ImageJ

- Install
- Operations with GUI
- Writing Macros

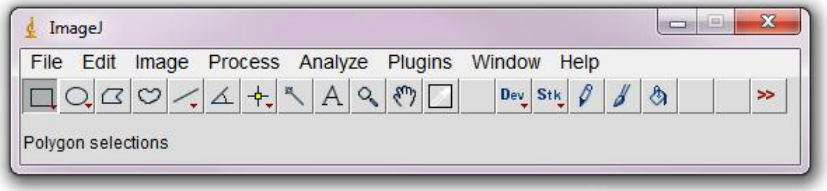
About ImageJ

- Open source image processing
- Developed at National Institutes of Health
- Java-based
- Extensible with plugins and macros

http://rsbweb.nih.gov/ij/

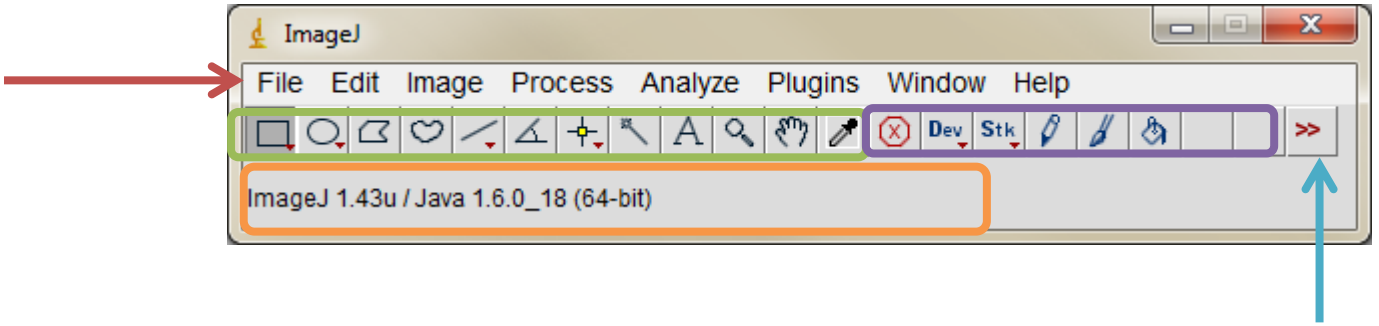
Installing ImageJ

- Already installed in lab computers (check?)
- Requires Java runtime
  - If you already have Java, simplest to use the platform independent .zip file



ImagJ GUI

- Menu
- Basic Tools
- Special Tools
- Special Tool Set Selector
- Status Bar

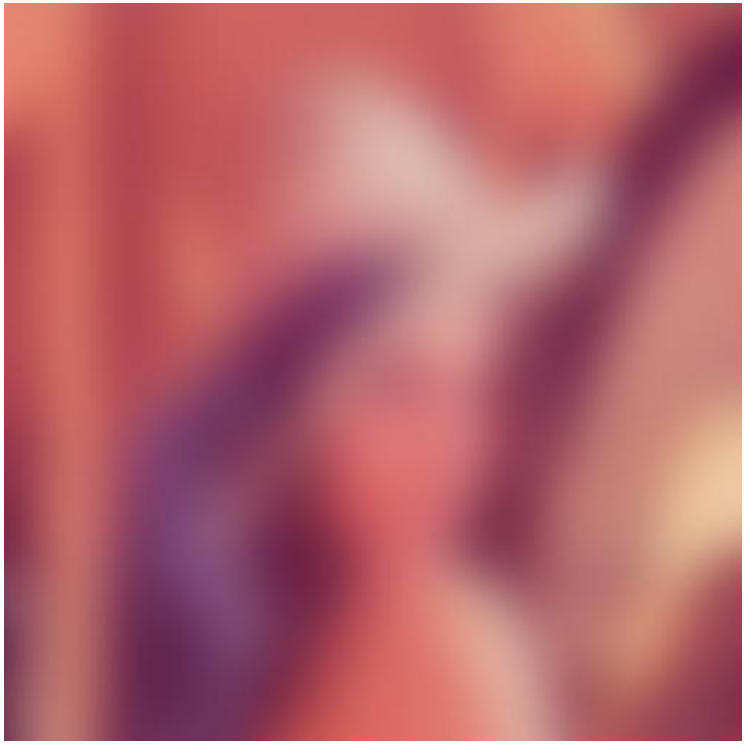


Basics

- Open File:
  - File > Open
  - Drag & Drop file to GUI
- Save File:
  - File > Save As > [FORMAT] // recommend PNG format
  - File > Save // default format is TIFF
- Zoom:
  - CTRL + + or NumPadPlus
  - CTRL + - or NumPadMinus
- Pan:
  - Hold Space and drag with LMB

