ImageJ2

Directions & Goals

Strengths of ImageJ

- Cross-platform and public domain
- Macro language makes batch processing easy
- Wealth of existing image processing plugins
- Large science-oriented community of experts
- WSR: strong leader/maintainer/authority

Weaknesses of ImageJ

- Grew organically over the past decade
  - Many technical issues (details later)
  - Code is “micro-optimized”
- Difficult for community to contribute to core
  - No source control
  - Everything goes through WSR
- As a result, fractured community efforts
  - Always difficult to overcome barriers to entry

Specific Problems

Problems: VisBio

- Limited support for large datasets
  - Image planes larger than 2GB
  - Datasets larger than available RAM
  - VirtualStacks cache only one plane at a time
- No support for 3D visualization
  - Volume rendering
  - Arbitrary slicing
  - Realtime animation
- Also needs better support for ROIs

Problems: Slim Plotter

- No support for new dimensions
  - Emission spectra
  - Lifetime
  - Polarization
- No support for processing inherent to viz
  - Exponential curve fitting
  - Spectral unmixing

Problems: Fiji

- Distributing plugins is external to ImageJ
- Keeping everything up to date is complex
- No standard for documenting plugins
- Not easy enough to prototype algorithms
  - Plugins require too much boilerplate code
  - No modular command framework for using Macro Recorder with scripts
  - Case logic for multiple pixel types is messy
- AWT dependencies preclude headless use

Problems: TrakEM2

- No support for displaying registered images
  - No display mechanism for multiple image tiles
  - No mechanism for transformation from data to display (e.g., affine)
- Regions of interest are limited
  - No vector-based ROIs (i.e., ROIs are bitmasks)
  - Multiple ROIs are tacked on (ROI Manager)
  - Confusing interplay between ROIs, masks & thresholds with measurement tools

Problems: ROIs (Michael Doube)

- Recently I've been frustrated by ROI's being limited to 2D. With the emerging utility of the 3D viewer and the proposal that ImageJ 2.0 handles N-dimensional data, it makes sense that ROIs should keep up with this development.
- In other words, in an N-dimensional image, one should be able to specify and visualise an N-dimensional ROI. So you can have a 3D VOI, and a 4D VOI with time limits (or even changing shape over time), or limit the ROI to a channel (5D).

Problems: ROIs (J-Y Tinevez)

- I recently tried to code weird shapes as ROIs in ImageJ. They were the results of a segmentation with constrained shapes. Because I wanted to have something nice for the user, The ROIs had to be mouse-interactive (resizable, moveable etc..). I had a difficult time.
- Johannes proposed on the Fiji-devel list an abstract class whose goal was to facilitate this interaction.
- But we still gave to comply to ImageJ ij.gui.Roi master class, which is a concrete class in charge of drawing rectangle ROIs. Inside this class, there is everything: the logic to draw it, to interact with the user, with the image container, and the image data. Any homemade ROI must inherit from this class, there is no interface to implement.

Problems: ROIs (J-Y Tinevez)

- What I would like to propose here is to go for an interface hierarchy for ROIs, that is well decoupled, and that would allow the flexible design of new ROIs.
- We use ROIs for many purposes, for instance:
  - user interaction
    - draw a rectangle to crop an image
    - measure intensity with a complex area
    - add non-destructive annotations
  - as input/output for plugins, for instance a result of segmentation
- From this you can see that they need to:
  - know how to draw themselves as an overlay
  - comply to some interface to be an input of some plugins
  - know how to interact with mouse clicks and drag

Problems: μManager (N. Stuurman)

- 1. The Brightness/Contrast tool. Display of the histogram cannot be reliably set to the dynamic range of the camera (i.e., it always automatically goes back to the range of the minimum and maximum pixel value in the image, which can be extremely deceptive). No gamma correction. No method to update histogram when the image changes. No log display of the histogram. We ended up writing our own, but things are still clunky because acquired images (shown in a modified Image5D viewer) can only be controlled by the ImageJ B&C tool.

