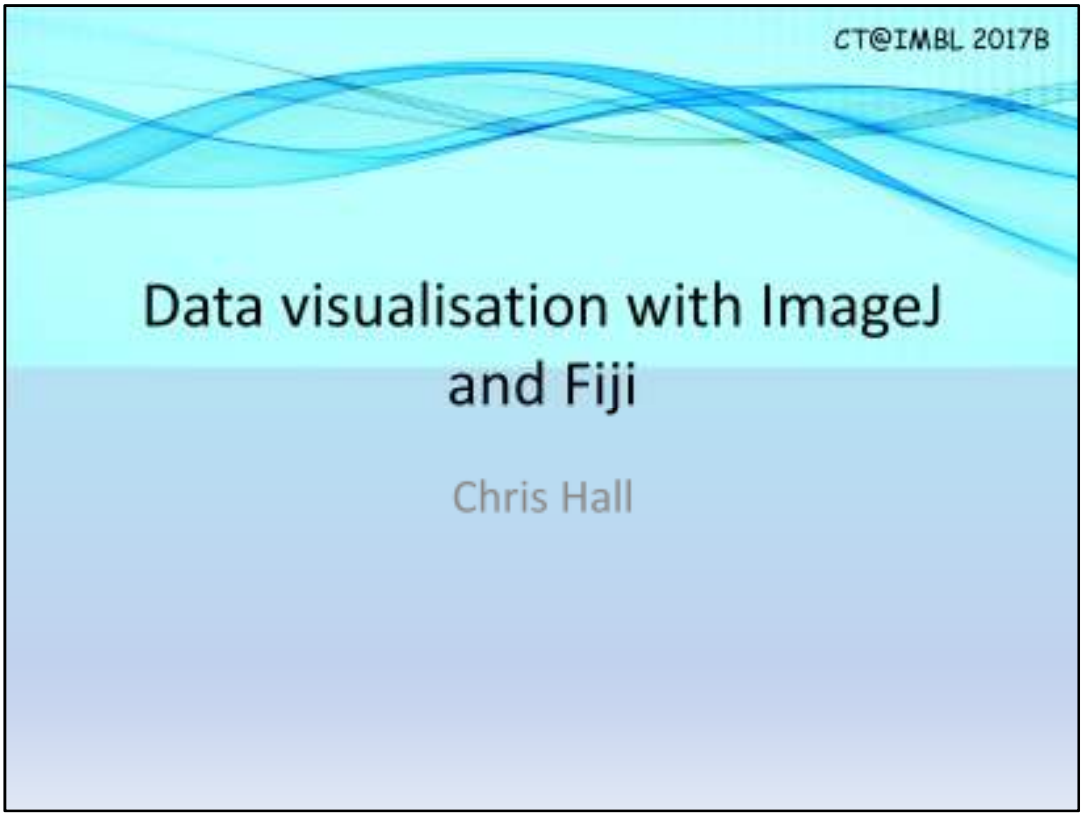
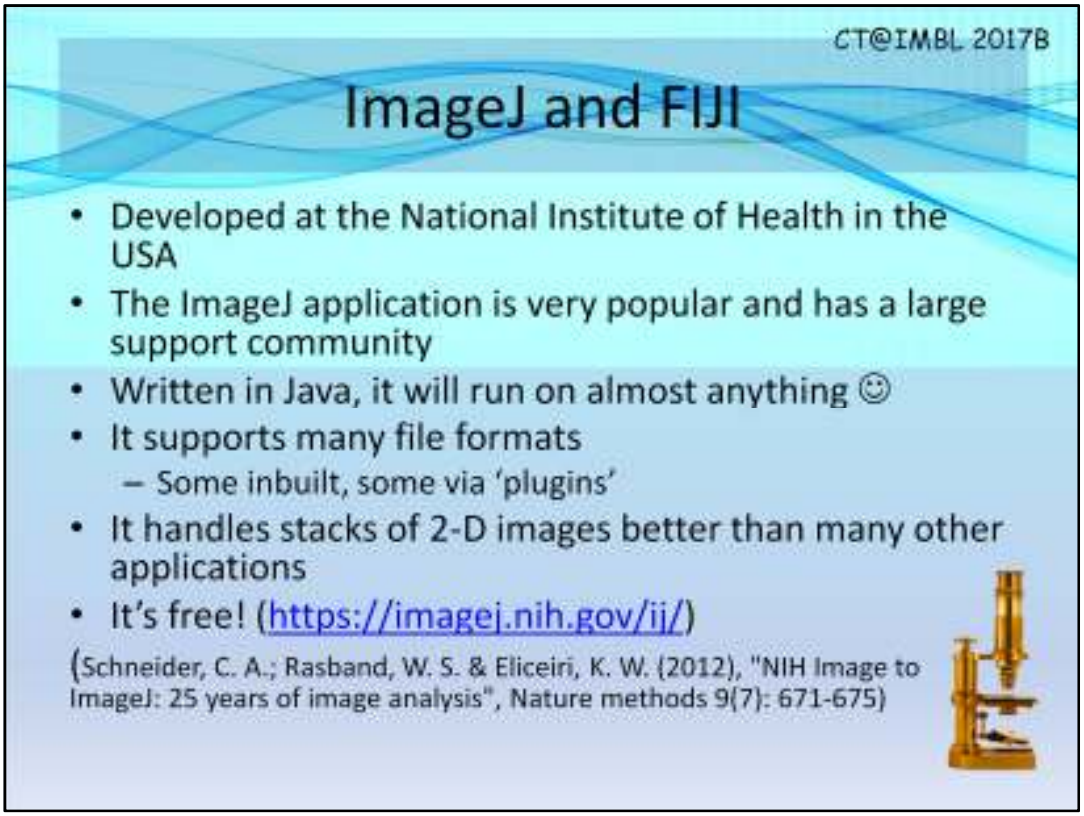
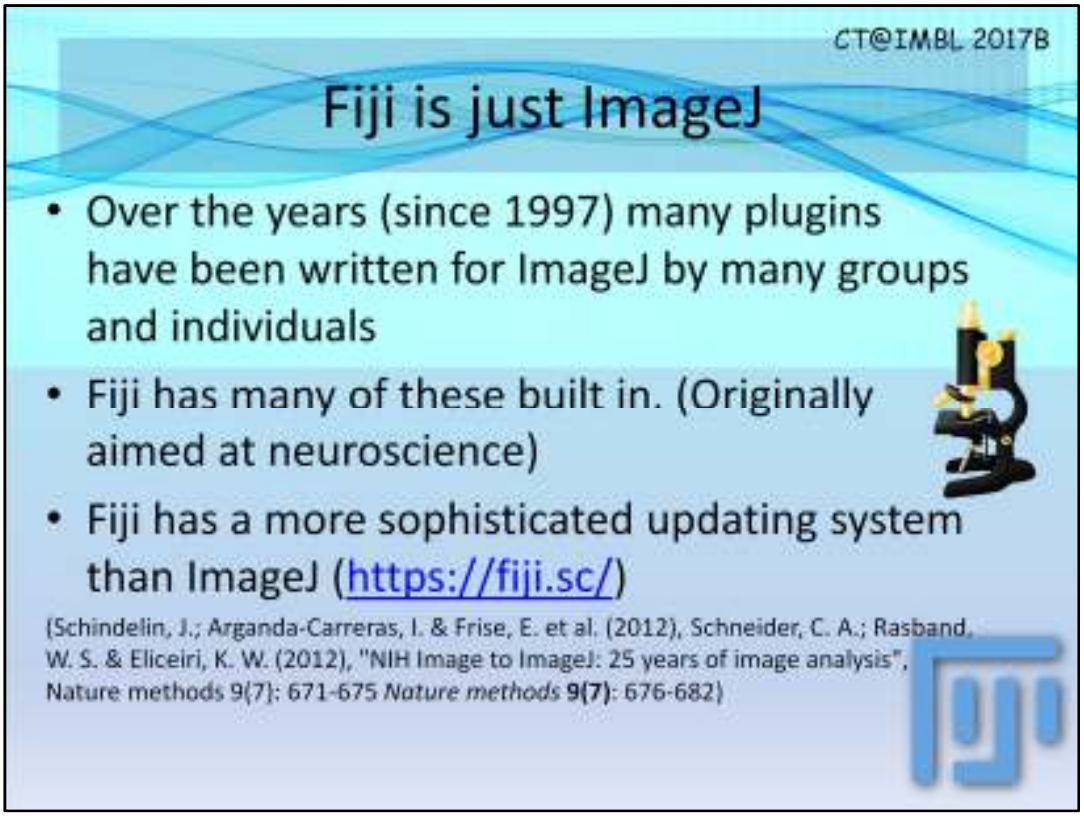