Operations

- Process > Filters > Gaussian Blur…



Histograms

- Analyse > Histogram

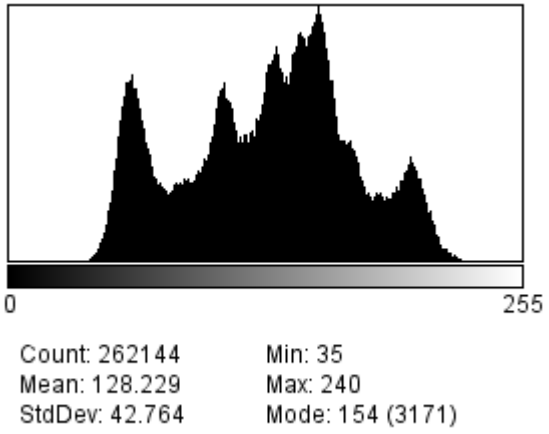
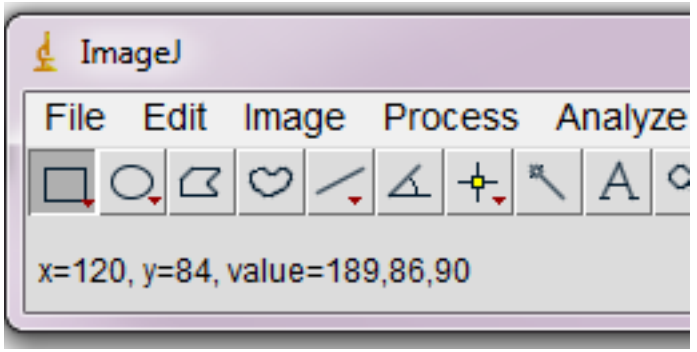


Image Types

- Greyscale
  - 8-bit, 16-bit*, 32-bit*
- Colour
  - 8-bit RGB(A), indexed*
- Stacks
  - Multiple images with same properties

