))$$

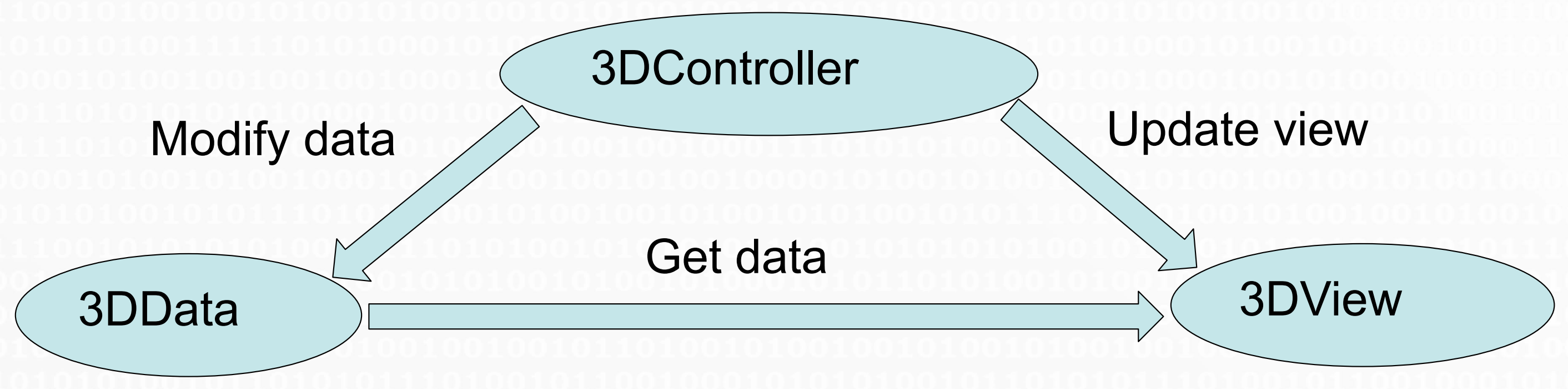
and the radius of the bounding sphere is