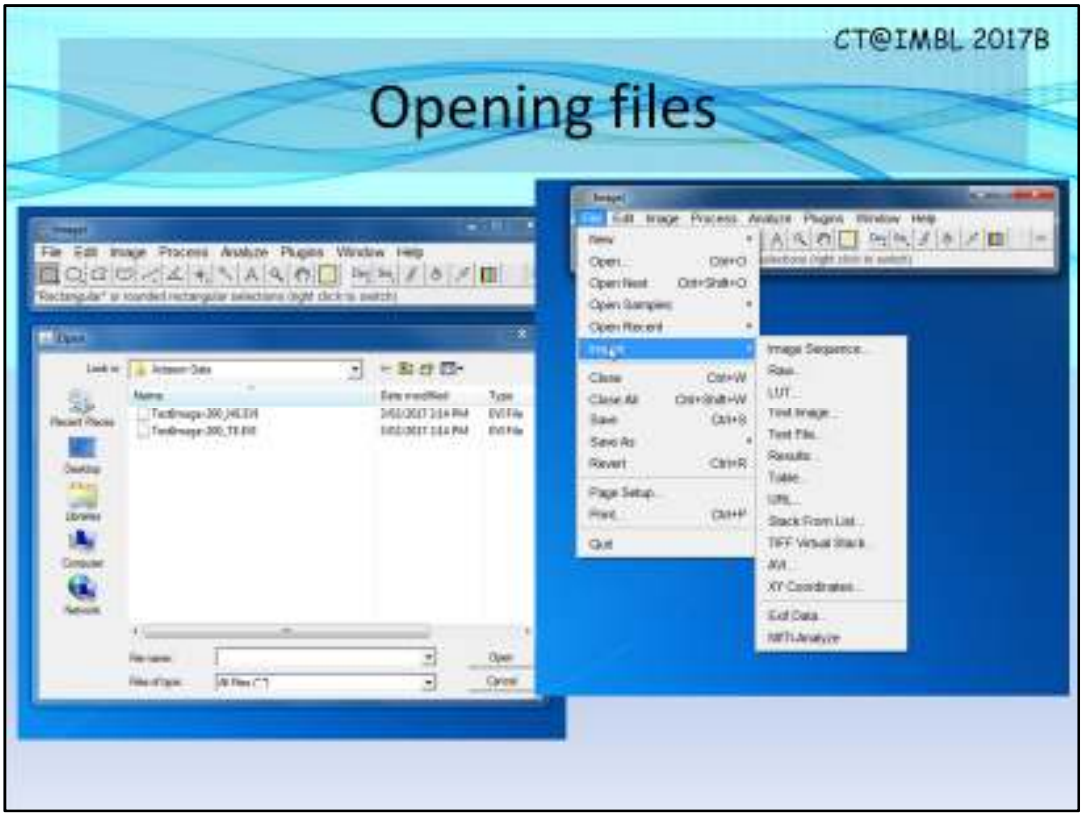
The preferred programs for examining, manipulating, and simple image analysis are ImageJ and Fiji



