LOCI, Fiji & ImageJ2

Philosophy of optical research

- *In vivo* developmental biology
- Capture everything possible about a sample
- Multiple dimensions
  - Emission spectra
  - Lifetime
  - Cell polarization
- Greater resolution and range
  - Adaptive optics (peer deeper into the soup)
  - Longer time series

Philosophy of software development

- Interoperability
  - Leverage existing tools
  - “Glue code”
  - Implement missing features
- Open source (federally funded—ethical obligation)
- Of benefit to LOCI's research
- Of benefit to the broader scientific community

Computational tools

- Acquisition software
  - WiscScan
  - Micro-Manager
- Visualization & analysis
  - ImageJ, Fiji
  - VisBio
  - Slim Plotter, TRI2
  - FARSIGHT, ITK
  - CellProfiler
- Data management
  - Open Microscopy Environment

A major goal of ImageJ2 is to integrate these tools into a common framework

LOCI's vision of scientific workflow

- ***Acquire*** data however you like (e.g., µManager)
  - If possible, record data in a standard format (OME-TIFF)
  - If not, convert from proprietary format (Bio-Formats)
- ***Store and organize*** data centrally (OMERO server)
- ***Analyze*** data through various client & desktop software
  - Smart clients can access data from the server (Fiji via plugins, OMERO.webclient, OMERO.insight)
  - Other software can work with data on disk (Fiji, etc.)
- ***Record changes*** back in the database, with versioning
- ***Share*** data over the Internet with colleagues
  - Both informally and as part of publication

What is Fiji?

- An easy way to develop novel algorithms
  - Plugins, macros or scripts
- A distribution platform for those algorithms
- A suite of tools to facilitate these goals
  - Script Editor (currently supports 7 languages)
  - Tutorial Maker
  - Fiji Updater
    - Access latest updates easily
    - Submit code directly to Fiji repository

Fiji vs. ImageJ

- Fiji is a *distribution* of ImageJ bundled with useful scientific image processing routines

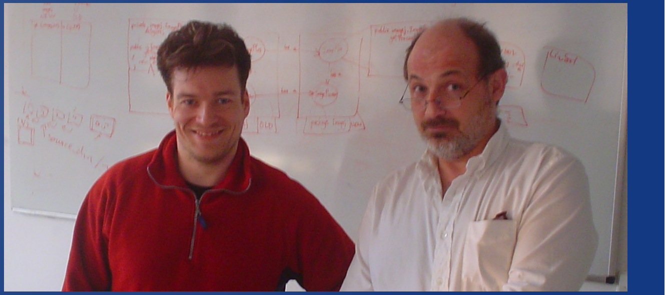
	ImageJ	**Fiji**
Size	~3-5 MB	~45 MB
Focus	Various image processing	Segmentation & registration
License	Public domain	GPL (various plugin licenses)
Tools	Minimal	Rich

- Fiji is the GNU/Linux distribution to ImageJ's Linux kernel
- “**F**iji **I**s **J**ust **I**mageJ—batteries included”
- “Some assembly required”

Origins of Fiji

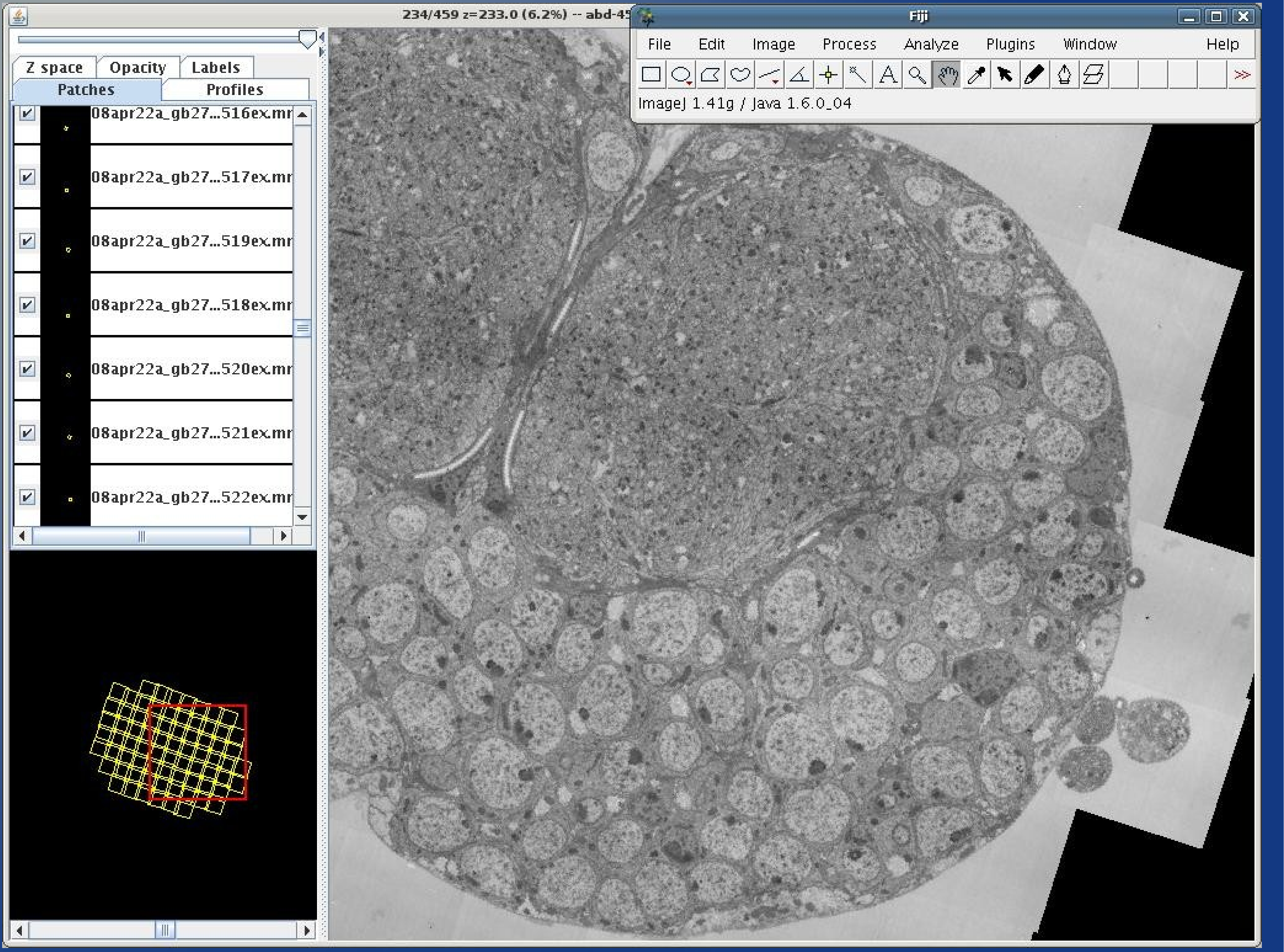
- Albert Cardona ("Father" of Fiji)

  - Institute of Neuroinformatics, Uni/ETH Zurich
  - Albert is a biologist who wrote TrakEM2
  - Began as a way to easily distribute TrakEM2
- Johannes Schindelin ("Mother" of Fiji)
  - Max Planck Institute, Cell Biology Division (MPI-CBG) in Dresden
  - Johannes is the one who did the hard work
  - Maintains the repository, build system, etc.



