OME + ImageJ2

OME User's Meeting 2011

Kevin Eliceiri
(eliceiri@wisc.edu)

Curtis Rueden
(ctrueden@wisc.edu)

Laboratory for Optical and Computational Instrumentation (LOCI)
http://www.loci.wisc.edu
U. Wisconsin at Madison

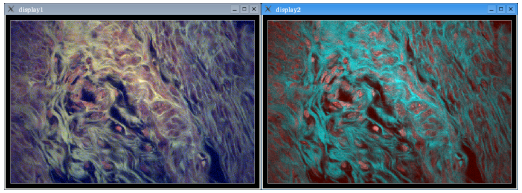
Laboratory for Optical and Computational Instrumentation

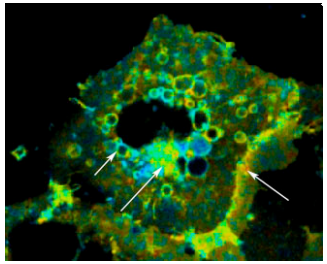
Mission of LOCI:

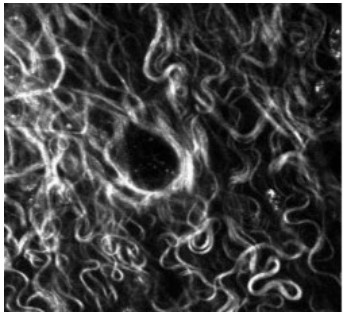
- New optical instrumentation to facilitate studies of the dynamics of living specimens.
- Better software for capture and visualization of dynamic, 3-D biological events
- Been OME Development partner since 2003
- Image Informatics for multidimensional data
  - spatial and temporal
  - spectral and lifetime dimensions
  - polarization

Laboratory for Optical and Computational Instrumentation

Our data:







Fluorescence,
Spectra, Lifetime
(λ,τ)

Time-Lapse
(t)

Dynamics

Chemistry

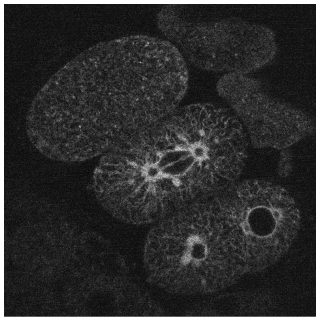
Optical Microscopy

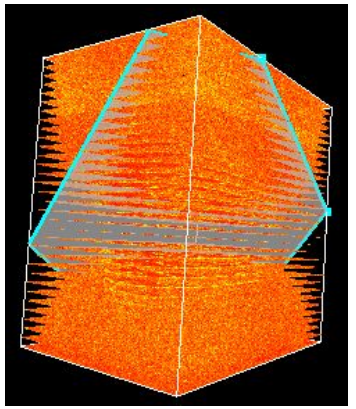
Physics

Structure

Phase, Polarization,
Scatter, Harmonics
(ϕ, θ,…)

Space
(x, y, z)





Supporting Technologies

Hardware
Acquisition software
Analysis and Visualization
Data Management

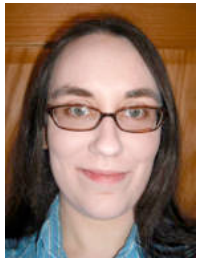
ome

LOCI
Laboratory for Optical and Computational Instrumentation

LOCI-Madison Informatics Group- 2011

Kevin Eliceiri
Director
LOCI

Curtis Rueden
Lead LOCI
Developer



Melissa Linkert
Lead Bio-Formats
(Glencoe Programmer
in residence)

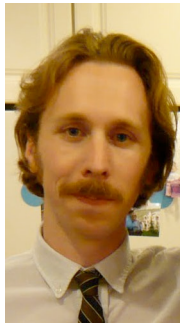
Johannes Schlindelin
Visualization
Scientist

Barry Dezonia
ImageJ2
Developer

Jimmy Fong
Lifetime Analysis
Researcher

Aivar Grislis
ImageJ2
Developer

Mark Hiner
Bio-Formats
Graduate Programmer



Andrew Johnson
Project Forward
Graduate Researcher

Eric Alexander
OME XML Metadata
Graduate Programmer

Abhinav Tallavajhula
Acquisition
Graduate Programmer

ome

Laboratory for Optical and Computational Instrumentation

Overview of our OME efforts

- Specific OMERO linked applications
  - BK Cho in Murphy lab on Omero.searcher
  - Forward Project for data dissemination

- Originated OME-TIFF in 2004
  - Now fully integrated into OMER0
  - Used by many open and commercial tools
  - Our current focus on robust tools to read and write OME-TIFF

- Originated Bio-Formats in 2005
  - partnership with OME and Glencoe
  - Over 100 formats
  - Over 25,000 installations
  - Recent focus on native bindings

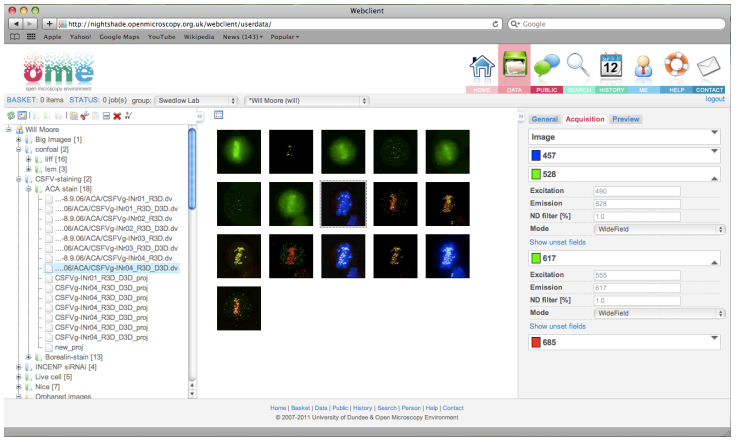
- XML Schema Improvements for Acquisition
  - Our WiscScan software and now MicroManager
  - Plans to extend to others that want richer “OME-TIFF”

- Interoperability between OME and other tools

- ImageJ 2.0 (ImageJDev.org)