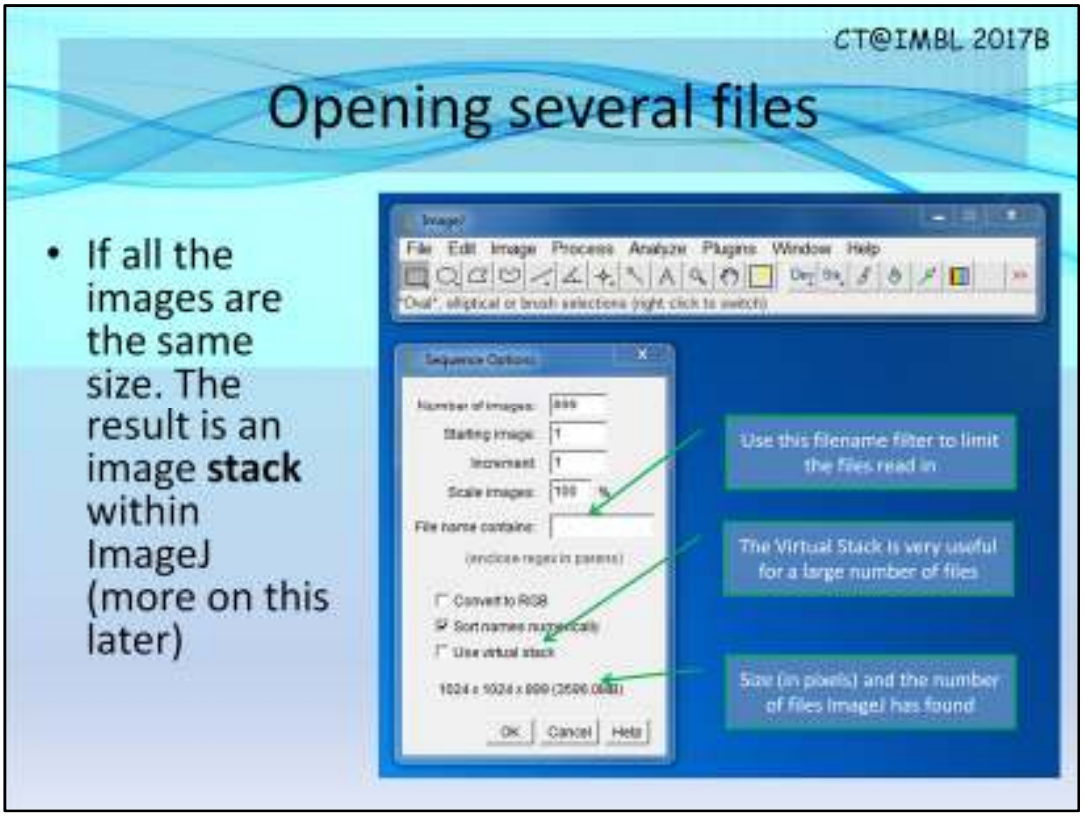
ImageJ is an open source Java application.
It runs on any computer with a Java 1.8 or later virtual machine.



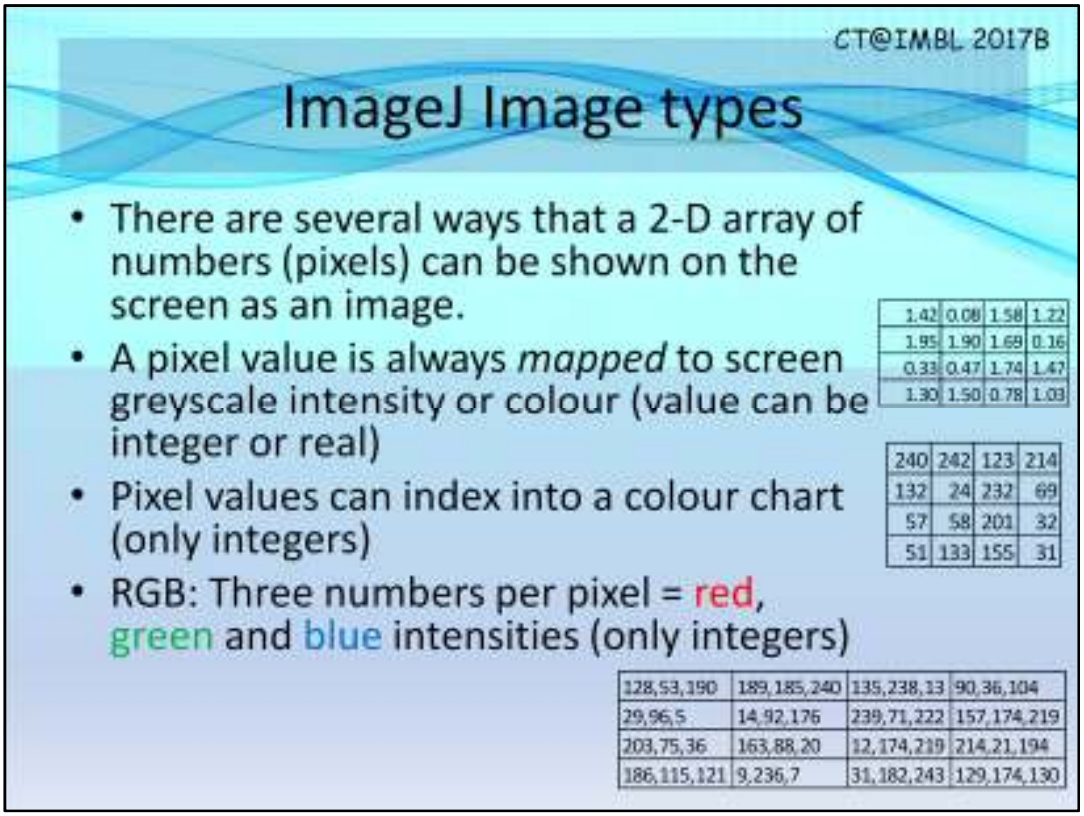
Fiji is built from ImageJ
It is a "batteries-included" distribution of ImageJ, bundling a lot of plugins which facilitate scientific image analysis