- Pavel Tomancak

  - Principal Investigator at MPI-CBG
  - Working to secure additional funding for Fiji



234/459 z=233.0 (6.2%) -- abd-45

ImageJ 1.41g / Java 1.6.0_04

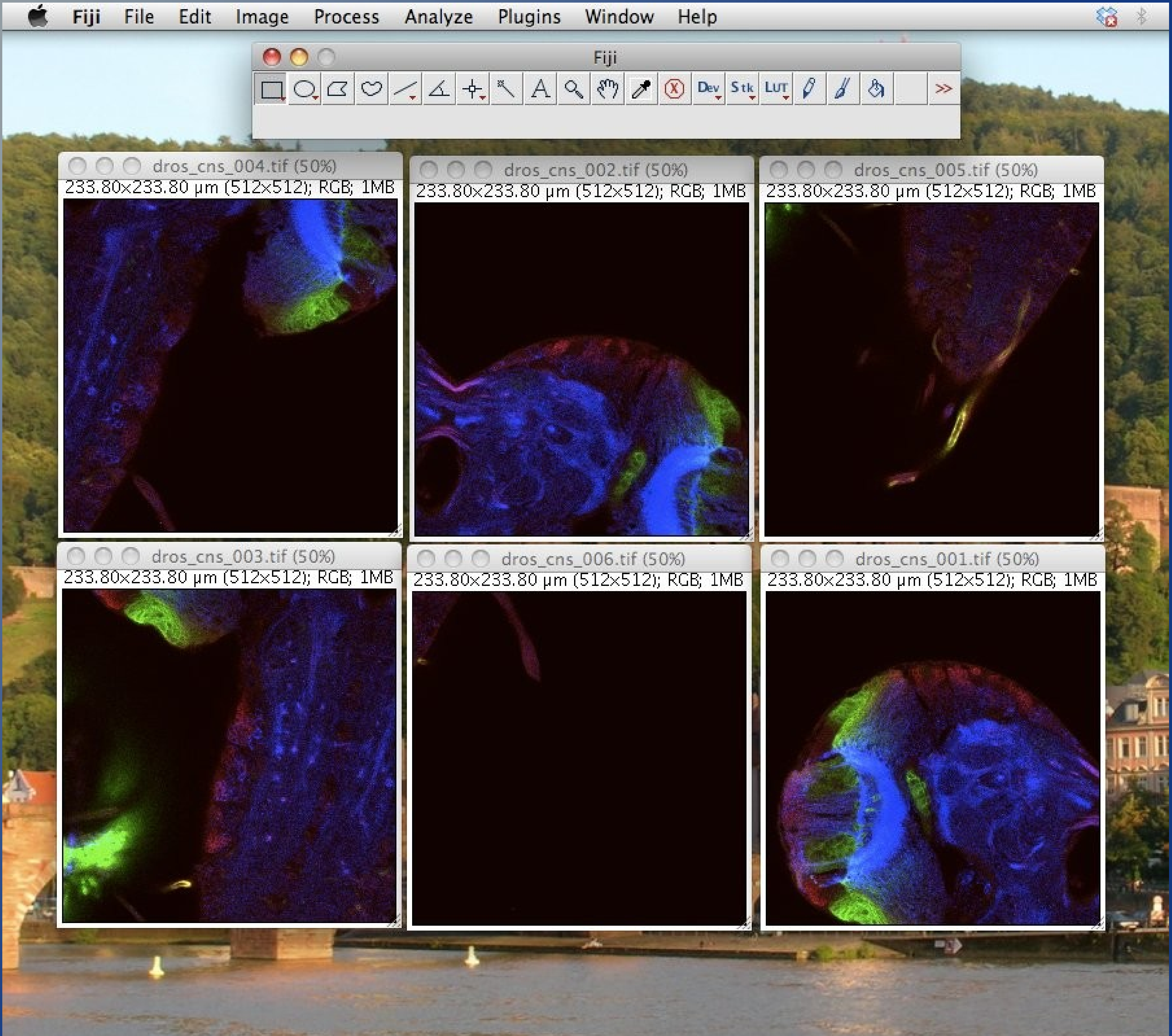
Z space
Opacity
Labels
Patches
Profiles
☑ 08apr22a_gb27...516ex.mr
☑ 08apr22a_gb27...517ex.mr
☑ 08apr22a_gb27...519ex.mr
☑ 08apr22a_gb27...518ex.mr
☑ 08apr22a_gb27...520ex.mr
☑ 08apr22a_gb27...521ex.mr
☑ 08apr22a_gb27...522ex.mr
Fiji
File
Edit
Image
Process
Analyze
Plugins
Window
Help
A
≫

