# The viewer4d: Viewing 4D (3D plus time) images with v4dlib

Contributed by VladimÃ-r Ulman
Last Updated Friday, 23 October 2015

This web page aims to present a CBIA developed open-source SW called viewer4d, a light-weight yet versatile viewer of 4D scientific images.

The purpose of the program is indeed to inspect 2D, 3D (stack of 2D images), 4D (usually a time-lapse sequence, or collection of results of image processing parametric study, of 3D images) images. On one hand, it allows to overview image data as a whole with such features as

- OpenGL volumetric rendering,
- various voxel intensity mapping regimes (incl. histogram stretches, manually adjusted levels, thresholding, and colour mappings),
- automatic annotations (user adjustable scale bar, time bar),
- automatic browsing (and looping) through the image sequence,
- on-the-fly subsampling and overlaying,
- exporting the views into image files for publication purposes.

On the other hand, it allows to zoom in and inspect values of individual voxels and their neighborhood.

The pixel values can be anything from 8bit and 16bit integers, real and complex values, or RGB and RGB16 colours. It is able to read single JPEG, TIF, PNG, ICS (Image Cytometry Standard), HDF5 images. To obtain broader support for file firmats, it is possible to attach the favourite Bioformats boundle to the program. The viewer4d can read a sequence of images across multiple folders and, of course, organize the order of images with our re-designed powerful Open dialog beforehand.

The program viewer4d is freely available "as is" under the GNU LGPL license, with open source code, also based solely on other open source packages. The program is an extension of our proven viewer3d .

The viewer4d was developed and tested on recent Windows and Linux platforms, for which we also provide binary packages.

The program is developed for scientists by scientists with the aim to address our daily image inspecting activities.



{mospagebreak title=Invoking the program, opening images}

Invoking the program

The viewer4d is a program with graphical user interface. It can be, therefore, invoked directly (without any command line parameters). The next step will typically be clicking the File->Open item to pop up the Open dialog... However, the program accepts also command line parameters.Â

The syntax is viewer4d [-command] [img1] [img2] ... [imgN] indicating an optional command as the first parameter followed by arbitrary many files. If no command is specified, the viewer4d starts with opened Open dialog in which the command line files will be pre-loaded into the middle panel. If, for instance, the command -seq is given, the viewer4d will try to directly open (without showing the Open dialog) the files from the command line as a sequence of images. Execute viewer4d --help or viewer4d.exe -help to see the current list of accepted commands. The utilization of the command line (and the commands) may prove useful when setting up associations between the viewer4d application and certain image file formats.

Opening images

The Open dialog has three main sections:

- the filesystem browser in the left panel,
- the list of images to be opened in the middle panel,
- the specification of the semantics of the list in the right panel.

The filesystem (left) panel allows to navigate towards the files of interest either by clicking the directory tree or by using

the (Directory chooser) button below. One may find it easier/faster to reach the desired folder via this button on some platforms, especially if the platform supports folder bookmarks (Note: one may set up a folder that would gather folder shortcuts on Windows; this folder would then act as a repository of folder bookmarks).

The image list (middle) panel is the main workhorse of the dialog. Here one may re-order the list items using the ensemble of buttons right to it. For clarity, the list displays only the filenames, in contrast to full paths. One may, however, display portion of the full paths by increasing the path separators number above. This may become handy when files in two folders are named the same.

Note that the Sort button acts on the currently displayed texts. The sort function sorts the whole list if no item is selected, otherwise it sorts only within the selected items leaving the unselected ones untouched.

The semantics (right) panel allows to specify, in the Group mode subsection, how many images make up one frame. Try to click through the radio buttons and notice how the background colours are changing in the image list. Note that some image formats (e.g. Image Cytometry Standard, ICS) support to store a complex image as one file (in contrast to keeping real parts in one file while imaginary parts in the second file separately); choose Scalar single files in this case.

If a single frame is to be composed from two or more files, the "Move to 1st" button in conjunction with "Select these:" button becomes extremely useful. The latter selects only those items in the image list that match (=include the pattern) positively with the expression pattern right below the button. Note that the expression honours actually standard regular expressions. The former then moves the selected files to the first slots within the frames. The "Move to 2nd" button moves to the second slot, and so on.