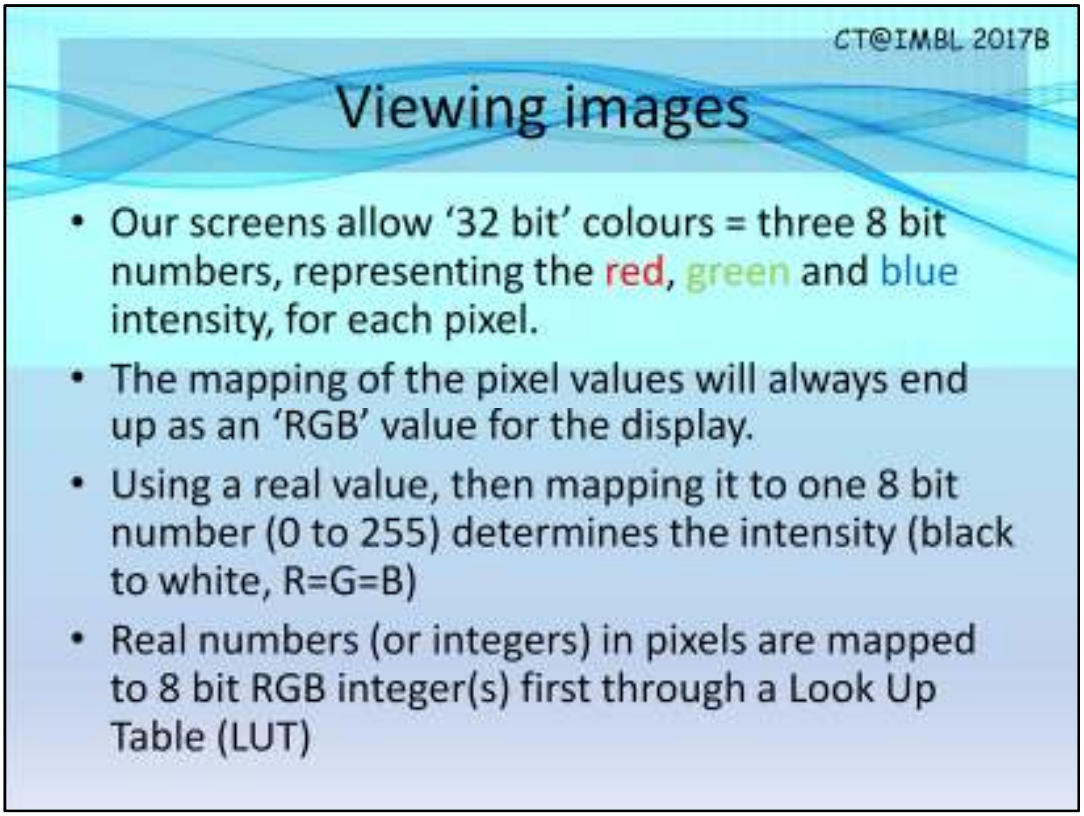
First step: opening up image files.
If the image file format is recognised by ImageJ, either by the extension or by recognition of the first part of the file it will just open it up.
Otherwise you 'Import' the data.
There are several importable file formats, including raw binary data, and text (ascii) files. ImageJ will even read in movies in .avi format. In doing this it will create an image stack.



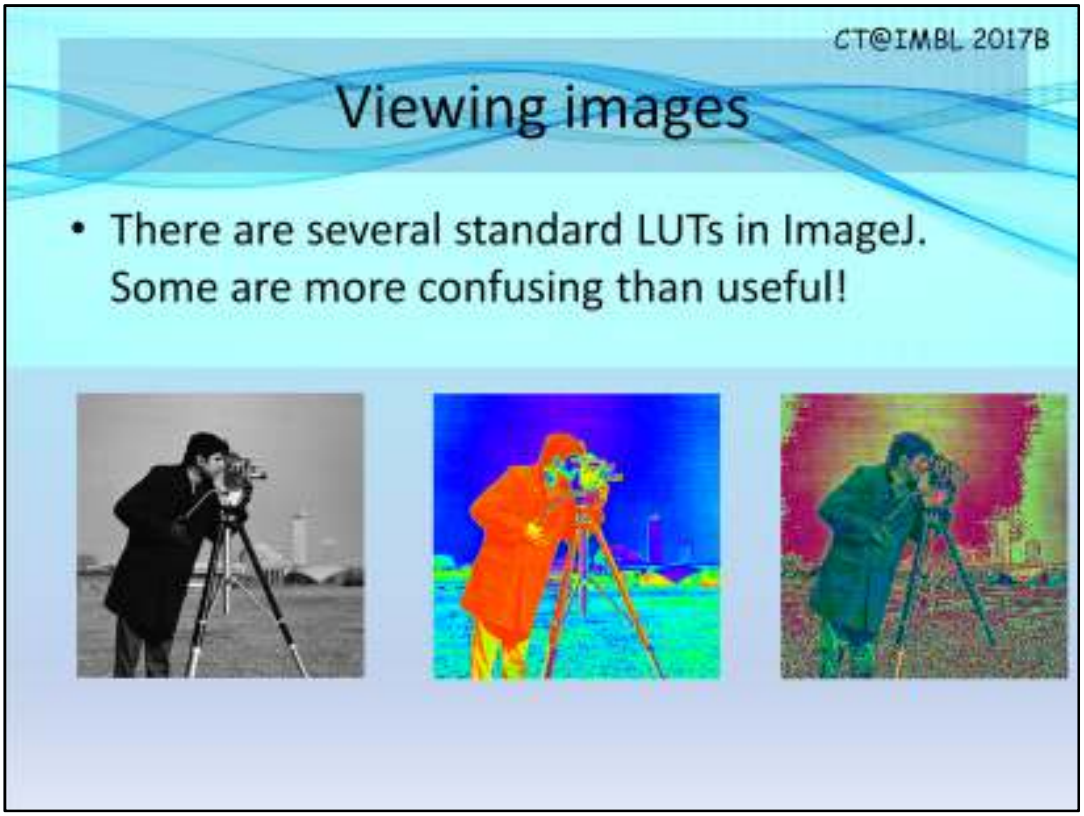
The image Sequence import is a very useful way to read in a lot of images.
If the files format is standard and the images are the same size. ImageJ will read them all in to a 'Stack'.
The image stack is a very useful construct. It is a 3-D cube created by layering each 2-D image data.
The natural X and Y axes are those of individual images. The Z axis runs down through the stack.
The raw CT data can be read in as a stack and examined quickly and easily this way.
If the stack of images is very large, ImageJ can leave them on the storage, rather than reading them all into memory.