Laboratory for Optical and Computational Instrumentation

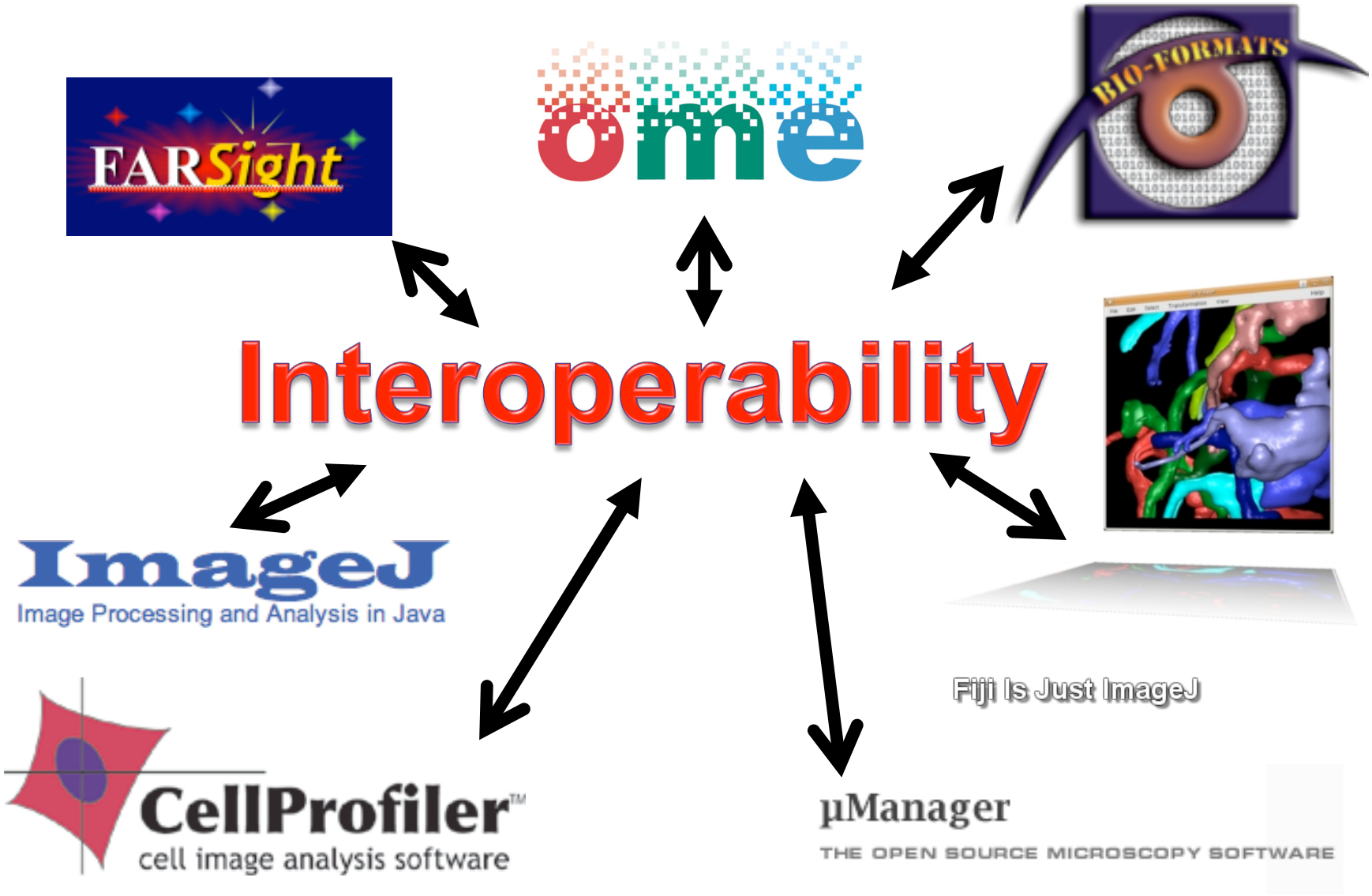
Bio-Formats: the tool for interoperability



JCB Data Viewer

ImageJ
Image Processing and Analysis in Java

Open Source Toolkit Development



Laboratory for Optical and Computational Instrumentation

LOCI/UW-Forward Collaboration

- Goal:
  - Provide access to LOCI datasets via Forward:
    - UW Libraries' system-wide resource discovery application (forward.library.wisconsin.edu)
- Requirements:
  - Mapping OME-XML elements to METS/MODS elements
  - Adding description and opt-in functionality to WiscScan
  - Automating metadata transformations (OME to METS) and export of data/metadata to Forward

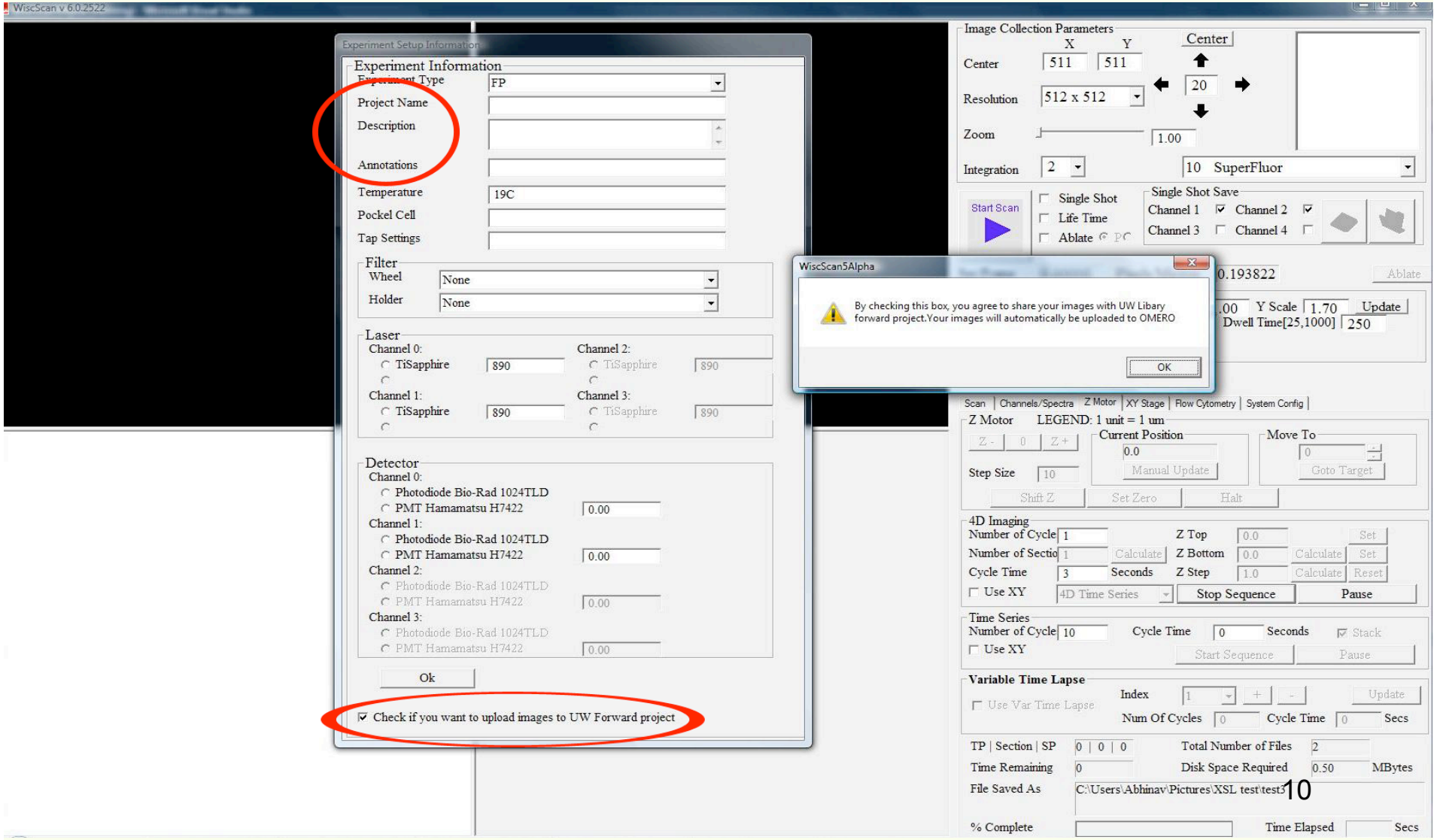
Laboratory for Optical and Computational Instrumentation