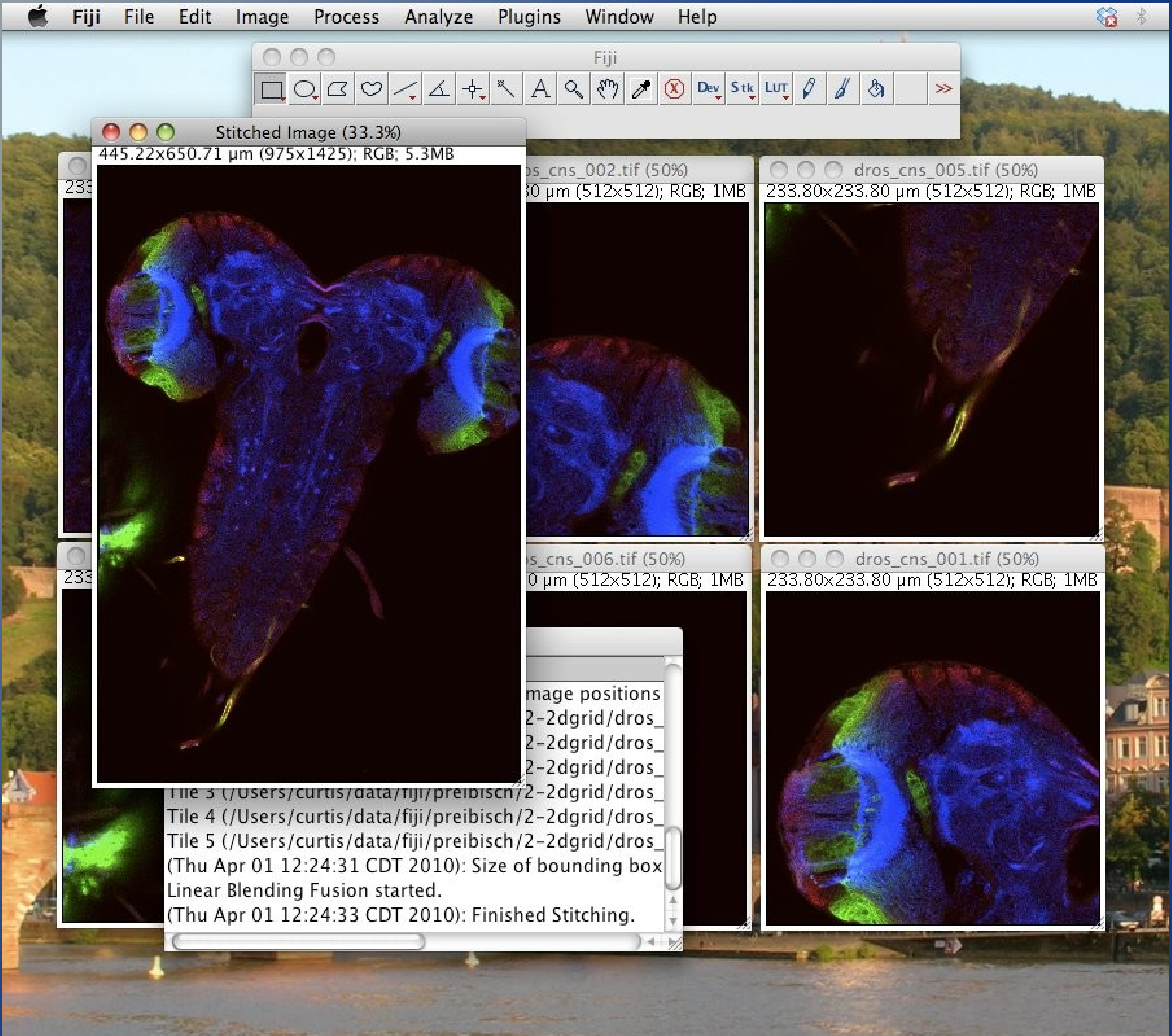
Other hackathon participants

- Stephan Saalfeld & Stephan Preibisch

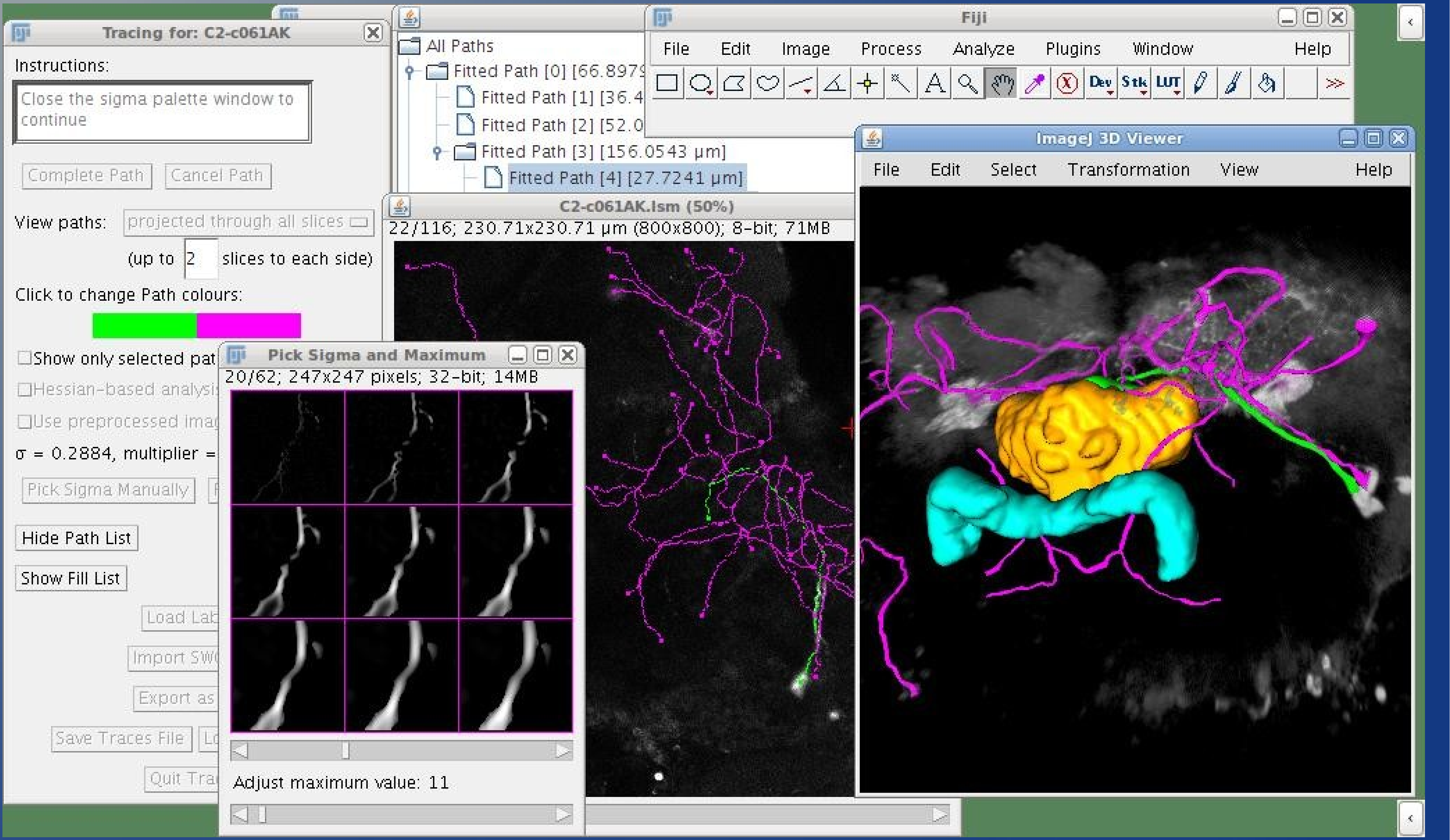
  - PhD students at MPI-CBG
  - Created excellent imglib library
  - Saalfeld developed much of TrakEM2's registration code
  - Preibisch developed Fiji's 2D & 3D Stitcher plugins
- Mark Longair

  - Just finished his PhD in neurobiology
  - Studies drosophila, specifically neuropils
  - Developed Fiji's Simple Neurite Tracer plugin
  - Moving from U. Edinburgh to Albert's group in Zurich


Fiji
File
Edit
Image
Process
Analyze
Plugins
Window
Help
Fiji
Dev
Stk
LUT
dros_cns_004.tif (50%)
233.80x233.80 µm (512x512); RGB; 1MB
dros_cns_002.tif (50%)
233.80x233.80 µm (512x512); RGB; 1MB
dros_cns_005.tif (50%)
233.80x233.80 µm (512x512); RGB; 1MB
dros_cns_003.tif (50%)
233.80x233.80 µm (512x512); RGB; 1MB
dros_cns_006.tif (50%)
233.80x233.80 µm (512x512); RGB; 1MB
dros_cns_001.tif (50%)
233.80x233.80 µm (512x512); RGB; 1MB



```
mage positions
2-2dgrid/dros_
2-2dgrid/dros_
2-2dgrid/dros_
Tile 3 (/Users/curtis/data/fiji/preibisch/2-2dgrid/dros_
Tile 4 (/Users/curtis/data/fiji/preibisch/2-2dgrid/dros_
Tile 5 (/Users/curtis/data/fiji/preibisch/2-2dgrid/dros_
(Thu Apr 01 12:24:31 CDT 2010): Size of bounding box
Linear Blending Fusion started.
(Thu Apr 01 12:24:33 CDT 2010): Finished Stitching.
```



Tracing for: C2-c061AK

Instructions:

Close the sigma palette window to continue

View paths: projected through all slices

(up to 2 slices to each side)

Click to change Path colours: 🟩🟪

□Show only selected pat

□Hessian-based analysi

□Use preprocessed imag

σ = 0.2884, multiplier =

All Paths
- Fitted Path [0] [66.8979
  - Fitted Path [1] [36.4
  - Fitted Path [2] [52.0
  - Fitted Path [3] [156.0543 µm]
    - Fitted Path [4] [27.7241 µm]

C2-c061AK.lsm (50%)