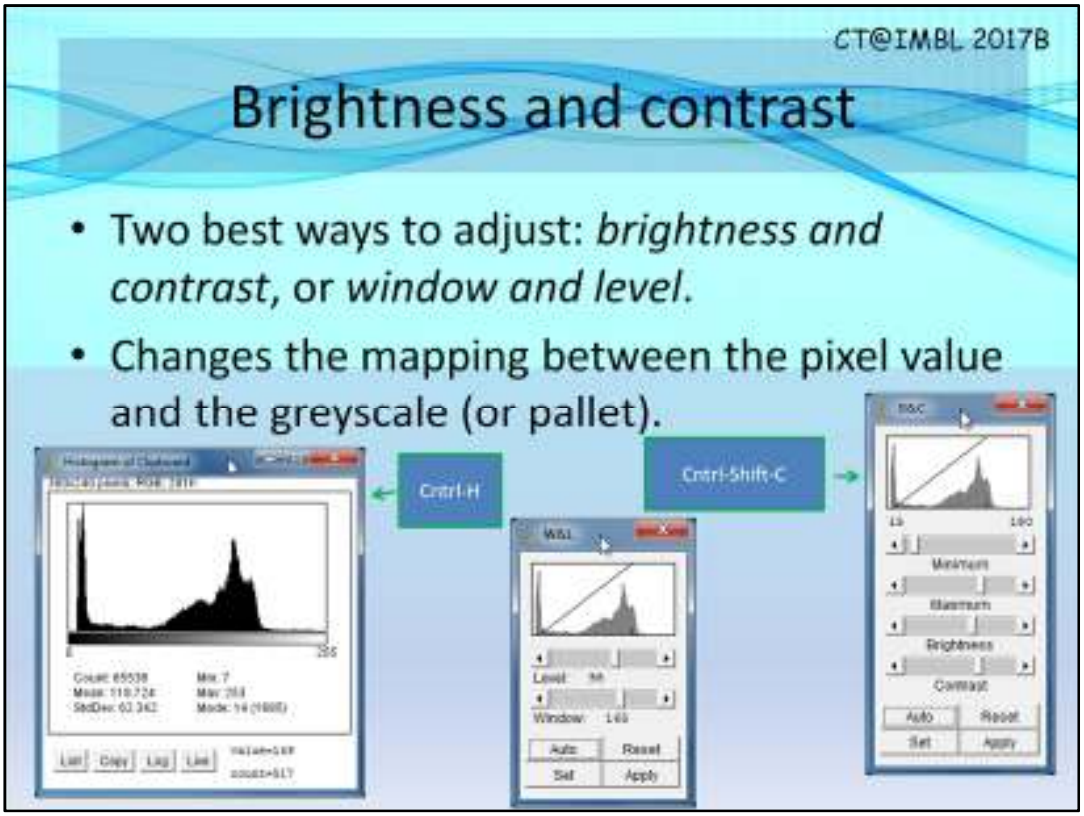
A brief introduction to image types. An image is really a 2-D array of numbers. Where the number usually represent some spatially resolved physical quantity.
To display them on a screen the numbers can be mapped to optical light colours, using the red/green/blue (RGB) system. Three integer numbers.
Colours can be confusing so often the image pixel intensity is mapped to just a grey scale luminosity. One integer number.
The numbers in the image array can be real, or integer values. They are generally called ‘pixel values’.
Raw data from the detectors is always quantised as integers.



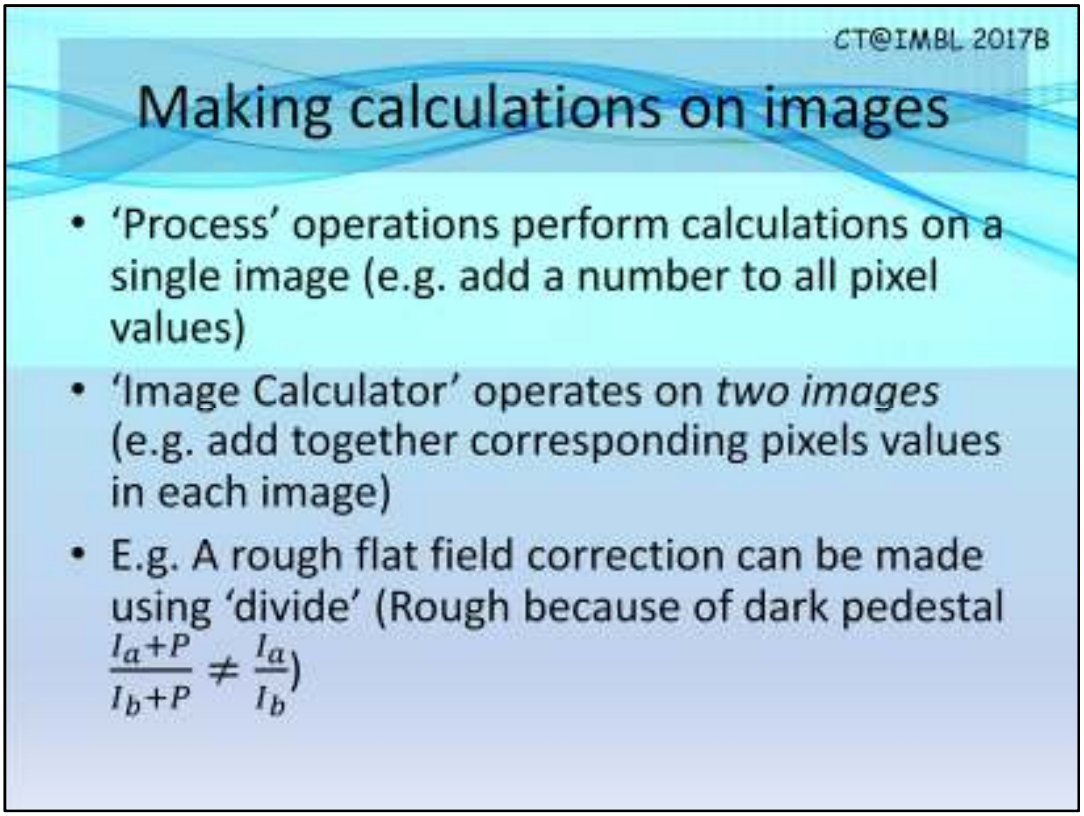
To display the image on the screen the pixel values are mapped to either a single integer for grayscale, or three integers for full colour.
The mapping is made through a look-up table or LUT.
ImageJ has a variety of in-built LUTs. It also has the ability to invert the LUT. So for instance the greyscale can range from white to black instead of black to white.
Full editing of the LUT is also supported.