Getting Data into Forward

- Step 1: Add descriptive metadata in WiscScan
  - Title (Project Name)
  - Experiment description (Description)
  - Keywords (Annotations)
- Step 2: Opt in
  - Check-box in WiscScan experiment window
- Step 3: WiscScan and Forward do the rest
  - “Thumbnail” and metadata automatically uploaded to an OMERO directory that Forward harvests regularly

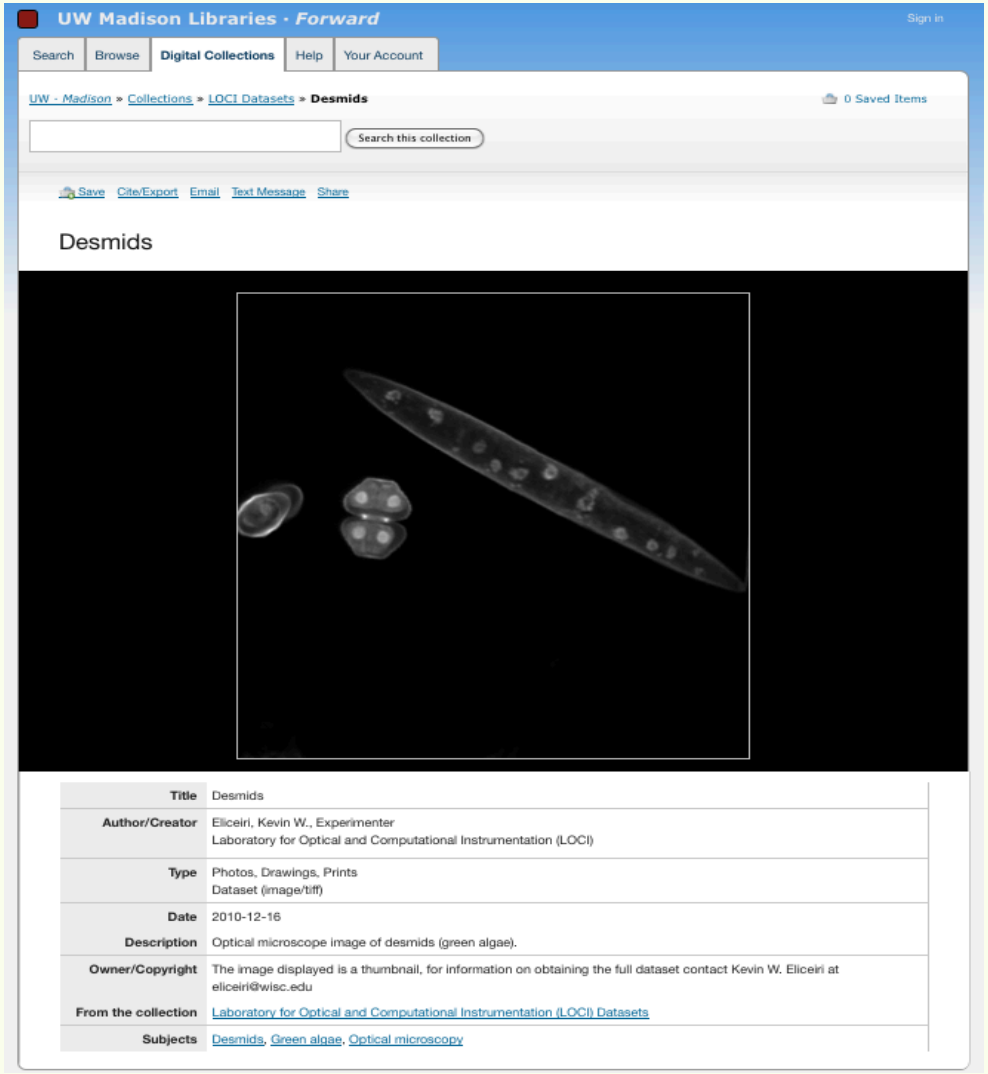
Laboratory for Optical and Computational Instrumentation

WiscScan Fields \Needed for Forward



Laboratory for Optical and Computational Instrumentation

Sample LOCI Record in UW-Forward



Laboratory for Optical and Computational Instrumentation

Big Picture

- LOCI data fully searchable and discoverable in UW-Forward
- Users searching for books/articles will now find datasets alongside traditional catalog results
- Greater exposure for data and emphasis on research data as a valuable campus resource
- Can be extended to Micromanager

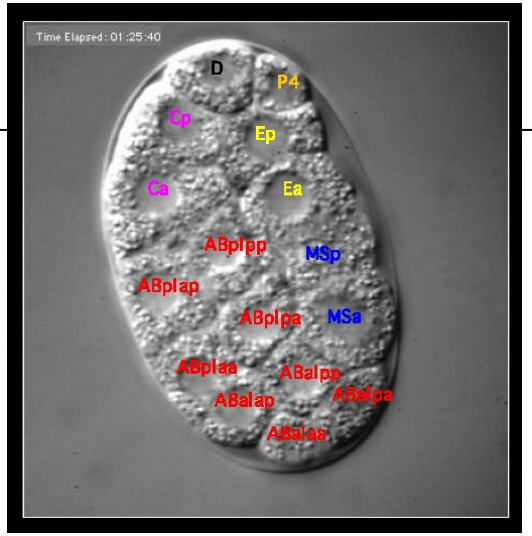
Do the Math

- The potential size of a biological dataset has exploded:
  - A typical biological image is 512 x 512 pixels
  - There can be 30+ slices in an image stack
  - There can be 100+ timesteps in a time series
  - There can be 30+ channels at each pixel
- 512 x 512 x 30 x 100 x 30 x 4 bpp > 80 GB

Desired Features for visualizing a modern biological dataset:

1. View image stacks in 2D and 3D
2. Fast animation of these stacks
3. Roam through all dimensions in real time
4. Useful ways to sift through data's non-spatial dimensions, to discover meaningful data quickly
5. Maintain a reasonable memory footprint even when the dataset is larger than the computer's available memory

Laboratory for Optical and Computational Instrumentation



ImageJ

- ImageJ is an excellent 2D image viewer
  - Developed by Wayne Rasband at NIH
  - http://rsb.info.nih.gov/ij/
- Supports image stacks (3D), but only one plane at a time
  - Plugins such as Volume Viewer allow for 3D rendering
  - "WiscScan 4D Data Browser" plugin facilitates quick browsing of 4D datasets produced by WiscScan
    - http://www.loci.wisc.edu/4d/
  - Another option for multidimensional support is the Image5D plugin
- Very large datasets can be handled using the Virtual Stack Opener plugin with reasonable performance
- Overall, ImageJ is excellent for #1, #2 and #3, but not as developed for #4 and #5

Laboratory for Optical and Computational Instrumentation

- ImageJDev: an NIH-funded project to produce the next generation of ImageJ
- Partnership between several institutions:
  - LOCI at UW-Madison
  - MBL at Woods Hole
  - Broad Institute of MIT and Harvard
  - Fiji group (MPI-CBG, Uni/ETH Zurich, etc.)