* Does not support full processing with built-in tools

Image Info

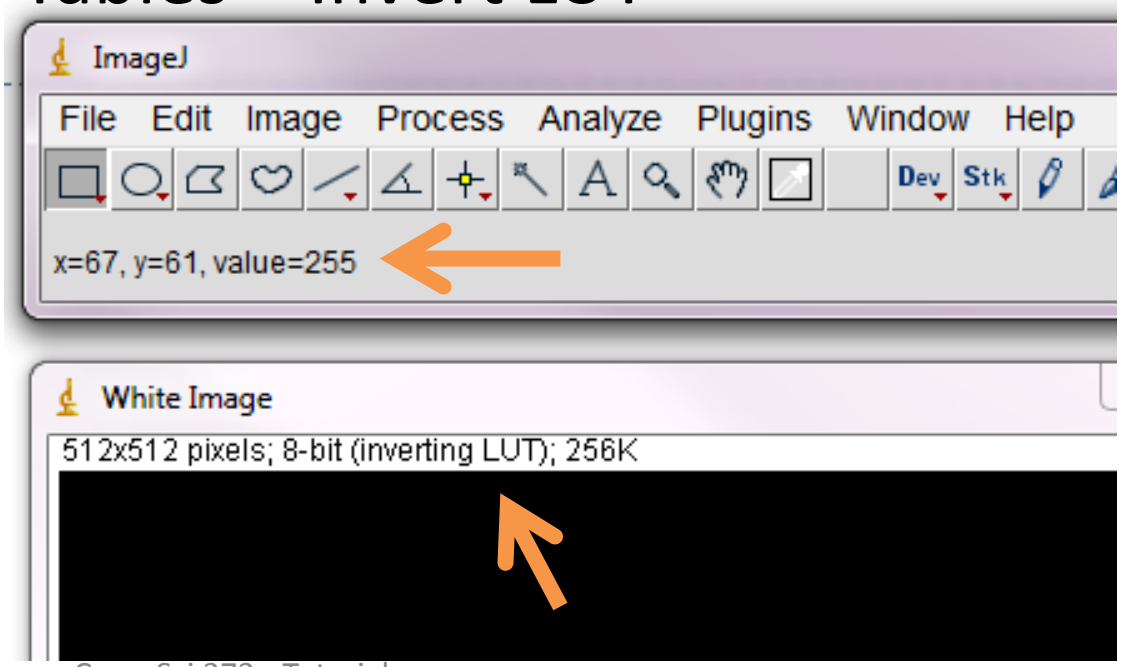


Mouse cursor position and pixel value

Image size; type; memory

ImageJ LUTs

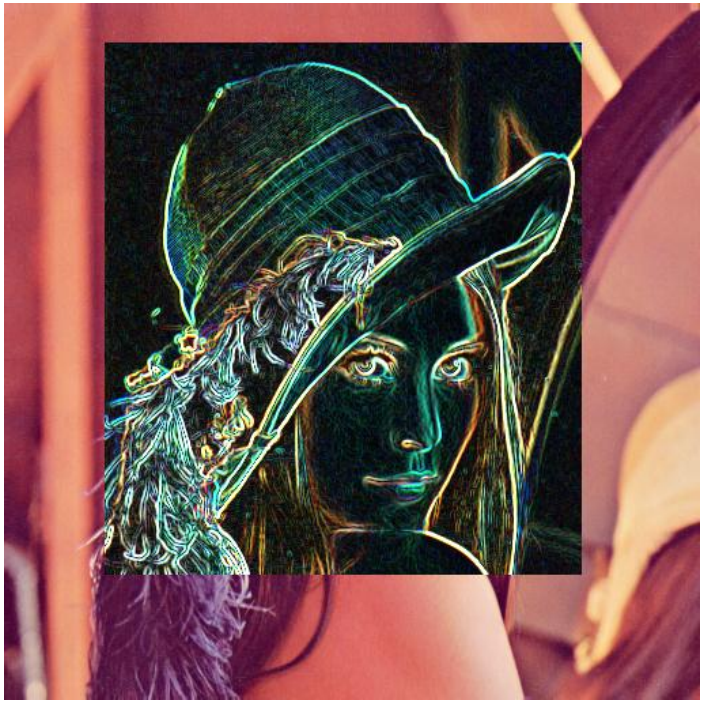
- Be ware of LUTs in ImageJ
  - LUT affects image rendering, not underlying data
- Remove inverting LUT with
  - Image > Lookup Tables > Invert LUT



Macros

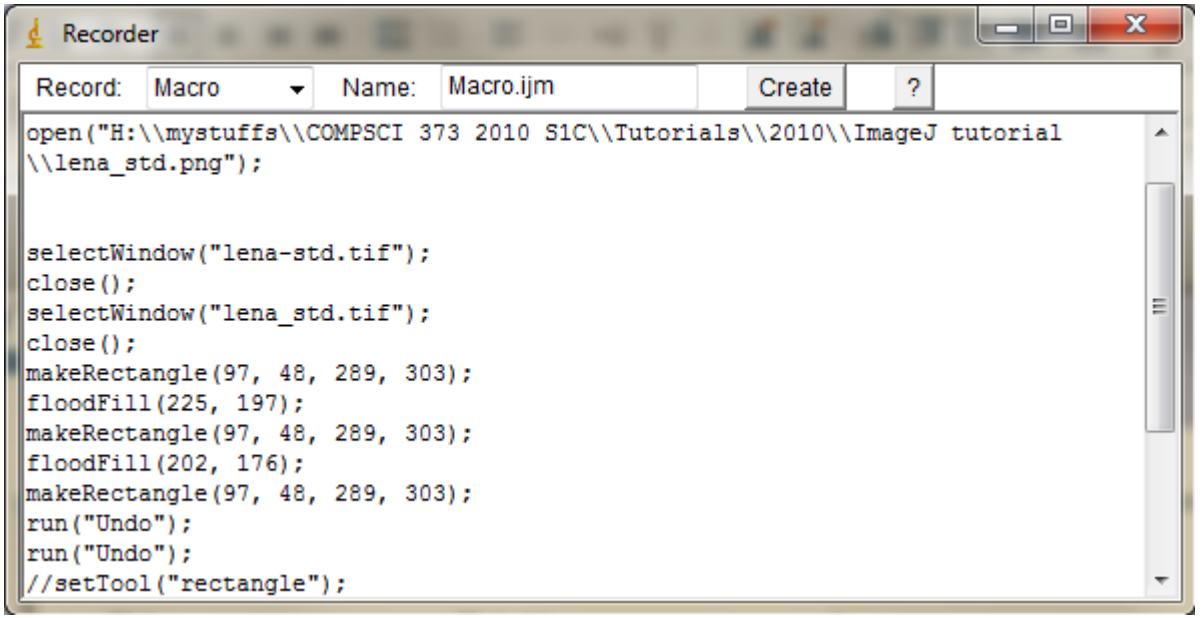
- Macro is a list of recorded commands in text
- File > New > Text Window

```
open("H:\\directory\\lena_std.png");
makeRectangle(74, 29, 351, 391);
run("Find Edges");
```