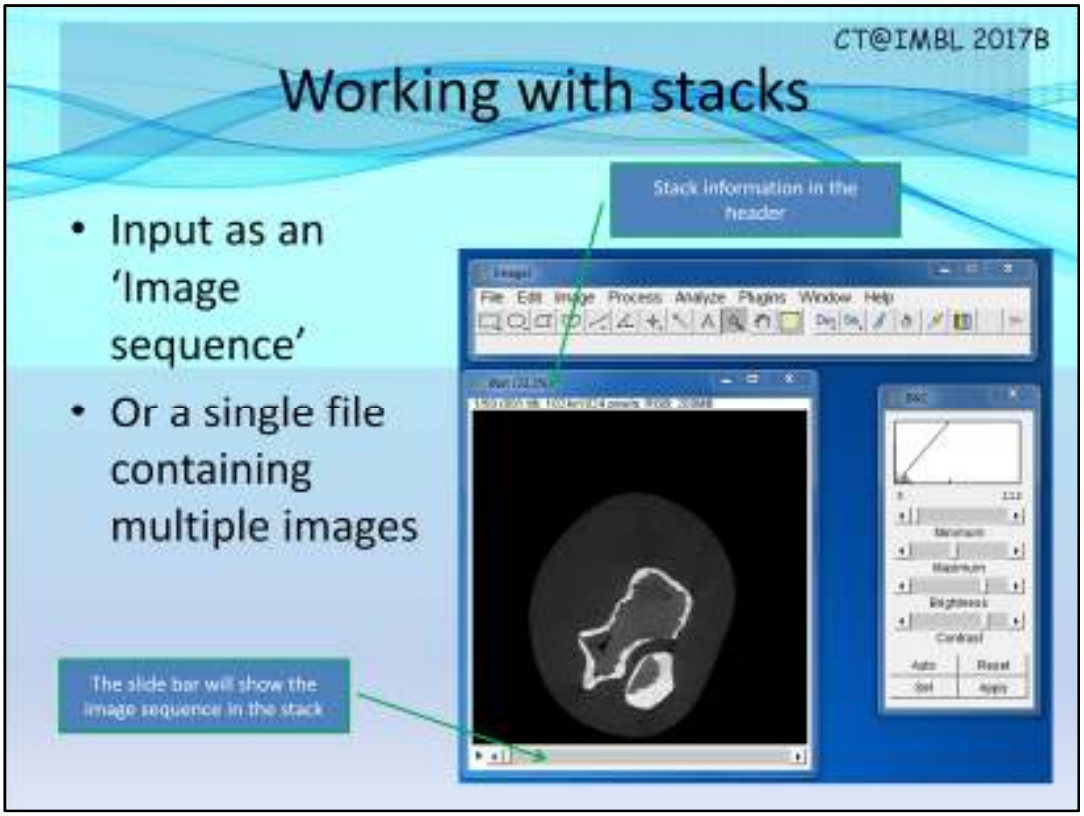
Choosing the correct LUT for your image can sometimes just be a matter of preference. The extra information gained from a colour LUT can be confusing in some cases.



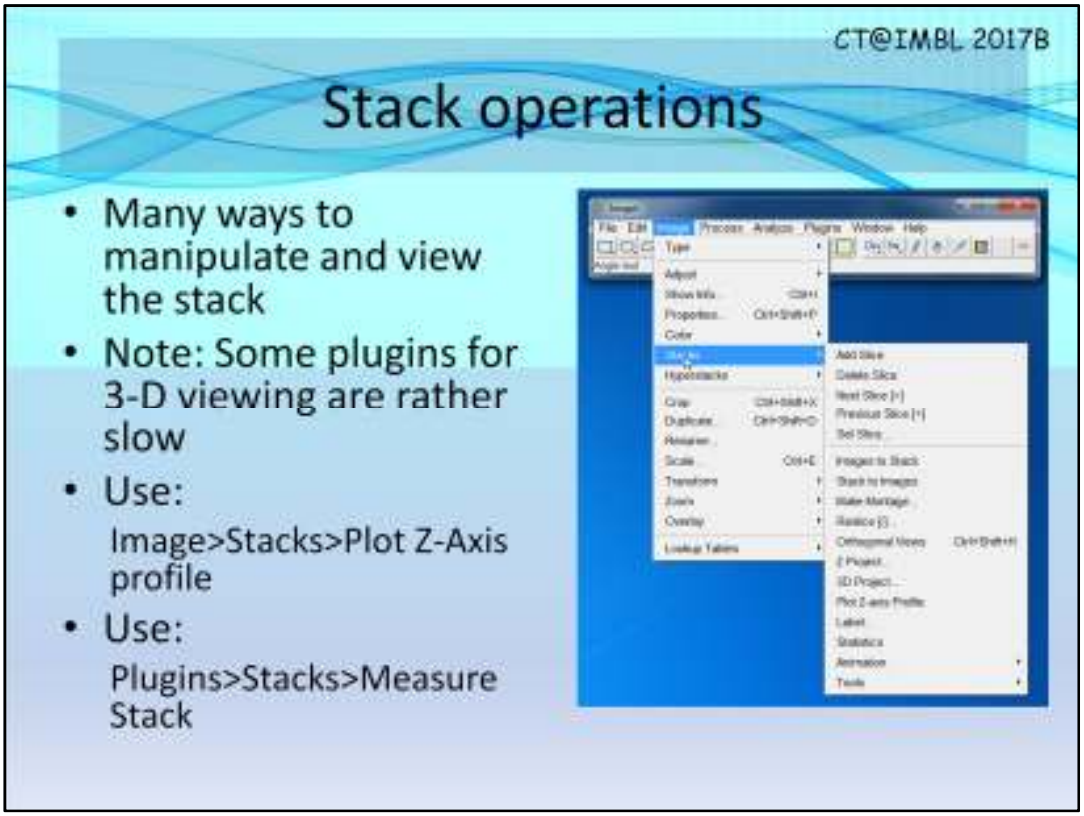
Changing the range of pixel values covered by the LUT can be done in several ways. ImageJ provides a histogram of the pixel values. Then allows the maximum and minimum values used by the LUT to be set.
When using a greyscale the slope, and offset of the intensity mapping is overlayed on the histogram.
This can be manipulated using the end-points, the brightness and contrast, or the window and levelling controls.