See also: imagejdev.org/collaborators

Laboratory for Optical and Computational Instrumentation

ImageJ Hackathon-Madison 2011

Guiding Principles

- Preserve backwards compatibility
- Maintain good performance
- Support N-dimensional imaging
- Use common input and output for data
- Minimize complexity
  - Introduce dependencies only when benefits outweigh disadvantages
- Employ modern software development practices

Vision

- What is ImageJ's greatest strength?

Vision

- What is ImageJ's greatest strength?
  - It's ***extensible*** by writing plugins

Vision

- What is ImageJ's greatest strength?
  - It's ***extensible*** by writing plugins
- How can we expand on this potential?

Vision

- What is ImageJ's greatest strength?
  - It's ***extensible*** by writing plugins
- How can we expand on this potential?
  - Plugins as ***modular*** “building blocks”

Vision

- What is ImageJ's greatest strength?
  - It's ***extensible*** by writing plugins
- How can we expand on this potential?
  - Plugins as ***modular*** “building blocks”
- What does modularity gain us?

Vision

- What is ImageJ's greatest strength?
  - It's ***extensible*** by writing plugins
- How can we expand on this potential?
  - Plugins as ***modular*** “building blocks”
- What does modularity gain us?
  - Modularity facilitates ***interoperability***

The Need

- Extensibility
- Modularity
- Interoperability

Aims

1. Improve ImageJ's core architecture
    a) Separate data model from user interface
    b) Develop extensions framework for algorithms
    c) Broaden the image data model
2. Expand interoperability with other tools
3. Grow ImageJ community resources

See also: imagejdev.org/proposal

The Challenge

- How do we maintain compatibility?
  - Will plugins and macros still work?
  - Do other programs work with ImageJ 2.0?

Design

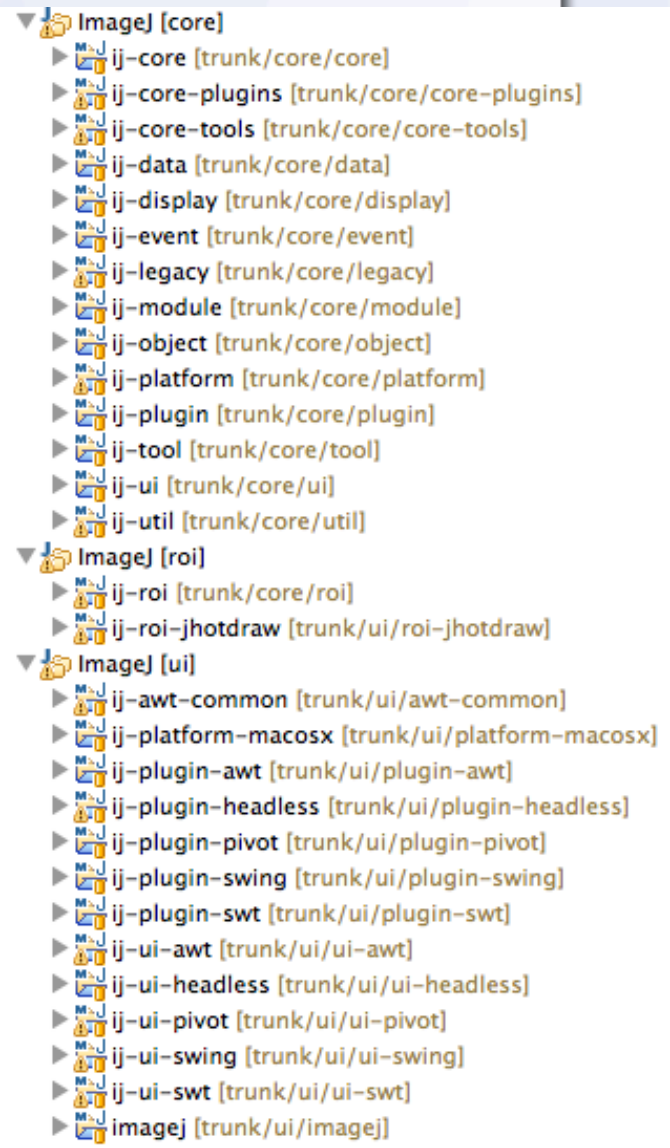
- Considered several design approaches
  - Iterative (modify existing IJ1)
  - Greenfield (new application)
  - Delegation (change IJ1's internals)
  - Adaptation (leave IJ1 alone)
- Adaptation: IJ2 includes IJ1 as a library
- IJ1 and IJ2 grow and evolve together

Progress


OUR COMPETITORS JUST MADE OUR NEW FIVE-YEAR PLAN MOOT.
WHILE WE WERE STRATEGIZING, THEY WERE DOING SOME-THING I BELIEVE THEY CALL "WORK."
www.dilbert.com scottadams@aol.com
©2008 Scott Adams, Inc./Dist. by UFS, Inc.

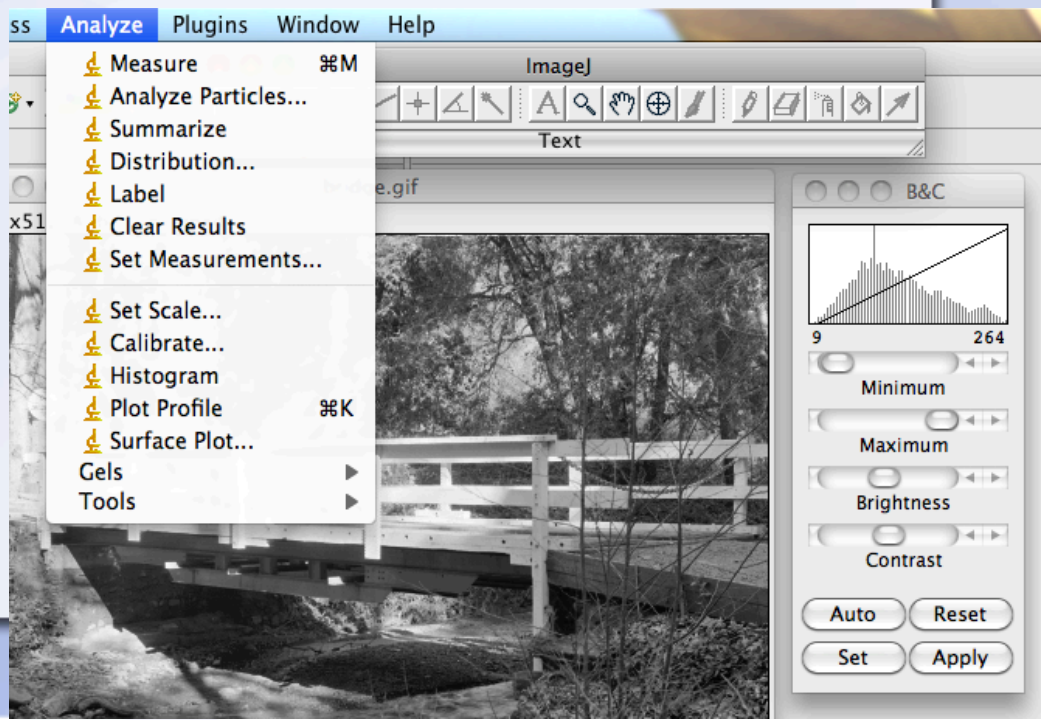
Modular Design

- Divided software into functional components
- Facilitates a cleaner design
- Provides flexibility to those using ImageJ as a library
- Mid-April: 15K lines of code
- Mid-June: ~43K lines of code
  - Excludes ImgLib and IJ1



