OpenGL 5DVisualizer Tutorial

Bugnon, Leandro A.
l.bugnon@gmail.com
Programación Avanzada - Universidad Nacional de Entre Ríos
Oro Verde,Argentina

April 12, 2011

Objetives.

This ImageJ plugin was developed intending to achieve two main objetives:

- On one hand, its an effort to simplify five dimensional (5D) image data visualization, taking advantage of two ray projection methods that allow an intuitive (and low time consumption) rendering, providing a way to group dimensional variables and simplify showing information process. This image types are common material in microscopy and biomedical imaging, where this plugin may be used.
- On the other hand, this software was built thinking it as a didactic application example, aplying some usefull OpenGL features through the Java Bindings for OpenGL (JOGL) library.

OpenGL library is used to cut down image rendering time, relegating most work to hardware acceleration device. For this reason, adequate graphic card and RAM memory are mandatory to get acceptable results. The benefit from using OpenGL is the parallelized GPU processing, which bring a great speed up in comparison with software based algorithms.

The algorithms are briefly described in the following sections. Both of them are built with the ray projection method, which means that each projected pixel in the screen is function of every voxel value on a row in line of sight through volume dataset. This principle allows to reduce dimensionality, obteining a representative 2D image from a dataset which may contains spacial, time and multispectral information.

Maximum Intensity Projection (MIP).

The highest luminance pixel of each projected ray is brought to the final image, resulting similar to a radiographic image. This method is very effective to visualize high contrast structures embedded in a low bright environment. As each color channel is processed in a diferent pipeline, its posible not only project luminance (gray scale) images, but also RGB ones, gathering independent channels with different colors.

Direct Volume Rendering (DVR)

Here the algorithm can be seen like a physical light-ray simulation, assuming that each voxel have transparency, opacity and reflectivity properties and the light is emmited from the user. The transparency value weighs one type of voxels (more opacity) from others (less opacity), letting the user define structures of interest, occluding unnecessary information and revealing hidden one. This algorithm is useful to set up different structures and discriminate their connections.

Voxel Classification

A simple thresholding classifier was implemented to assist in the visualization task, exemplified at Figure 1. The classification method is histogram based, similar to a lookup table (LUT); the user can define one dimensional trapezoidal transfer functions (TFs) by giving lower and upper threshold, slopes and color index. Each TF is defined for only one channel, so any channel can be classified independently. The output of TFs is defined according the next modes:

- *Original gray scale*: The TFs output is the voxel opacity. The color index is the same as the original image, so the voxel may be visible or not deppending the sum of TFs affecting it, but conserves the original shades.
- *RGB*: Both opacity and color are defined by the sum of TFs. Observe in the figure that if slopes aren't too sharp , two TFs may overlap in some point and the effects of each TF are added in a certain percentage according the slope, making parameter selection less rigid and also taking into account voxels that cant be grouped in only one class.

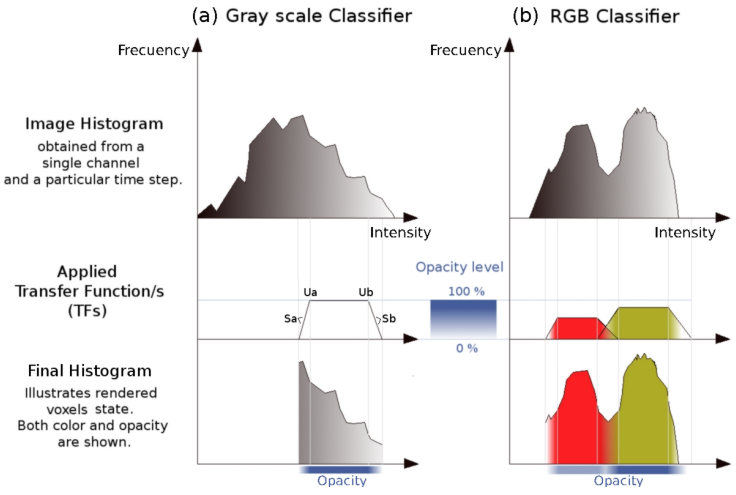

Figure 1: Transfer Function illustration
Aca podria faltar un pie de foto decente

Software features and control

This plugin accept 8/16-bit images (gray scale) formatted as usual stack configurations (XYZCT, XYCZT, XYZTC). Any other format (DICOM, 5DImage, etc.) can be converted to stack using other plugins avaible at `http://rsbweb.nih.gov/ij/plugins/index.html`. Size limitations are based only in PC hardware (RAM and graphics card memory avaible).