Lastly, the content of the image list is, by default, understood as a list of frames to make up an image sequence. The whole content is hence attempted to load into a single instance/window of the viewer4d. If one checks the "Each group in own window", then every frame is opened in an individual instance of the program. This will pop up that many new viewer4d windows as how many frames are there displayed on the image list.

One may further ask to transpose the "z" and "t" axes of the image sequence, and may request to downsample the images. Both modifications happens on copies of the original images as they are read from hard disk prior storing them in the program memory. The copies are not stored on the hard disk.

See the demonstration short video to briefly get the idea of how the Open dialog was intended to be used.

{mospagebreak title=Viewing a sequence of images}

Viewing a sequence of images

Once a 4D image sequence is loaded, one can directly inspect the single 3D frames of it. See the instructions on how to navigate within the 3D image.

Press "-" and "+" to change the current frame. Note that the viewing position is not changed, only the content of the views.

One can use the automatic browsing through the sequence. It is available from the Sequence menu. For the browsing, either manually or automatically, one can limithe range of frames to be iterated over, one can alter how many frames the program should step over (step over = Browsing step - 1), and what to do when end of the sequence is reached (e.g., nothing, loop from the beginning, change direction of browsing).Â

The mapping modes, that derive their parameters from histogram, are initiated from a histogram aggregated from all images in the sequence. If a histogram stretch is active (from the Mapping menu), the very same stretch is applied on all

images in the sequence.

See the demonstration short video to briefly get the idea of what features are available in the viewer4d.

{mospagebreak title=Studying a complex image}

Studying a complex image

The viewer4d handle complex images, that is image, where a complex number is associated to every pixel as its "pixel value". The program can display only the real parts of the number, only the imaginary part, the phase and the amplitude of the complex number. For each of the four regimes, its own histogram is calculated and the standard mapping modes can be applied (independently for every of the four regimes). Notice the mapping menu extends when viewing complex images.Â

Â

2D vector field as complex imageÂ

A 2D vector field, that is, an image where two-element vector is assigned to every pixel instead of an scalar intensity, can be also displayed in the viewer4d. One only needs to store and load the field as a complex image. The x-elements (y-elements) of all vectors have to stored instead of the real (imaginary) parts of the complex image. Inspecting pixel values of the complex image gives us values in the form a+b i which one should read as vector (a,b). Note that i is the imaginary unit. Note that phase mode is basically giving us azimuths (orientation angle with the 2D plane) of the vectors and aplitude mode reveals magnitudes of the vectos.

See the demonstration short video to briefly get the idea of what features are available in the viewer4d.

{mospagebreak title=Managing annotations}

Â Managing annotations

The sequence can be annotated with View->Show windows annotation. By default, the projection windows is decorated with a time bar in the XY window in bottom right corner, provided a sequence (more than 1 image) is loaded.

If windows annotation are enabled, the windows get further decorated with explanation of the projection windows (XY, XZ, YZ) in top left corners and with scale bars in bottom left corners.

In the preference dialog, View->Preferences, user may choose several parameters that affect appearance of the annotations:

- The text of annotation of the projection windows
- What real distance the scale bar should show
- How long should be the time bar so that the current relative position within the sequence can be best understood
- Both bars can adjusted in their length, width, colour, corner and offset
- Annotation texts can be changed colour, font (style and size) and position as well

Hints:

- Leaving an empty annotation text in fact disables this particular annotation
- Setting scale (time) bar width to 0px disables drawing of the scale (time) bar
- By adjusting offset one can avoid interference of the drawing with image data
- Setting appropriate negative offset of the scale bar, one can display only the bar itself (without its annotating text)

The offset, for instance in the case of top right corner, is a displacement vector of the top left corner of the annotation from the top left corner of the respective window. For bottom right corner, it is the displacement of bottom right corner of the annotation from bottom right corner of the window; and so on.

See the demonstration short video to briefly get the idea of what features are available in the viewer4d.Â

{mospagebreak title=Exporting views}

Exporting views

There are three export options implemented in the program to facilitate sharing results of the inspection of the data, by

means of another images... :-)

The View->Save 3D data is basically a tool to convert the viewed image into another image format. In particular, the current mapping mode as well as currently displayed annotations are not affecting the output image. The image data is saved as is, only in (possibly) different image format.