22/116; 230.71x230.71 µm (800x800); 8-bit; 71MB

Pick Sigma and Maximum

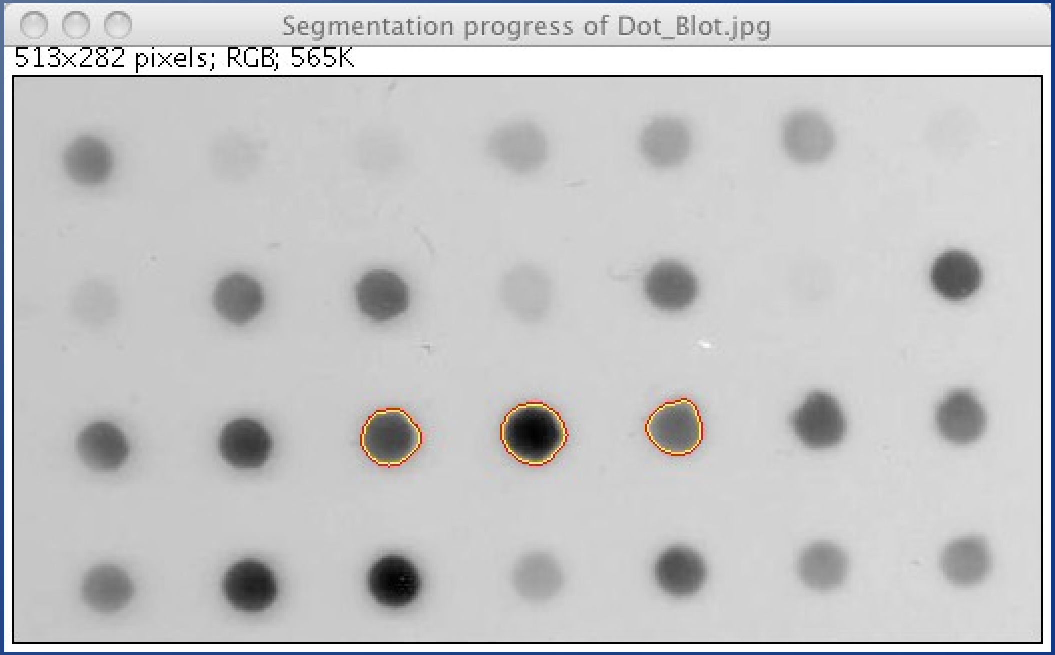
20/62; 247x247 pixels; 32-bit; 14MB

Adjust maximum value: 11

Other hackathon participants

- Bene Schmid
  - Developed Fiji's 3D Viewer (uses Java3D)
- Erwin Frise
  - Developed level sets plugin for image segmentation
- Larry Lindsey

  - U. Austin
  - Charged with maintaining a package called “Reconstruct”
  - Saw TrakEM2/Fiji could already do most of the same things
  - Spent his time learning about imglib



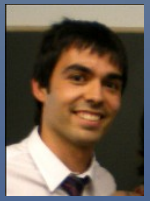
513x282 pixels; RGB; 565K

Other hackathon participants

- Verena Kaynig

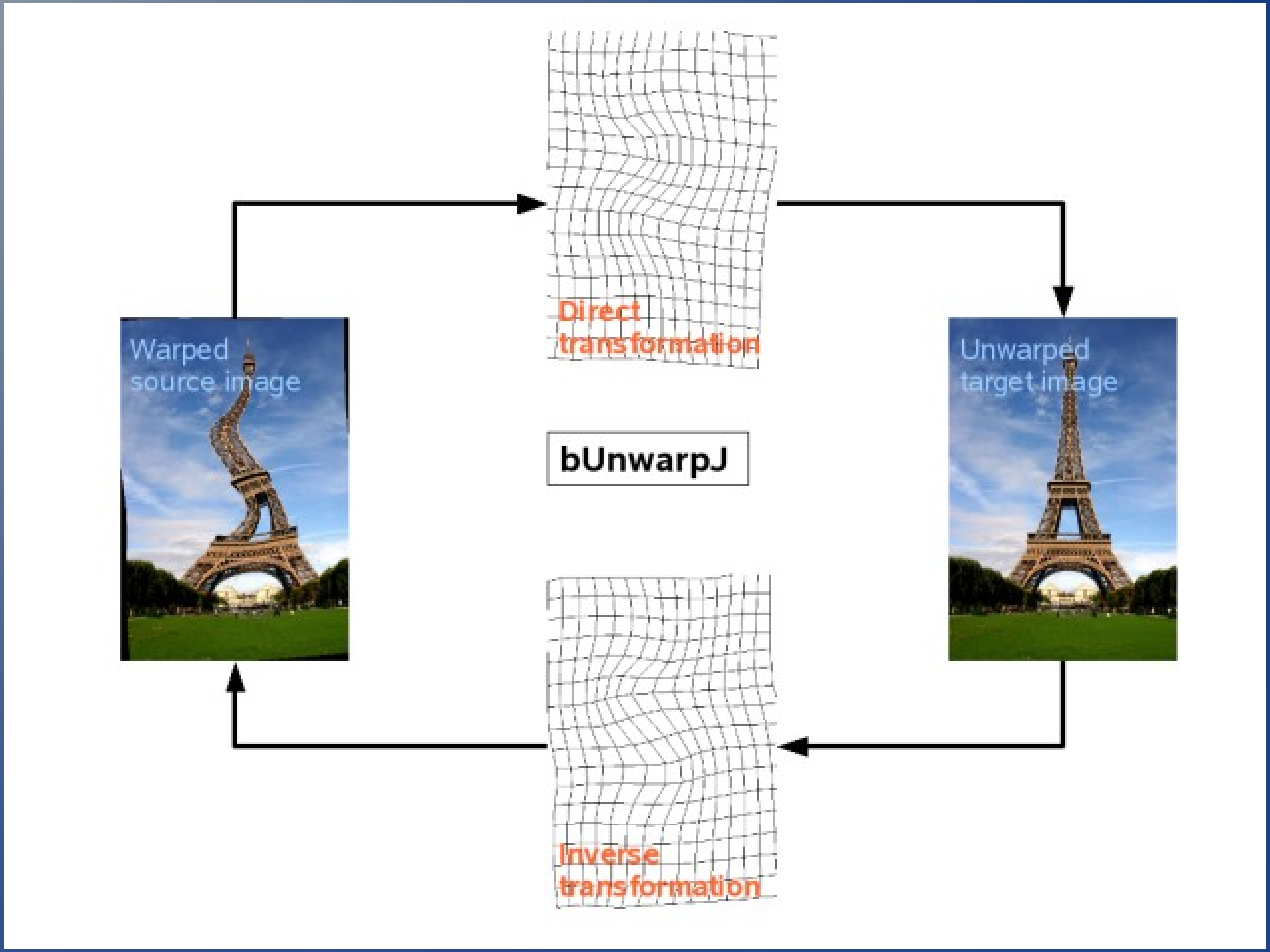
  - Computer scientist interested in machine learning
  - Wrote a “trainable segmentation” plugin