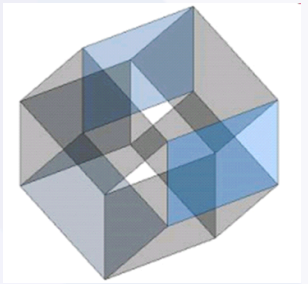
Progress: Compatibility

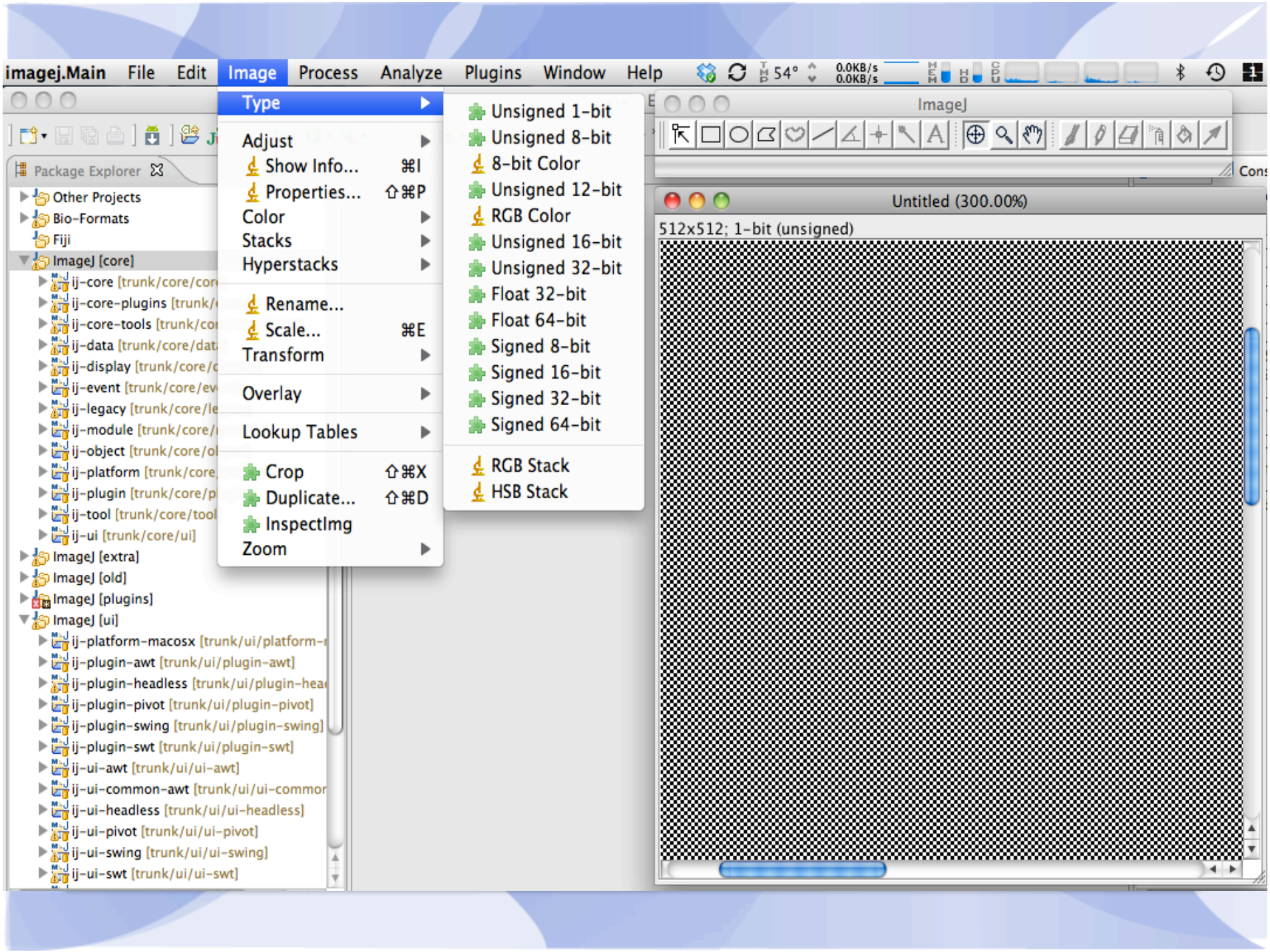
- Data structures converted between IJ1 and IJ2 behind the scenes
- Microscope icon indicates IJ1 legacy plugin



Progress: ImgLib

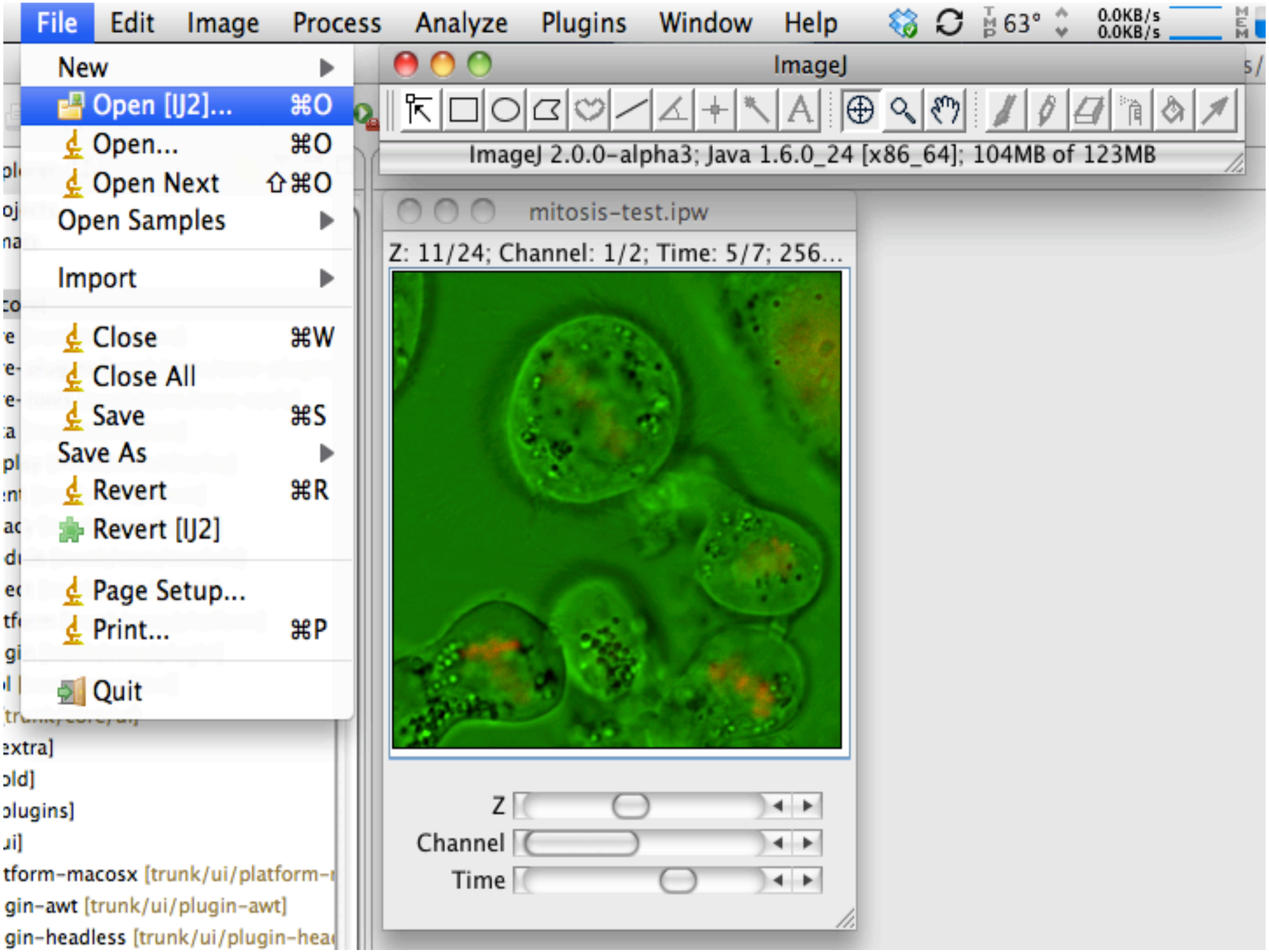
- ImgLib written by Stephan Preibisch & Stephan Saalfeld of MPI-CBG
- Many possible storage strategies
  - Data in array, file on disk, database...
- Type-independent algorithms and plugins
  - Signed and unsigned integer, floating point
  - Bit depths: 1, 8, 12, 16, 32, 64 bit