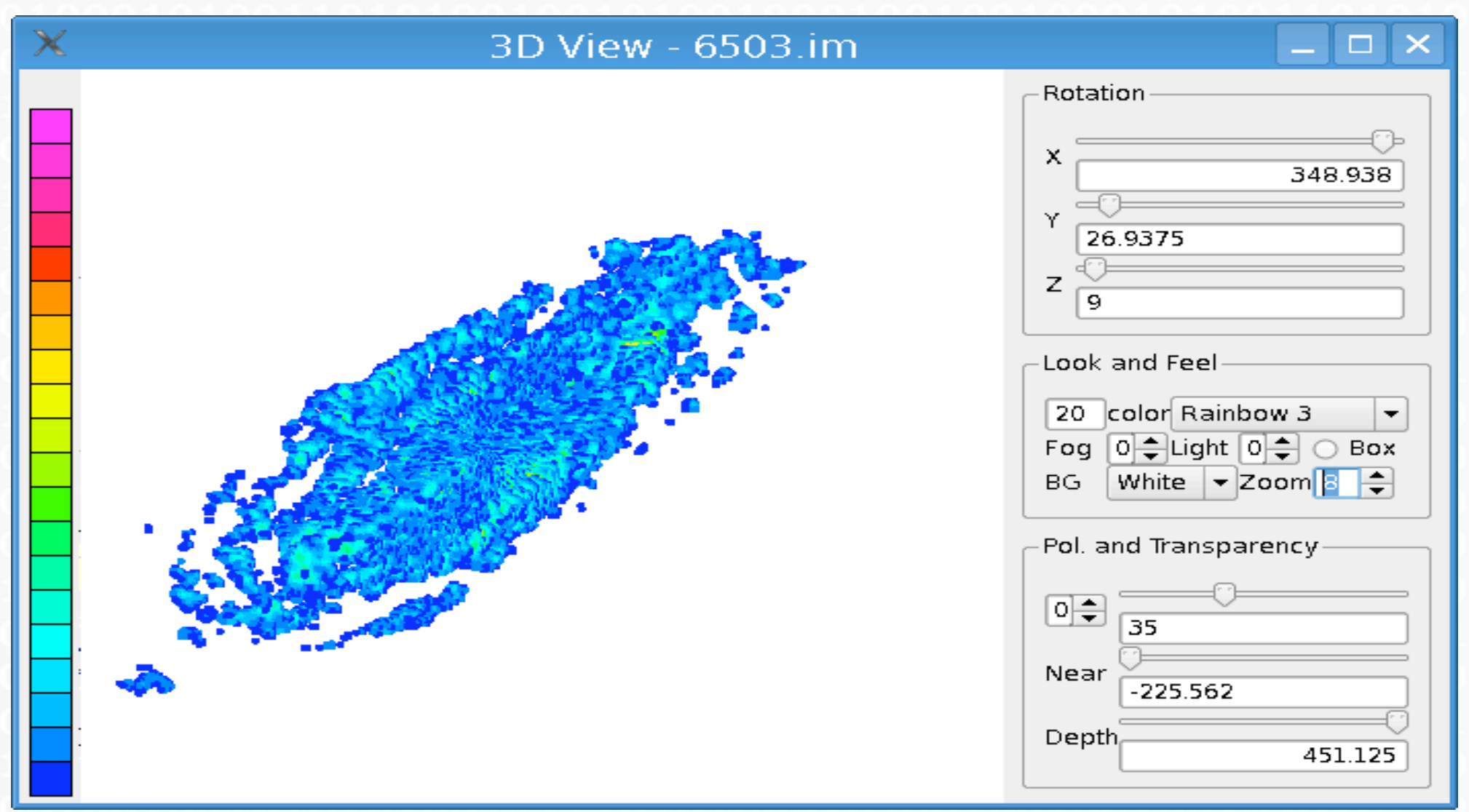
$$r=\sqrt{(\alpha-\alpha_0)^2+(\delta-\delta_0)^2+(\omega-\omega_0)^2}\,/2$$