The View->Save view is the main exporting feature of the viewer4d. It indeed exports the current inspection set up, including the currently used mapping mode, cut pointers, displayed annotations, position in the image, and the orthogonal projection windows.

In this option, one may choose what single projection window to be in the output image, or whether all of the available projection windows should be included. In the latter case, there is an artificial border drawn between the windows.

While the previous exporting option is saving the orthogonal views of a displayed 3D image, the View->Save gallery exporting option saves a gallery of z-slices to, basically, sample the original 3D image. The current displaying setup is honoured just like in the previous case.

Exporting a sequence

Obviously, when there is a sequence of images loaded in the program, users are allowed to choose whether to export their image data on the currently inspected image or on a given (sub)sequence. This is a common part of the three exporting dialogs. In the case of exporting sequence, user is allowed to choose subinterval of the loaded sequence, over how many frames to step over and the output format. Currently, only a sequence of images is implemented. Note again that in this case, the template name has to include substring of several consecutive 'X' characters; these will be replaced with appropriate index of the output image.

See the demonstration short video to briefly get the idea of what features are available in the viewer4d.Â

{mospagebreak title=Compiling the program}

Dependencies

We have managed to compile the program on Windows platform with MS Visual Studio 2013 and 2015, on Linux with GNU GCC 4.9.x. The configuration of the program and adaptation to the particular system/environment is achieved with the CMake. The program, however, depends on several open source libraries (that have to be installed in the system beforehand):

- CBIA i3dcore and i3dalgo libraries (for handling image formats and some image manipulations)
- Many 3rd party libraries that both i3dcore and i3dalgo libraries depend on
- OpenGL interface and GLEW library (to achieve volumetric rendering of the images)
- wxWidgets (to obtain natively looking graphical user interface)

We provide the collection of supporting libraries for the MS Visual Studio 2013 and 2015 (on the second page). On Debian Linux, one may need to run the following commands:

sudo apt-get install libglew1.10-dev libjpeg62-turbo-dev libtiff5-dev libpng12-0-dev zlib1g-dev liblapack3-dev libblas3-dev libwxgtk3.0-dev libfftw3-dev

Compiling the program

On Windows, double-click the .sln file in the CMake output folder and compile the program the usual way. One has to pay attention not to mix the debug/release compilation configuration within the Visual Studio, the debug/release versions of libraries pulled in during the CMake configuration, and the version of the used MS VC (Visual Compiler) and the one with which the libraries were compiled.

On Linux, unzip the source code ZIP, cd into this folder, mkdir BUILD, cd BUILD, ccmake .., make, make install.

{mospagebreak title=License, Download, Authors}

License and Download

The program viewer4d, which includes the binary of the v4dwins library,
is available under the GNU LGPL license and is freely available, see links below.

- Installer for Windows operating systems [24.0 MB]

- Binary for Linux operating systems [3.0 MB]

- Source code [0.3 MB]

- Dependency libraries for Windows operating systems and Visual Studio 2013 compiler [105.7 MB]

- Dependency libraries for Windows operating systems and Visual Studio 2015 compiler [119.4 MB]

The ZIP archive with binaries for Debian Linux systems should be just unzipped anywhere and executed from that directory. Note that these binaries were compiled without the support of the Bioformarts boundle. A possible workaround is to open the image in FiJi and save there as TIFF.

On Windows you may need to download and install the MS Visual Compiler 2015 Redistributable pack (vcredist_x64.exe [13.9 MB]) prior running the program.

On Debian Linux you may need to execute the following two commands to ensure all required supporting libraries are available in the system:

sudo apt-get install libglew1.10 libjpeg62-turbo libtiff5 libpng12-0 zlib1g liblapack3 libblas3

sudo apt-get install libwxgtk3.0-0 libfftw3-long3 libfftw3-single3 libfftw3-double3 f2c

(If any of the packages is already installed, the command will skip over it.)

Authors

The program and the llibrary are extending, and hence are heavily based on, the viewer3d project . The viewer4d has been
developed in the Center for Biomedical Image Analysis, Faculty of Informatics, Masaryk University,
Brno, Czech Republic, also by several people.
The principal authors of the extension are Adriana Å mijÃ¡kovÃ¡ and VladimÃ-r Ulman. Contact VladimÃ-r Ulman for more

information about this project.