imagej.Main
File
Edit
Image
Process
Analyze
Plugins
Window
Help
Type
Adjust
Show Info... ⌘I
Properties... ⇧⌘P
Color
Stacks
Hyperstacks
Rename...
Scale... ⌘E
Transform
Overlay
Lookup Tables
Crop ⇧⌘X
Duplicate... ⇧⌘D
InspectImg
Zoom
Unsigned 1-bit
Unsigned 8-bit
8-bit Color
Unsigned 12-bit
RGB Color
Unsigned 16-bit
Unsigned 32-bit
Float 32-bit
Float 64-bit
Signed 8-bit
Signed 16-bit
Signed 32-bit
Signed 64-bit
RGB Stack
HSB Stack
ImageJ
Untitled (300.00%)
512x512; 1-bit (unsigned)
Package Explorer
Other Projects
Bio-Formats
Fiji
ImageJ [core]
ij-core [trunk/core/core]
ij-core-plugins [trunk/core-plugins]
ij-core-tools [trunk/core-tools]
ij-data [trunk/core/data]
ij-display [trunk/core/display]
ij-event [trunk/core/event]
ij-legacy [trunk/core/legacy]
ij-module [trunk/core/module]
ij-object [trunk/core/object]
ij-platform [trunk/core/platform]
ij-plugin [trunk/core/plugin]
ij-tool [trunk/core/tool]
ij-ui [trunk/core/ui]
ImageJ [extra]
ImageJ [old]
ImageJ [plugins]
ImageJ [ui]
ij-platform-macosx [trunk/ui/platform-macosx]
ij-plugin-awt [trunk/ui/plugin-awt]
ij-plugin-headless [trunk/ui/plugin-headless]
ij-plugin-pivot [trunk/ui/plugin-pivot]
ij-plugin-swing [trunk/ui/plugin-swing]
ij-plugin-swt [trunk/ui/plugin-swt]
ij-ui-awt [trunk/ui/ui-awt]
ij-ui-common-awt [trunk/ui/ui-common-awt]
ij-ui-headless [trunk/ui/ui-headless]
ij-ui-pivot [trunk/ui/ui-pivot]
ij-ui-swing [trunk/ui/ui-swing]
ij-ui-swt [trunk/ui/ui-swt]

Progress: Bio-Formats

- Adapted ImageJ to use Bio-Formats natively for reading file formats
- Files are opened as N-dimensional, ImgLib-backed images


File
Edit
Image
Process
Analyze
Plugins
Window
Help
New
Open [IJ2]... ⌘O
Open... ⌘O
Open Next ⇧⌘O
Open Samples
Import
Close ⌘W
Close All
Save ⌘S
Save As
Revert ⌘R
Revert [IJ2]
Page Setup...
Print... ⌘P
Quit
ImageJ
ImageJ 2.0.0-alpha3; Java 1.6.0_24 [x86_64]; 104MB of 123MB
mitosis-test.ipw
Z: 11/24; Channel: 1/2; Time: 5/7; 256...
Z
Channel
Time

Progress: Declarative Plugins

- Existing plugin example:

```
ImagePlus original = WindowManager.getCurrentImage();

GenericDialog gd = new GenericDialog("\"Tubeness\" Filter”);
gd.addNumericField("Sigma: ",
  (calibration==null) ? 1f : minimumSeparation, 4);
gd.addMessage("(The default value for sigma “ +
  "is the minimum voxel separation.)");
gd.addCheckbox("Use calibration information", calibration!=null);

gd.showDialog();
if (gd.wasCanceled()) return;

double sigma = gd.getNextNumber();
boolean useCalibration = gd.getNextBoolean();

TubenessProcessor tp = new TubenessProcessor(sigma, useCalibration);
```