The viewing volume is created with orthographic projection of the shape of a cube and its bounding planes are at ±r. The region of interest can be further confined by 2 clipping planes adjustable for Z=near and Z=near+depth. The cube is then rotated in the order of X, Y and Z and scaled. Finally the cube is rendered on to the screen and the controller is reset to listen to the next user request.

MVC Architecture

This application consists of 3 major C++ classes, 3DData, 3DView, and 3DController. Each class implements the functions of a component in the MVC (Model, View, and Controller) design pattern. 3DData selects data and builds the image cube. It is purely independent on 3DController and 3DView modules. 3DView component renders the OpenGL visual onto screen. It executes 3DController commands to update display contents. 3DController first takes user inputs from keyboard, mouse and control buttons, then updates Model's states, and finally notifies 3DView to render the cube.

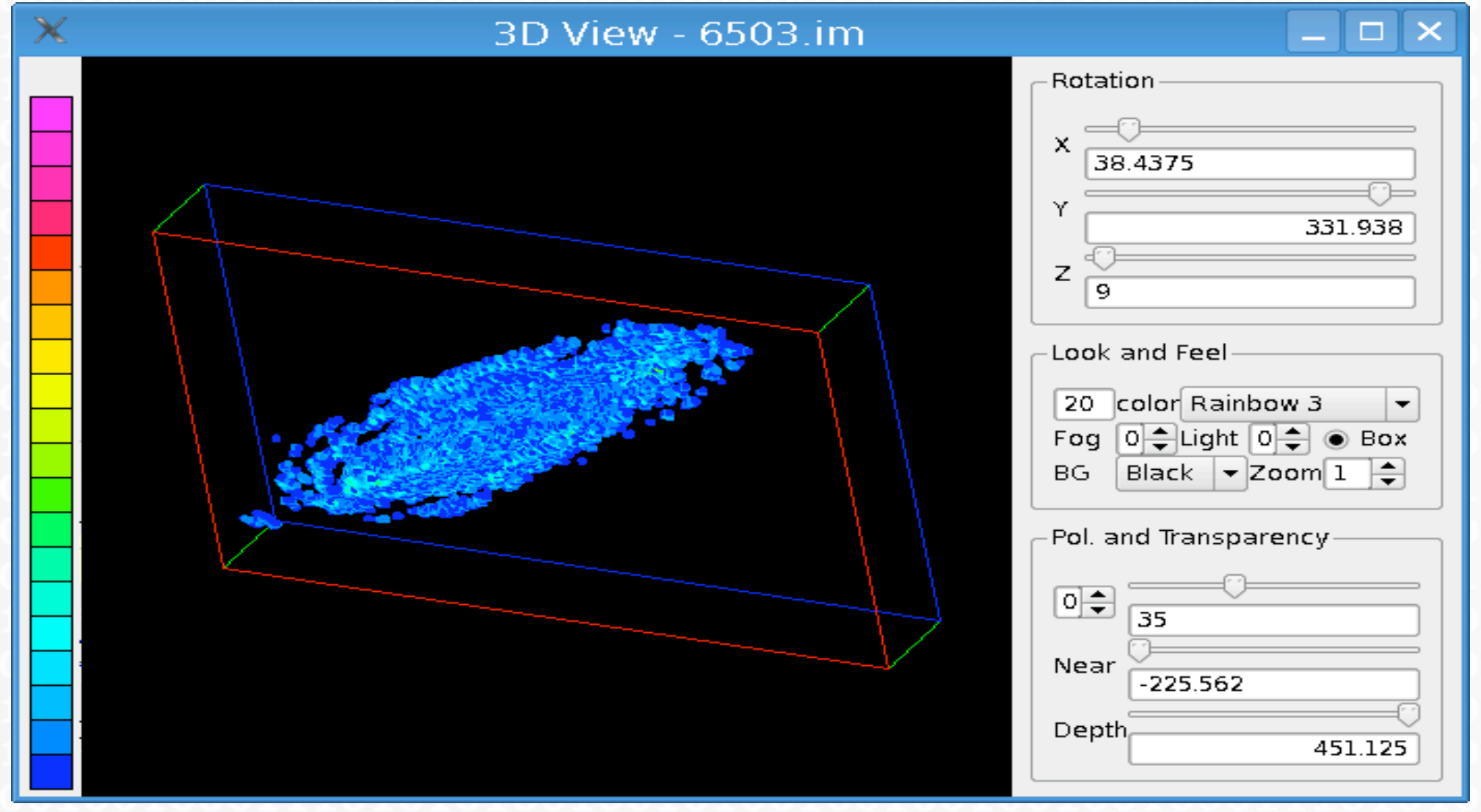

3DController
Modify data
Update view
Get data
3DData
3DView



Rotation
X 348.938
Y 26.9375
Z 9

Look and Feel
20 color Rainbow 3
Fog 0 Light 0 ○ Box
BG White Zoom 8

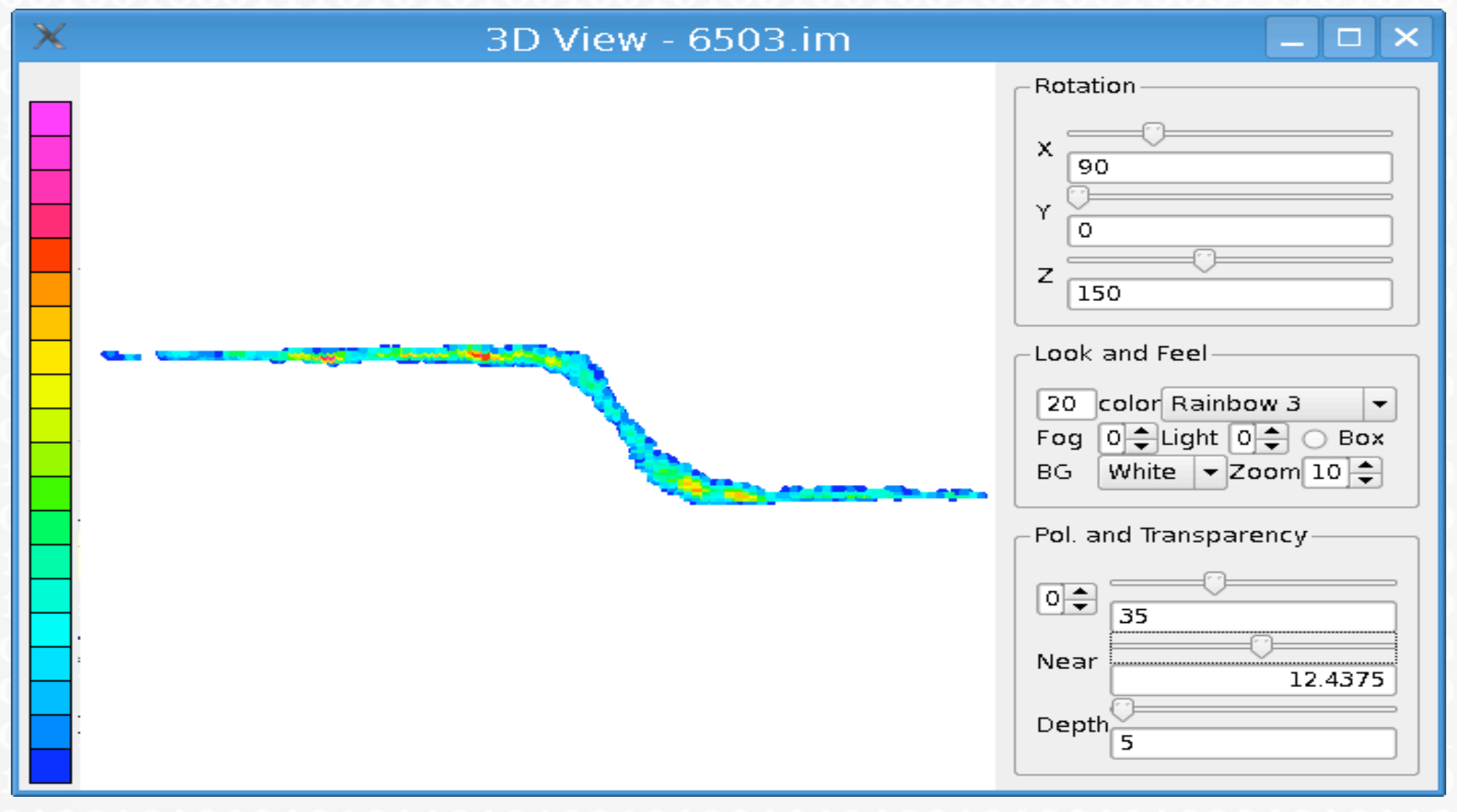
Pol. and Transparency
0 35
Near -225.562
Depth 451.125