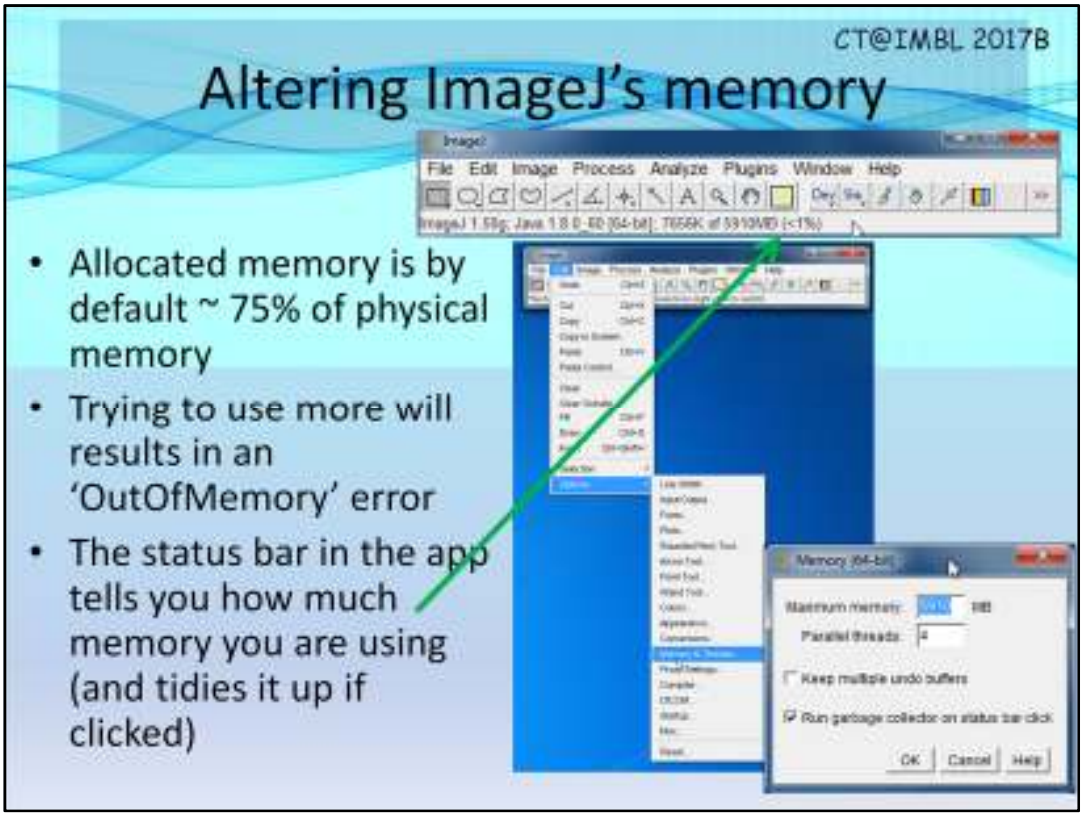
Calculations on pixel values are most often made from the Process dropdown.
An Image Calculator for allows the operands for the calculation to be pairs of images.
You can use this to make a rough flat field corrections to images with corresponding white fields by making a simple division between the two images.



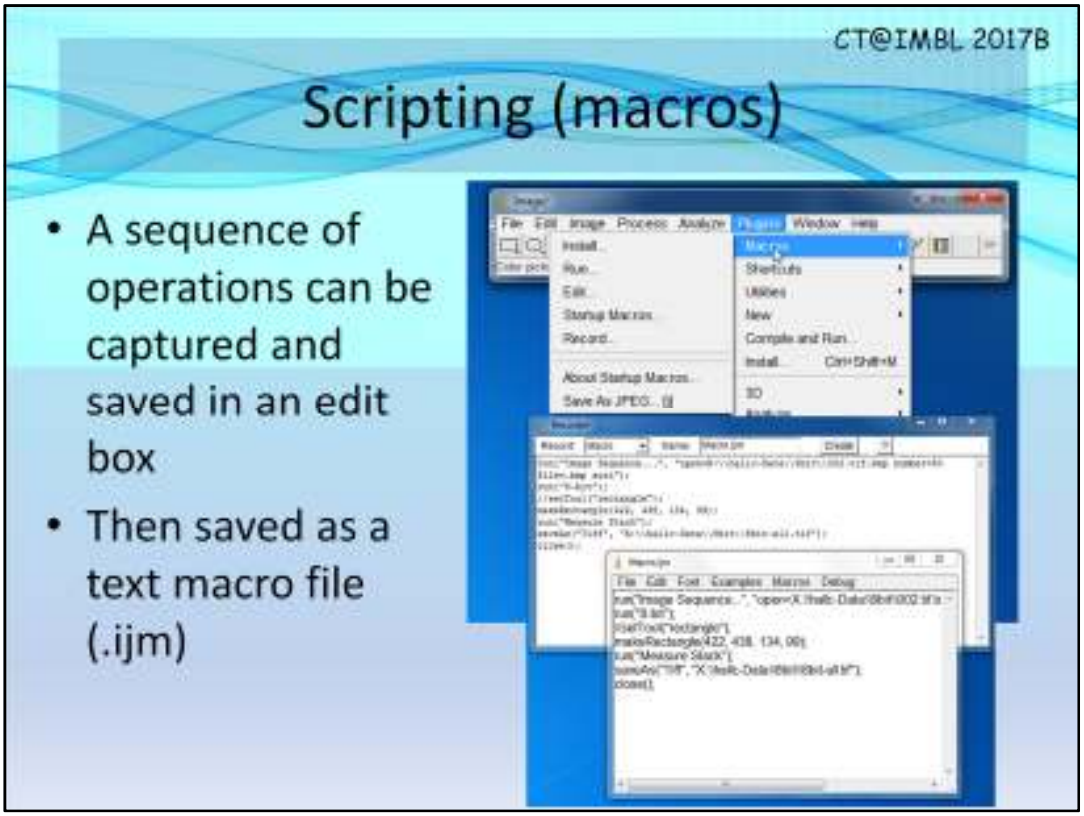
The 'stack' is a very useful data structure in ImageJ and Fiji.
It is a series of images which can be viewed and manipulated as a 3-D array (volume) of data.
Sequences of images (including moveis0 can be read into an image stack.
Individual 2D images or planes through the array, can be viewed in the image area.
As the slide bar on the bottom is moved, images or planes are displayed at a 'depth' corresponding to the slider position.