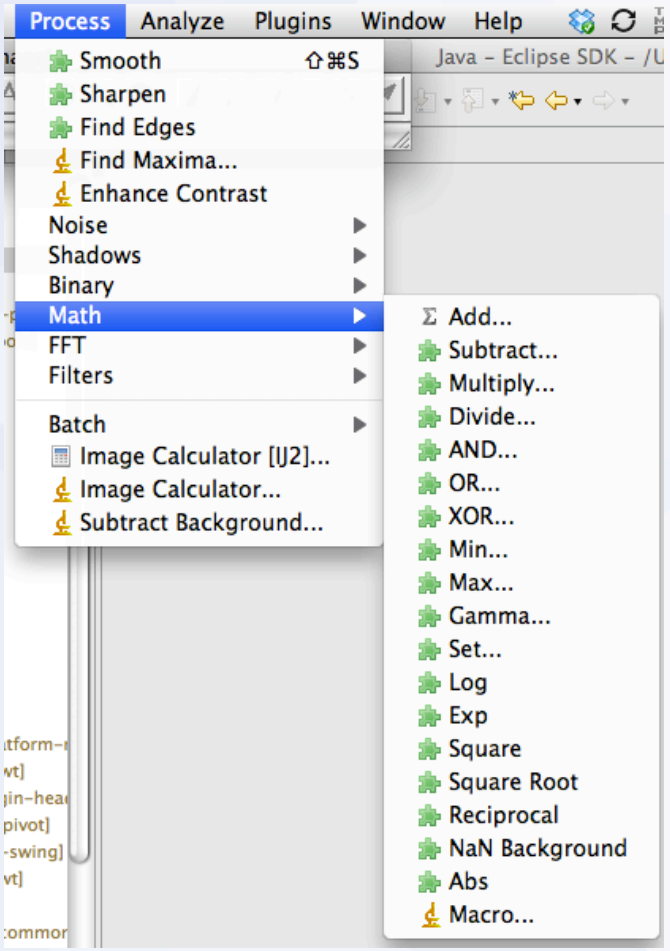
Progress: Declarative Plugins

- Declarative plugin example:

```
@Parameter(label="Input image")
public ImagePlus original = null;

@Parameter(label="Sigma")
public double sigma = 1.0;

@Parameter(label="Use calibration")
public boolean useCalibration = false;

@Parameter(label="Output image", output=true)
public ImagePlus result = null;

public void run(String ignored) {
  if (original == null)
    original = WindowManager.getCurrentImage();
  TubenessProcessor tp = new TubenessProcessor(sigma, useCalibration);
  ...
}
```

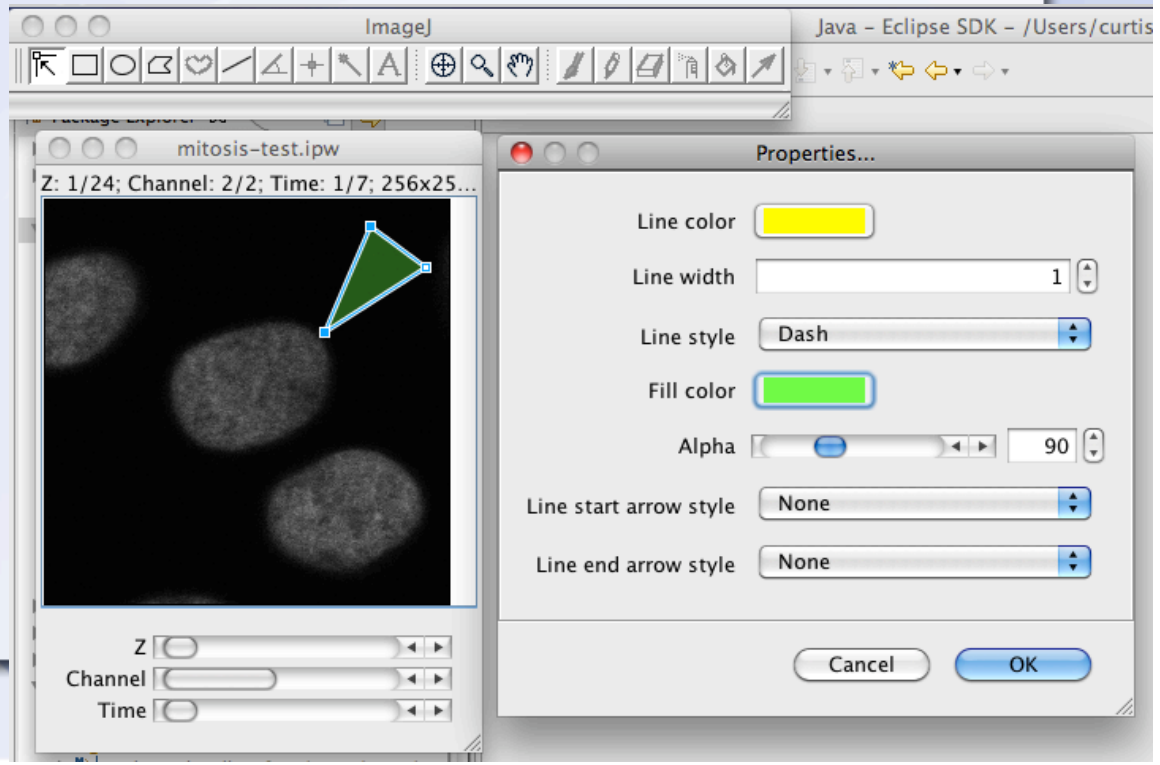
Progress: Core Plugins

- Translating ImageJ core plugins to ImageJ2 and/or ImgLib—93 done so far



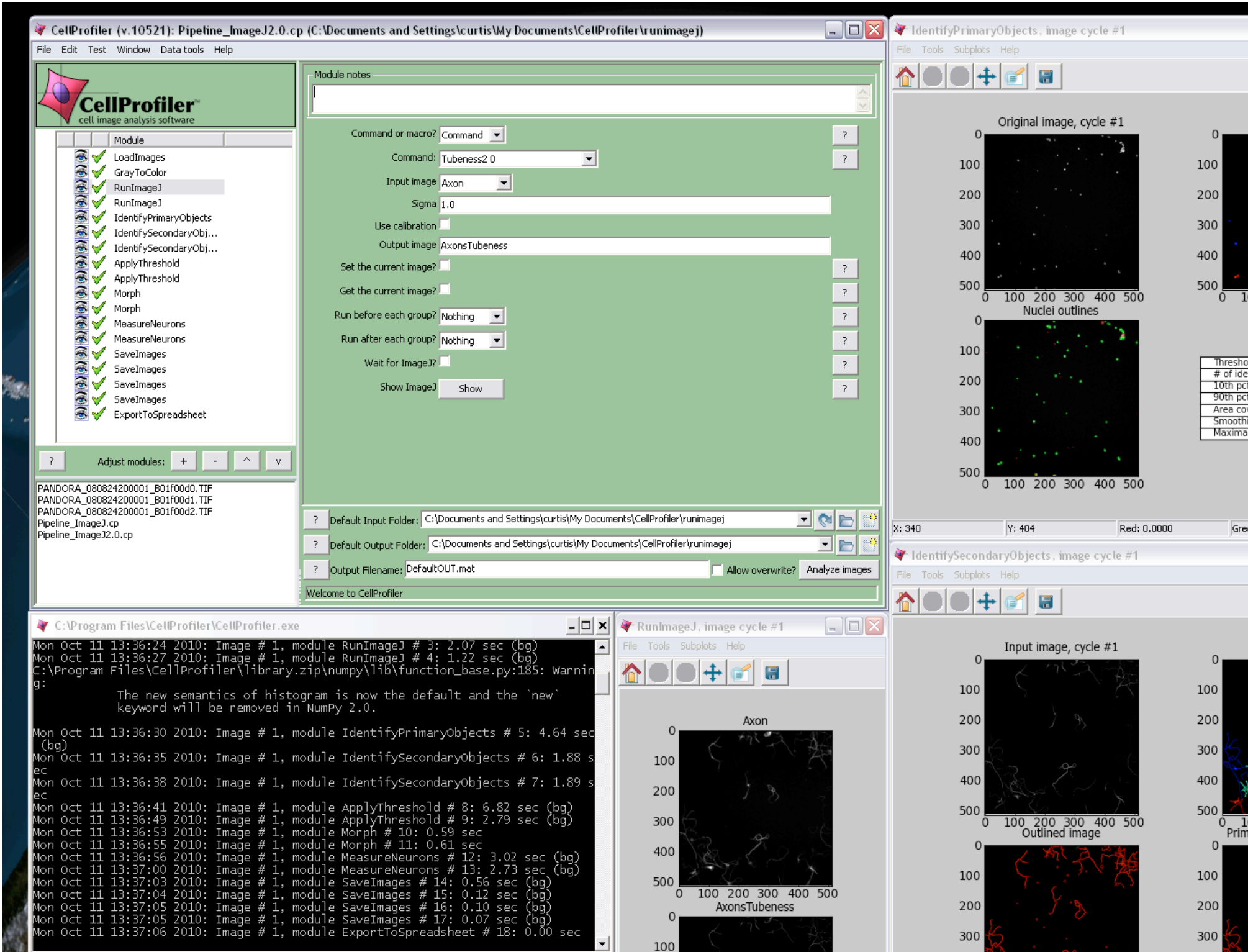
Progress: ROIs

- 2D planar ROIs only for now; more later
- Using JHotDraw (same as OMERO.insight)