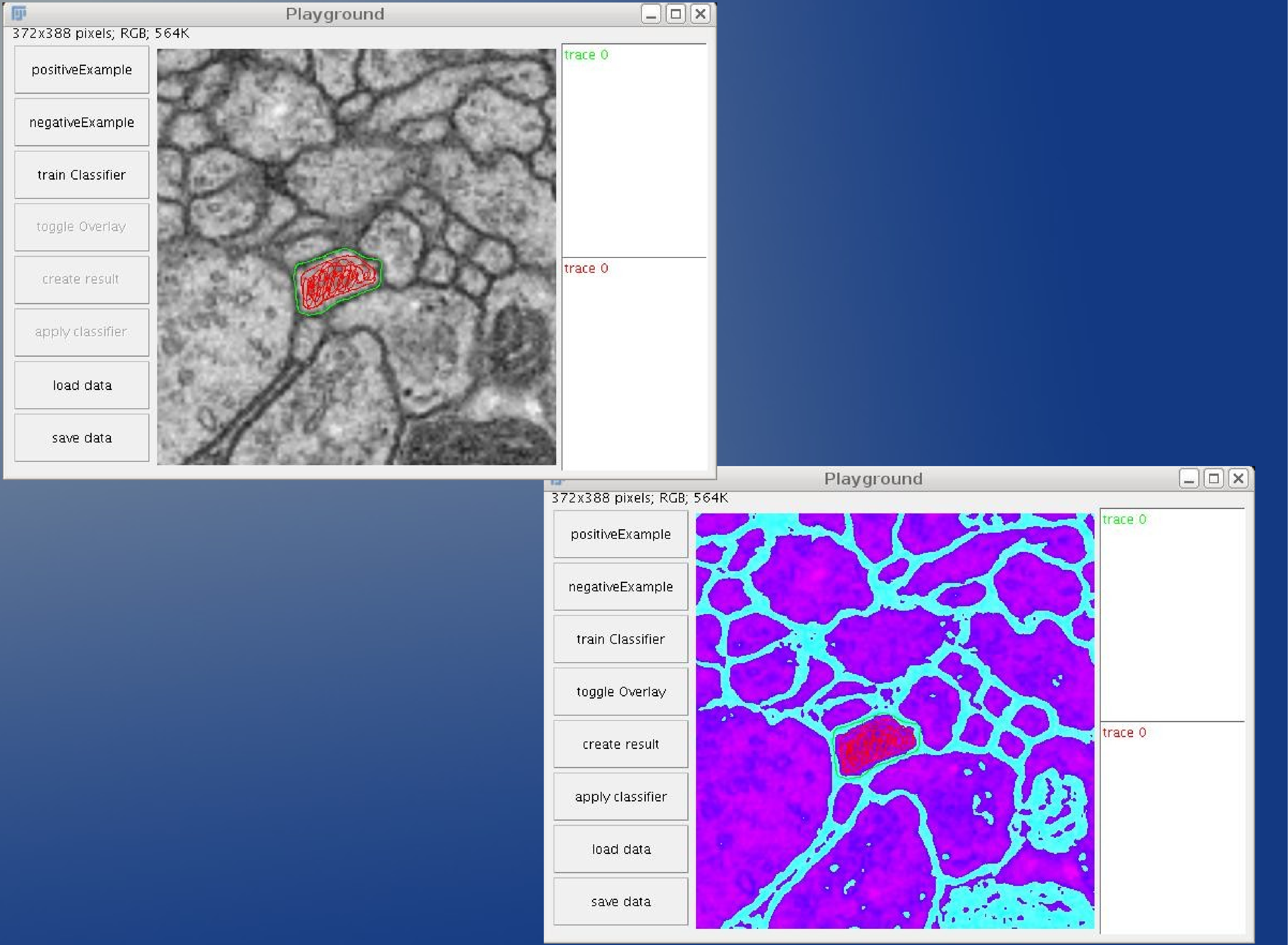
- Ignacio Arganda



  - Developed Fiji's bUnwarpJ plugin for image registration
  - Improved Verena's Trainable Segmentation plugin after she left

- Rubén Muñoz
  - EMBL programmer working with Olympus ScanR systems
  - Discussed Bio-Formats and OME
  - Developed his own database with web interface
  - Uses OME-TIFF


Direct transformation
Warped source image
bUnwarpJ
Unwarped target image
Inverse transformation

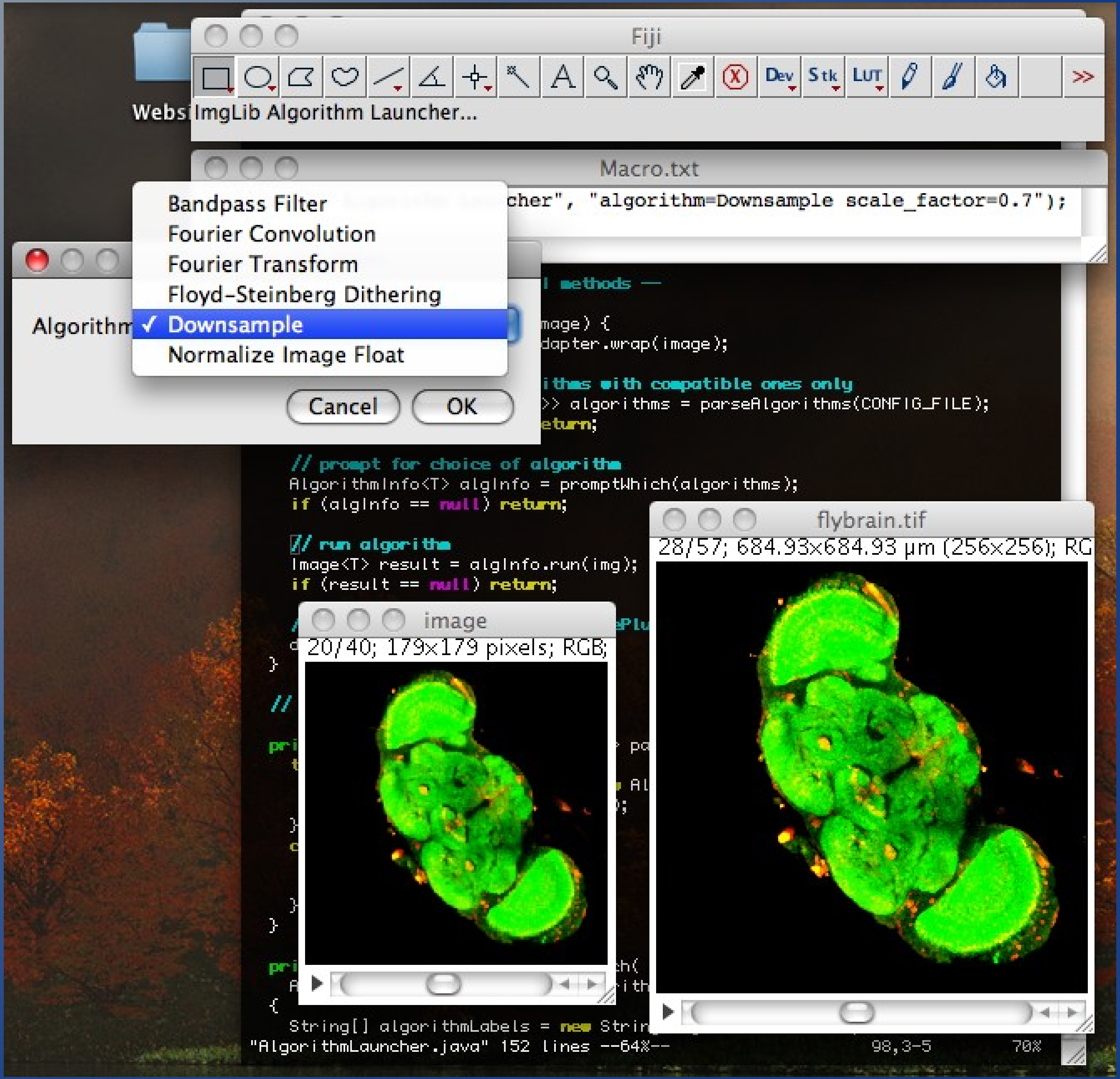

Playground
372x388 pixels; RGB; 564K
positiveExample
negativeExample
train Classifier
toggle Overlay
create result
apply classifier
load data
save data
trace 0
trace 0
Playground
372x388 pixels; RGB; 564K
positiveExample
negativeExample
train Classifier
toggle Overlay
create result
apply classifier
load data
save data
trace 0
trace 0