Problems: µManager (N. Stuurman)

- 2. Lack of plugin API. We have been bitten a number of times by internal changes in ImageJ breaking our code. Wayne is very responsive, but this still causes confusion.
- 3. Lack of standard for Multi-Dimensional viewer. We ended up using Image5D viewer, Hyperstacks came later. My impression is that the UI of Image5D is easier for users than the UI of Hyperstacks. In any case, we will be helped by a standard viewer for multi-dimensional images that integrates nicely with other ImageJ tools (like 3D viewers), and that is extensible (we do need to add a number of buttons that interface with image acquisition).

Problems: µManager (N. Stuurman)

- 4. MDI versus SDI. Not sure if this was on your list already (all of you have certainly debated this in the past!), but it seems that many people prefer the MDI model. On the Mac, it is pretty weird that a single application has different menus depending on which window you select (in our case, ImageJ windows versus Micro-manager window).

Problems: Miscellaneous

- G. Landini: no color space support (e.g., HSB)
- F. Hessman: domain coordinate systems
  - S&S are planning support within imglib
  - ImageJX consensus is to punt on this for now
  - Need to find a group with this use case first
- Legacy AWT interface limits use of Swing
  - ImageJ cannot use different L&Fs
  - AWT is missing features (JSpinner, JInternalPane)
  - Swing development is active, unlike legacy AWT

Problems: Compatibility

- Advantage of ImageJ: wealth of existing code
- Problem: ImageJ2 will break that code
- Examples:
  - ImageProcessor.getPixels()
  - All non-private, non-final fields
  - Subclasses created to sidestep API issues
  - Even private fields—setAccessible(true)

Problems: Interoperability

- FARSIGHT: ITK-driven segmentation routines are difficult to use from Java
- CellProfiler: How can scientists combine workflows between CellProfiler and ImageJ?
- OMERO: Database-backed images are kludgy
- Others: KNIME, Endrov, BioImageXD, PSLID...

Problems: Performance

- Traditional tradeoff between space & time
- Tradeoff between generality & performance
  - Moving toward generality requires that we remain aware of performance issues
  - But flexibility and usability remain paramount
- OpenCL is promising but negates many of imglib's gains in generality

Solutions?

Components of ImageJ2

- Major components of ImageJ2
    1) **Data model** – ij.process
    2) **Display** – ij.gui
    3) **Input/output** – ij.io
    4) **Regions of interest** – various
    5) **Scripting & plugins** – ij.macro
- All of these areas have significant limitations and will benefit from enhancements and refactoring

Components of ImageJ2

- Define/refine API for each component
  1) **Data model** – ij.process→imagej.process
  2) **Display** – ij.gui→imagej.gui
  3) **Input/output** – ij.io→imagej.io
  4) **Regions of interest** – various→imagej.roi
  5) **Scripting & plugins** – ij.macro→imagej.scripting
- How to draft the new API?
  - Spend a week in a room fleshing this out?
  - Or assign different packages to each of us?
  - Danger of over-planning

Components of ImageJ2

- Relevant technologies
  1) **Data model** – imglib library
  2) **Display** – Java AWT, JAI, Swing
  3) **Input/output** – Bio-Formats architecture
  4) **Regions of interest** – Java AWT, JHotDraw, OME-XML
  5) **Scripting & plugins** – Java 6 Scripting Framework
- More exploration of some technologies needed

Components of ImageJ2

- Break down implementation focus?
  1) **Data model** – Aivar?
  2) **Display** – Brian? Grant? Rick?
  3) **Input/output** – Barry? Curtis?
  4) **Regions of interest** – Brian?
  5) **Scripting & plugins** – Grant? Rick?
- Everyone must be familiar with good dev practices:
  - Unit testing (maybe even test-driven development?)
  - Dependency injection
  - Short methods & classes, few side effects

Next steps?