Progress: CellProfiler

- CellProfiler is a tool for executing high-throughout image analysis pipelines
- Achieves better interoperability with ImageJ using the declarative plugin mechanism


CellProfiler (v.10521): Pipeline_ImageJ2.0.cp (C:\Documents and Settings\curtis\My Documents\CellProfiler\runimagej)
File Edit Test Window Data tools Help
CellProfiler
cell image analysis software
Module
LoadImages
GrayToColor
RunImageJ
RunImageJ
IdentifyPrimaryObjects
IdentifySecondaryObj...
IdentifySecondaryObj...
ApplyThreshold
ApplyThreshold
Morph
Morph
MeasureNeurons
MeasureNeurons
SaveImages
SaveImages
SaveImages
SaveImages
ExportToSpreadsheet
Adjust modules:
PANDORA_080824200001_B01f00d0.TIF
PANDORA_080824200001_B01f00d1.TIF
PANDORA_080824200001_B01f00d2.TIF
Pipeline_ImageJ.cp
Pipeline_ImageJ2.0.cp
Module notes
Command or macro? Command
Command: Tubeness2 0
Input image Axon
Sigma 1.0
Use calibration
Output image AxonsTubeness
Set the current image?
Get the current image?
Run before each group? Nothing
Run after each group? Nothing
Wait for ImageJ?
Show ImageJ Show
Default Input Folder: C:\Documents and Settings\curtis\My Documents\CellProfiler\runimagej
Default Output Folder: C:\Documents and Settings\curtis\My Documents\CellProfiler\runimagej
Output Filename: DefaultOUT.mat
Allow overwrite?
Analyze images
Welcome to CellProfiler
IdentifyPrimaryObjects, image cycle #1
File Tools Subplots Help
Original image, cycle #1
Nuclei outlines
X: 340
Y: 404
Red: 0.0000
IdentifySecondaryObjects, image cycle #1
File Tools Subplots Help
Input image, cycle #1
Outlined image
C:\Program Files\CellProfiler\CellProfiler.exe
Mon Oct 11 13:36:24 2010: Image # 1, module RunImageJ # 3: 2.07 sec (bg)
Mon Oct 11 13:36:27 2010: Image # 1, module RunImageJ # 4: 1.22 sec (bg)
C:\Program Files\CellProfiler\library.zip\numpy\lib\function_base.py:185: Warning:
The new semantics of histogram is now the default and the `new` keyword will be removed in NumPy 2.0.
Mon Oct 11 13:36:30 2010: Image # 1, module IdentifyPrimaryObjects # 5: 4.64 sec (bg)
Mon Oct 11 13:36:35 2010: Image # 1, module IdentifySecondaryObjects # 6: 1.88 sec
Mon Oct 11 13:36:38 2010: Image # 1, module IdentifySecondaryObjects # 7: 1.89 sec
Mon Oct 11 13:36:41 2010: Image # 1, module ApplyThreshold # 8: 6.82 sec (bg)
Mon Oct 11 13:36:49 2010: Image # 1, module ApplyThreshold # 9: 2.79 sec (bg)
Mon Oct 11 13:36:53 2010: Image # 1, module Morph # 10: 0.59 sec
Mon Oct 11 13:36:55 2010: Image # 1, module Morph # 11: 0.61 sec
Mon Oct 11 13:36:56 2010: Image # 1, module MeasureNeurons # 12: 3.02 sec (bg)
Mon Oct 11 13:37:00 2010: Image # 1, module MeasureNeurons # 13: 2.73 sec (bg)
Mon Oct 11 13:37:03 2010: Image # 1, module SaveImages # 14: 0.56 sec (bg)
Mon Oct 11 13:37:04 2010: Image # 1, module SaveImages # 15: 0.12 sec (bg)
Mon Oct 11 13:37:05 2010: Image # 1, module SaveImages # 16: 0.10 sec (bg)
Mon Oct 11 13:37:05 2010: Image # 1, module SaveImages # 17: 0.07 sec (bg)
Mon Oct 11 13:37:06 2010: Image # 1, module ExportToSpreadsheet # 18: 0.00 sec
RunImageJ, image cycle #1
File Tools Subplots Help
Axon
AxonsTubeness

Progress: Requirements

- Gathered feedback from the community
- Major areas of ImageJ
  - Data model & image processing
  - Visualization & user interface
  - Input & output
  - Segmentation & regions of interest
  - Scripting & plugins

Progress: Development Tools

- Web site
- Unit test suite
- Continuous integration: Hudson
- Source control: Subversion & Git
- Project management: Maven & Trac

Future Releases

- ImageJ v2.0.0-beta series due July-September
  - Focus on documentation and community feedback
- ImageJ v2.0.0-final is due on October 1
  - Finalized API for ImageJ v2.x
- For details, see:
  - http://dev.imagejdev.org/trac/imagej/roadmap

Benefits of ImageJ2

- What Will ImageJ 2.0 Do for Me?
  - Work with existing plugins and macros
  - Work with new, exciting plugins and scripts
  - Handle larger, more complex datasets
  - Multidimensional visualization tools
  - Easier to link with other software
  - Easier plugin management

ImageJ2 + OME

- OMERO is powerful server-side imaging software
- ImageJ is powerful client-side imaging software
- Both paradigms are valuable to scientists
- Many operations are common to client & server
  - E.g.: Bio-Formats is used for both
- Goal is to identify more areas for code sharing:
  - Big images (e.g., tiling with mipmaps)
  - Image rendering and thumbnails
  - Many others

Acknowledgements

- Principal Investigators
  - Kevin Eliceiri (LOCI), Rudolf Oldenbourg (MBL), Anne Carpenter (Broad)
- Developers
  - Grant Harris, Barry DeZonia, Aivar Grislis (ImageJDev)
  - Lee Kamentsky, Adam Fraser (CellProfiler)
- Collaborators
  - Wayne Rasband (ImageJ)
  - Pavel Tomancak, Johannes Schindelin, Albert Cardona (Fiji)
  - Stephan Preibisch, Stephan Saalfeld (ImgLib, Fiji)
  - Mark Longair, Jean-Yves Tinevez (Fiji)
  - Jason Swedlow, OMERO development team (OME)