Initial configuration

All image format data is loaded in the first dialog shown. Here the user have to specify:

- *Voxel aspect ratio*: It define the relative size of a voxel. The ratio 1:1:1 means that every spacial direction (X, Y or Z) have the same pixel size.
- *Number of Z slices (Nz), channels (Nc) and time steps (Nt)*: Remember that Nz*Nc*Nt is equal to the stack size.
- *Stack order*: It Specify if the image archive order is XYZCT, XYCZT or XYZTC.
- *Classification task*: You can choose gray scale or RGB classification. Only for MIP its possible rendering without classification.

When accepted this dialog, the image is loaded into memory and visualization starts.

Control panel

On the right is placed the Control Panel, with the next options:

- *Fog settings*: To improve depth discrimination, voxel intensity can be decreased along object-observer distance. This is know as fog, and the attenuation function can be setted as *linear* (soft and homogeneous attenuation), *exponential* and *square exponential* (sharp attenuation). The fog coefficient modify fog intensity function.
- *Interpolation Mode*: In the render process, volume have to be scaled and voxels interpolated. The interpolation modes provided are *nearest neighbor interpolation* (Fastest, it only copy the nearest value) and *linear interpolation* (slower, but voxels are averaged with surroundings, smoothing the textures).
- *Volume settings*: *Angular position*, *volume aspect ratio*, *zoom* and *advancing frame speed* (for time varying datasets) can be set here. Theres also a *Reset button*, that revert all transformations applied.
- *Classification settings*: The classification methods can be switched anytime in the render process, and transfer functions aplied are modified with *Classifier Elements button*.

It also provide some keyboard control keys:

- Arrows (left, right, up, down): Apply volume rotation.
- Plus and minus: Advance and go back time steps.
- A and Z: Control zoom.
- T: Open classifier configuration panel.

Classifier transfer functions

The classification task is of great importance in DVR visualization, and usefull for MIP too. The classifier implemented is based in unidimensional transfer functions, which are easy to define but only uses histogram data to work, so It doesn't discriminate morphological information. In the transfer function configuration dialog, the user has to specify the number of TFs and their parameters (please refer to Figure 1):

- *Threshold*: Transfer function limits are defined by threshold (Ua and Ub). Their values are refered to the histogram domain (0-256 for 8-bits images and 0-65532 for 16-bits ones).
- *Slope*: The transition slopes (Sa and Sb) between 0 and the maximum function value can be defined, i.e. if Sa=0.1 and Ua=200, the transition is set from 190(0%) to 200(100%).
- *Output*: The user can choose how filtered voxels appearance would be, specifying maximum TFs values. If the classification mode is RGB, the output color is defined by its three RGB values, otherwise color is the original one. The opacity can be set in the same way. All them are clamped between [0.0-1.0].
- *Channel*: Each TFs responds only for one channel, so this have to be given. Channels are listed in order of appearence in the image archive, starting with 0.

Install instructions

To install, only a few steps are required:

1. Copy OpenGL 5DVisualizer .jar into the plugins folder.
2. Copy the JOGL JARs into the plugin folder, theyre placed in the JOGL directory inside the ZIP file.
3. According to your operating system and CPU architecture, copy the JOGL native libraries (*.os* for Linux users, *.dll* for Windows ones) to the folder where imageJ can load them when required.

   Windows: Copy the *.dll* files into ImageJ root folder (the folder where *imagej.exe* is contained).

   Linux: If You have root permission, You can copy the *.os* files into */usr/lib* or */lib*, where the Java Virtual Machine (JVM) usually search for

libraries. In case you cannot access this permision, or the JVM has another configuration, the path to the libraries have to be specified.

This can be done adding the *-Djava.library.path* option to the console execution line at the moment imagej is executed, or adding it to the run script contained in the *linux installer.jar* .

```
:/linuxConsole$ java -Djava.library.path=/home/user/
    JOGLlibraries/ -jar ij.jar
```

If your ImageJ version was installed through Synaptic, Aptitude or other system package managers, it probably has generated some script to execute ij.jar and automaticaly set its parameters when user call it with the line:

```
:/linuxConsole$ imagej
```

In this case, you have to find the script (usually in */usr/bin*) and add the option explained above when the command *java* is called.

Debian example:

```
1 {...}
2
3 JNI=-Djava.library.path=/home/user/JOGLlibraries
4
5 $JAVA_HOME/bin/java ${arch} -mx${mem}m ${JNI} ${modules} ij.
      ImageJ -ijpath ${ij_user_path} -port ${count} ${images}
6
7 {...}
```

Now you can run the plugin.

Feedback

Your feedback is really appreciated; any question, suggestion or possible ideas, please don't hesitate in write me and I'll try to fix it.