Other Fiji developers

- Jean-Yves Tinevez
  - Institut Pasteur
- Gabriel Landini
  - Prof. Of Analytical Pathology
  - School of Dentistry, U. Birmingham, England
- Us!
- Probably others I'm forgetting

What I worked on

- Learned Fiji development workflow
- Algorithm launcher for imglib
    - Execute any imglib OutputAlgorithm in Fiji
- Autogenerated an ImageJ delegation layer
    - Attempt failed; now using a simpler approach
- Fiji/ITK integration
    - Proof of concept; call ITK image filters from Fiji
- Some work on Bio-Formats
    - Same dependency injection pains as us

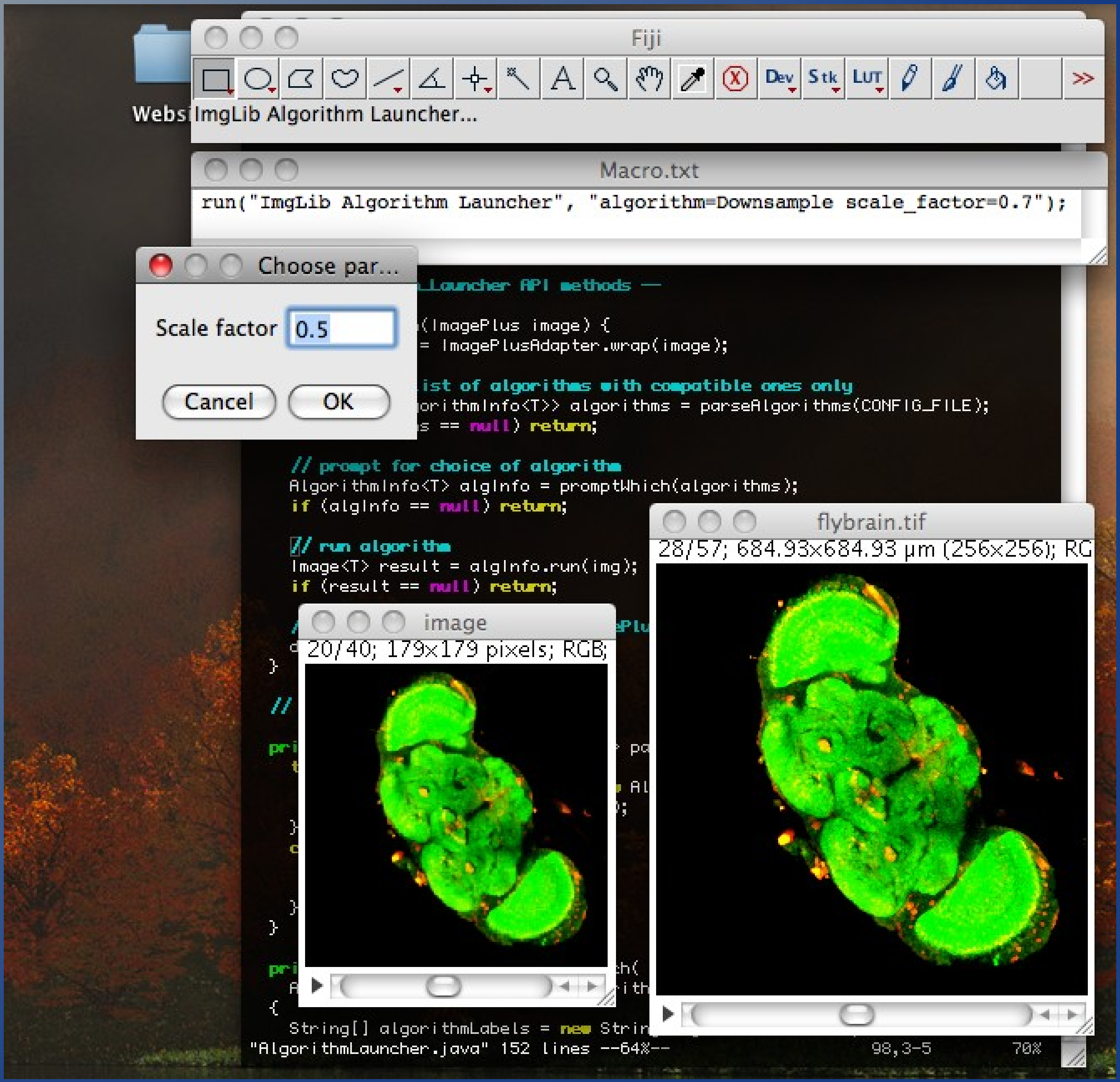


```
cher", "algorithm=Downsample scale_factor=0.7");
```

```
l methods --

mage) {
dapter.wrap(image);

ithms with compatible ones only
>> algorithms = parseAlgorithms(CONFIG_FILE);
eturn;

    // prompt for choice of algorithm
    AlgorithmInfo<T> algInfo = promptWhich(algorithms);
    if (algInfo == null) return;

    // run algorithm
    Image<T> result = algInfo.run(img);
    if (result == null) return;

    String[] algorithmLabels = new Strin
"AlgorithmLauncher.java" 152 lines --64%--               98,3-5        70%
```

20/40; 179x179 pixels; RGB;