Rotation
X 38.4375
Y 331.938
Z 9

Look and Feel
20 color Rainbow 3
Fog 0 Light 0 ● Box
BG Black Zoom 1

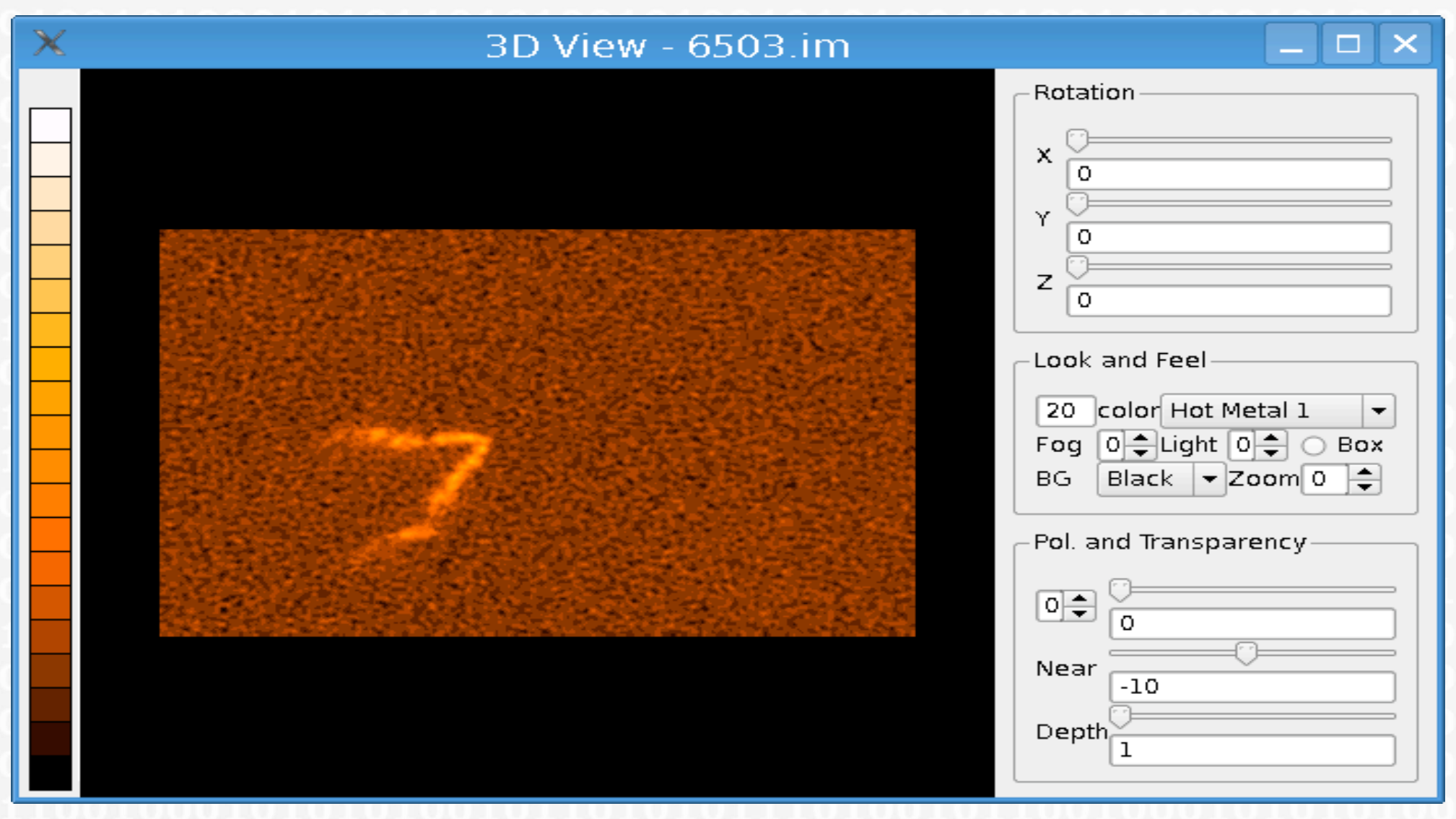
Pol. and Transparency
0 35
Near -225.562
Depth 451.125



Rotation
X 90
Y 0
Z 150

Look and Feel
20 color Rainbow 3
Fog 0 Light 0 ○ Box
BG White Zoom 10

Pol. and Transparency
0 35
Near 12.4375
Depth 5



Rotation
X 0
Y 0
Z 0

Look and Feel
20 color Hot Metal 1
Fog 0 Light 0 ○ Box
BG Black Zoom 0

Pol. and Transparency
0 0
Near -10
Depth 1

Example: NGC 6503

NGC 6503 is a normal spiral galaxy, type Scd. The cube used in this example is observed using NRAO's VLA and processed using AIPS package. The 3D views of the HI distribution over the whole disk show its spiral arms. 3D view does not calculate the statistic moments of the HI distribution. It shows the "surface" luminosity directly. By adjusting the threshold and cube orientation, one can quickly examine detailed features of the spiral. One can adjust rotation angles from keyboard arrow keys, from the widget's sliding bar or edit box, and by mouse dragging over the displayed object.

The inner structure of the spiral can be revealed by setting the positions of the "near" and "far" clipping planes to view only a slice of the big structure. Each thin slice, e.g., depth=5, can be inspected by adjusting the "near" plane position to move the slice backward or forward. The figure on the left shows the HI distribution in a slice that is parallel to the major axis of NGC 6503 (its position angle is about 30°).

If the 3 rotation angles are zero, the cube's orientation is the same as that in a usual 2D viewer, thus 3D viewer acts as 2D viewer.

Acknowledgement

I thank Kumar Golap for his comments and suggestions about the 3D viewer. I greatly benefited from reading 2009, AJ 137, pp. 4718-4733, by Eric Greisen, Kristine Spekkens, and Gustaaf van Moorsel.

Atacama Large Millimeter/Submillimeter Array