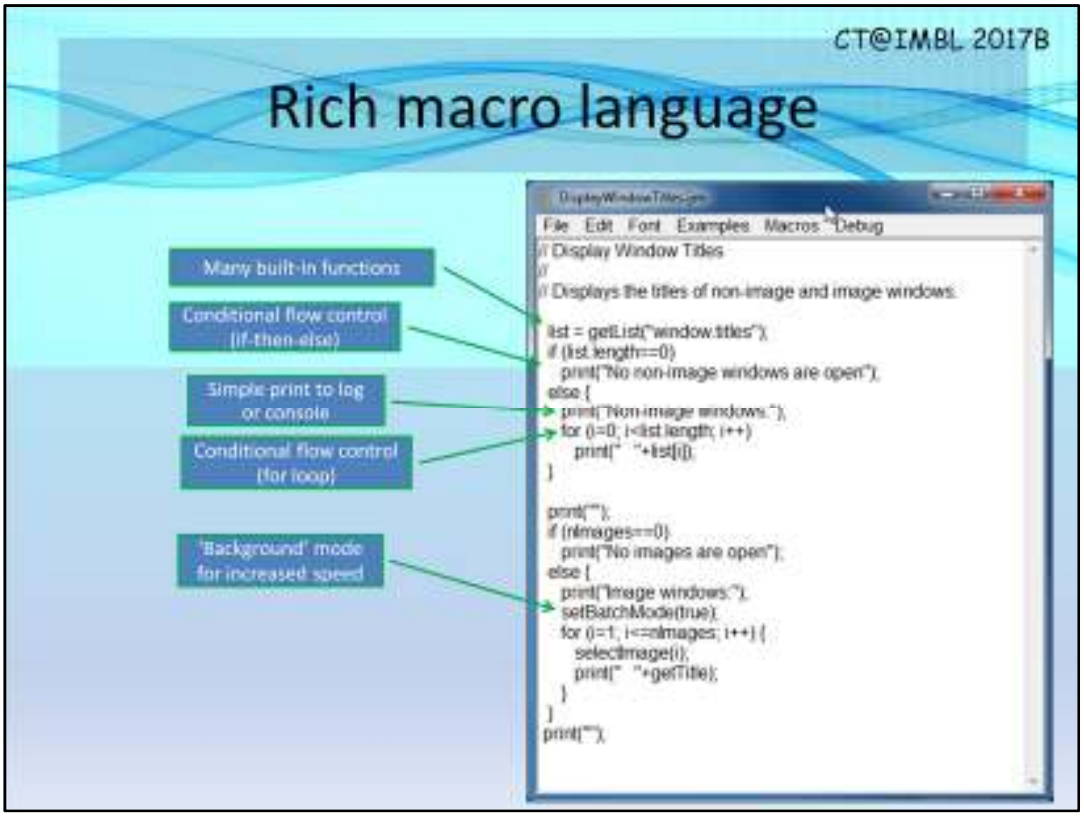
Measurements, line-outs (profiles) and many other parameters can be extracted from the image stack.
Most (but not all) of these abilities are in the Image>Stacks drop down menu.
When using the image calculators tool, ImageJ will ask if you want to make the calculation on the whole stack, or just the visible image.



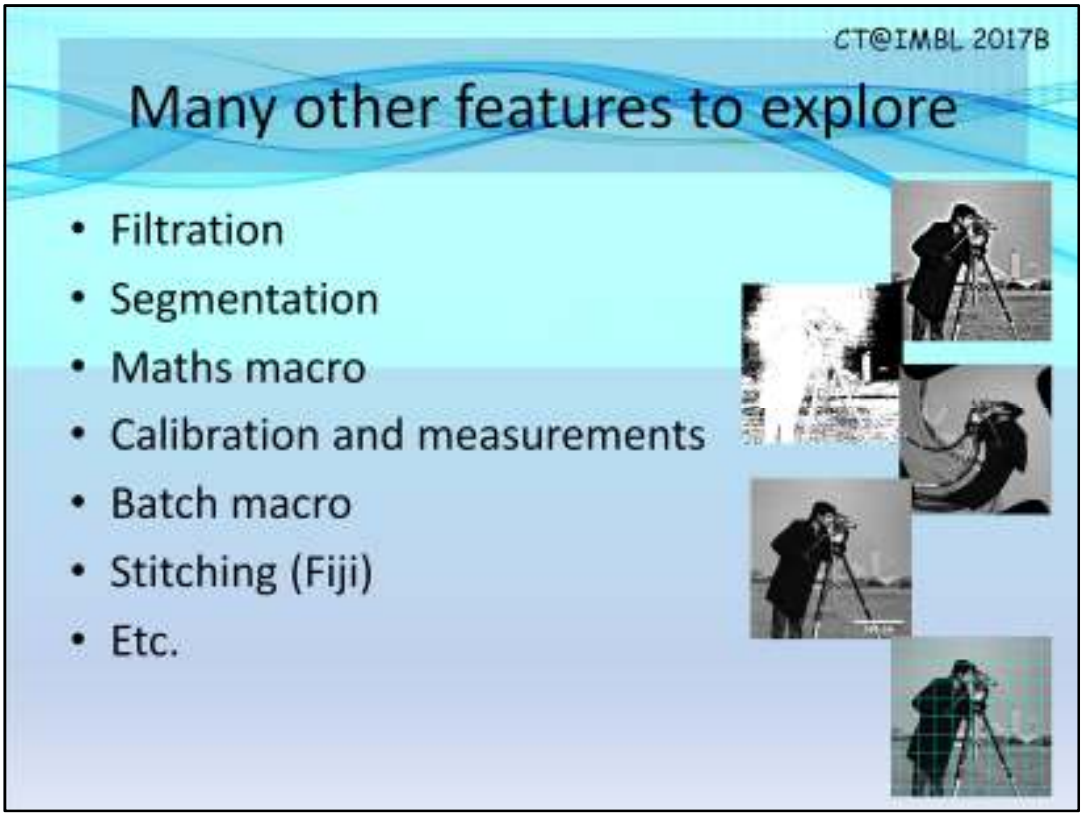
If the stack is deep and the images are large, the memory requirements for the stack can be high.
One way around this is to use the 'Virtual stack' option when reading in the image sequence.
Some operations cannot be run on virtual stacks and do require the whole set of images to be stored in RAM.
The maximum memory ImageJ/Fiji can allocate is determined by the parameter in the Edit>Options>Memory and Threads preferences.
It is recommended that no more than 75% of the available memory is allocable to the program.



Any ImageJ operation can be captured and replayed through the macro capability. Macros can be recorded whilst performing the operations. Then saved as java-script like text in a file.
Fiji has even more extensive scripting options. Allowing Python scripts to be interpreted.



The macro (script) language is full of features. Including program flow control, and many built-in functions.



There are lots of other features to explore. Too many to cover in this short talk.
A few of the more prominent ones are listed here.
In particular the ability to register, and stitch two adjacent images (with some overlap) is important.
That facility will be used in the presentation on image stitching later in the workshop.
The recommendation from me is to try these out yourself with your own images or image stacks.