28/57; 684.93x684.93 µm (256x256); RG



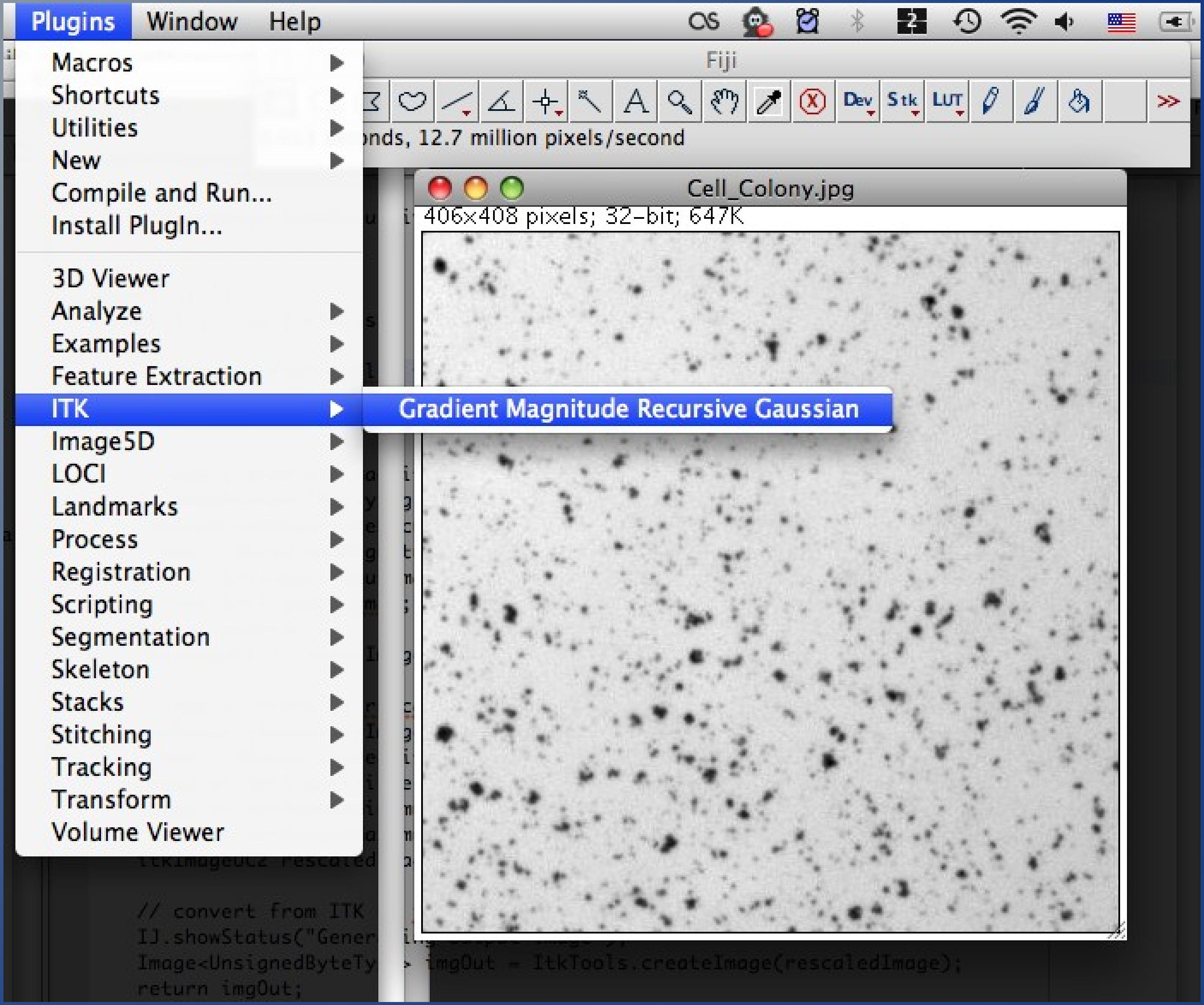
```
run("ImgLib Algorithm Launcher", "algorithm=Downsample scale_factor=0.7");
```

Scale factor 0.5

```
 _Launcher API methods --

 (ImagePlus image) {
 = ImagePlusAdapter.wrap(image);

 ist of algorithms with compatible ones only
 orithmInfo<T>> algorithms = parseAlgorithms(CONFIG_FILE);
 s == null) return;

    // prompt for choice of algorithm
    AlgorithmInfo<T> algInfo = promptWhich(algorithms);
    if (algInfo == null) return;

    // run algorithm
    Image<T> result = algInfo.run(img);
    if (result == null) return;

    /                                        ePlu
    c
  }

  //

  pri                                        > pa
    t
                                             AL
                                             );

    }
    c

    }
  }

  pri                                        h(
    A                                        ith
  {
    String[] algorithmLabels = new Strin
"AlgorithmLauncher.java" 152 lines --64%--                98,3-5        70%
```

28/57; 684.93x684.93 µm (256x256); RG

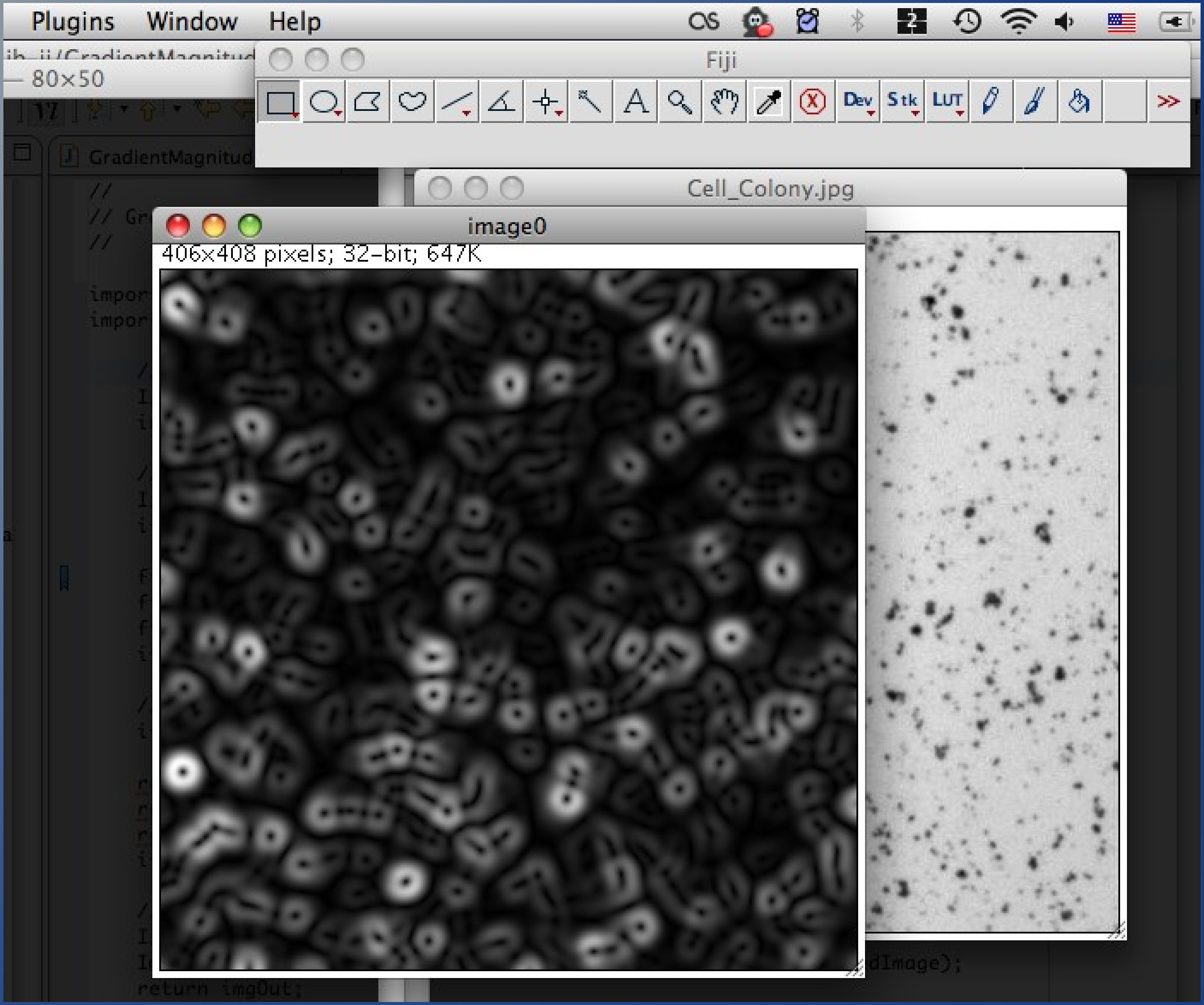
20/40; 179x179 pixels; RGB;



…nds, 12.7 million pixels/second

406x408 pixels; 32-bit; 647K

```
// convert from ITK
IJ.showStatus("Gener
Image<UnsignedByteTy    > imgOut = ItkTools.createImage(rescaledImage);
return imgOut;
```


Plugins
Window
Help
80×50
GradientMagnitud
Fiji
Dev
Stk
LUT
Cell_Colony.jpg
image0
406x408 pixels; 32-bit; 647K
return imgOut;

Reminder: Vision of Fiji

- An easy way to develop novel algorithms
  - Plugins, macros or scripts
- A distribution platform for those algorithms
- A suite of tools to facilitate these goals
  - Script Editor (currently supports 7 languages)
  - Tutorial Maker
  - Fiji Updater
    - Access latest updates easily
    - Submit code directly to Fiji repository

Vision of ImageJ2

- Core image processing library for Fiji
- Major goals:
    - Improve the technical design
    - Integration and interoperability
    - Grow the ImageJ community

Vision of ImageJ2

- Improve the technical design
  - A solid base on which to build scientific tools
  - Flexible enough to interface with a variety of existing software
  - Able to represent scientific image data in a variety of contexts (e.g., N-dimensional)
  - Buzzwords like “modular” and “extensible”

Vision of ImageJ2

- Integration and interoperability
  - Adapt Slim Plotter & VisBio features into ImageJ2
  - Glue code between ImageJ2 and CellProfiler
  - Glue code between ImageJ2, ITK and FARSIGHT

Vision of ImageJ2

- Grow the ImageJ community
  - Backwards compatible with existing plugins as much as possible (at least initially)
  - The more users of ImageJ, the easier it will be to share analysis routines and results
  - Fiji certainly shares this goal

Vision: Fiji vs. ImageJ2

ImageJ2	Fiji
Software development infrastructure	Image processing algorithm development and deployment
General image processing	Scientific image processing, particularly life sciences (microscopy, neuroscience, but also computer vision)
Boring baseline stuff	Fun domain-specific applications

- Where ImageJ2 ends and Fiji begins is fluid
- We may develop analysis routines useful to LOCI that end up in Fiji rather than ImageJ2

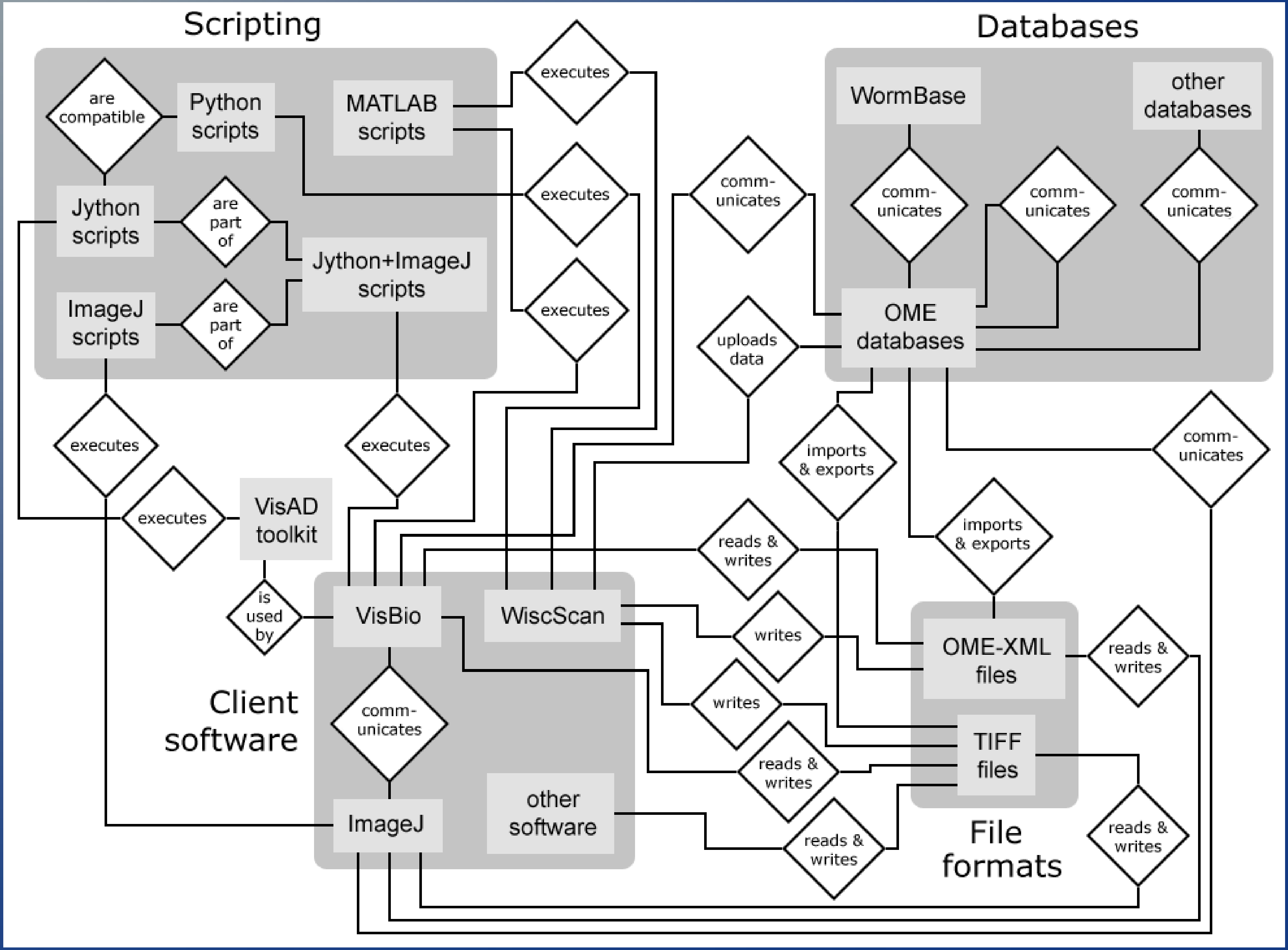
Components of ImageJ2

- Major components of ImageJ2
  1) **Data model** – imglib library
  2) **Display** – Java AWT, JAI, Swing
  3) **Input/output** – Bio-Formats architecture
  4) **Regions of interest (ROIs)** – Java AWT, JHotDraw
  5) **Scripting & plugins** – Java 6 Scripting Framework
- All of these areas have significant limitations and will benefit from enhancements and refactoring

Components of ImageJ2

- To start, each of us focus on one?
  1) **Data model** – Aivar?
  2) **Display** – Rick?
  3) **Input/output** – Barry?
  4) **Regions of interest (ROIs)** – Brian?
  5) **Scripting & plugins** – Grant?

An old MIDIAS diagram, just for fun... :-)


Scripting
Databases
are compatible
Python scripts
MATLAB scripts
executes
Jython scripts
are part of
Jython+ImageJ scripts
executes
ImageJ scripts
are part of
executes
WormBase
other databases
comm-unicates
comm-unicates
comm-unicates
comm-unicates
OME databases
uploads data
executes
executes
executes
imports & exports
comm-unicates
VisAD toolkit
reads & writes
imports & exports
is used by
VisBio
WiscScan
writes
OME-XML files
reads & writes
comm-unicates
writes
Client software
reads & writes
TIFF files
ImageJ
other software
reads & writes
File formats
reads & writes