Macro Recorder

- Plugins > Macros > Record…
- Shows your GUI commands as macros



Executing Macros

- Plugins > Macros > Run…
  - Runs a stored text file as macro
- In Text Window: Macros > Run Macro
  - CTRL + R

ImageJ Macro Language

- Similar to JavaScript
- No variable declaration needed
- Allows simple control flows
- For a list of available built-in functions
  - http://rsbweb.nih.gov/ij/developer/macro/functions.html

Important Functions

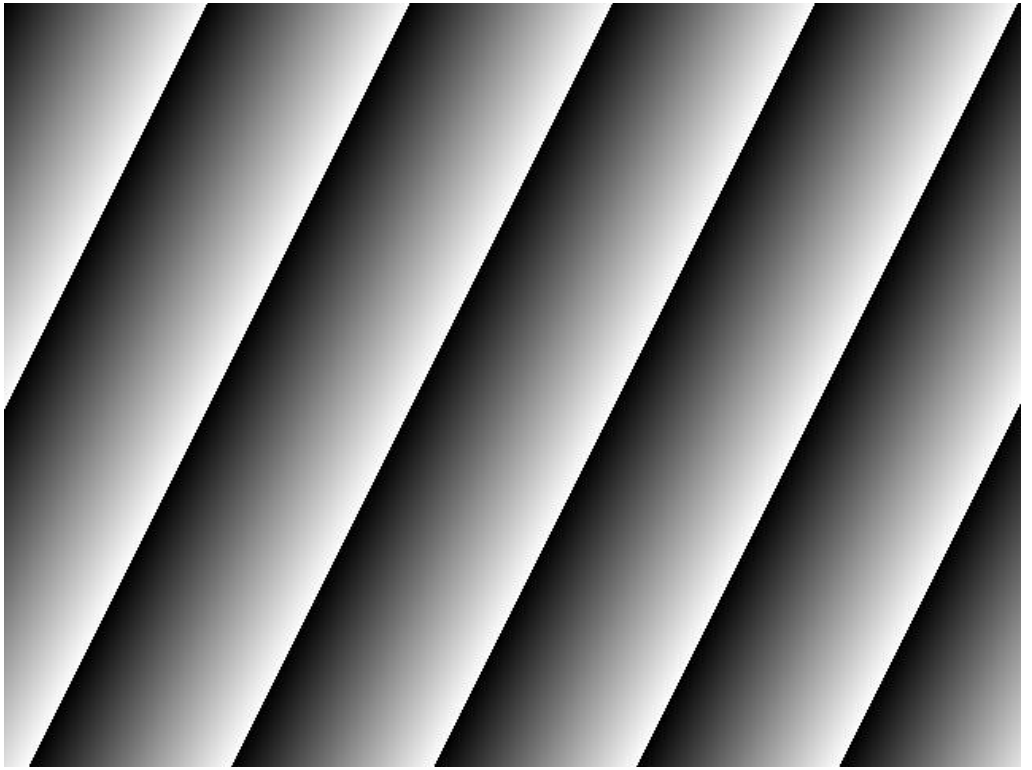
- Pixel access
  - getPixel(x, y); setPixel(x, y, value);
- Get image properties
  - getWidth(); getHeight(); bitDepth();
- Maths
  - sin(a); cos(a); tan(a); floor(n); round(n);
- Text output
  - print(string);

Active Image

- ImageJ only operates on the active image (selected image)
- Use functions to select different images
  - getImageID();
  - selectImage(id);

Macro Example

```
width = 640;
height = 480;
// 8-bit single channel image
newImage("Test Gray Image", "8-bit", width, height, 1);
for (y=0; y<height; y++) {
  for (x=0; x<width; x++) {
    v = (2*x+y) % 255;
    setPixel(x, y, v);
  }
}
updateDisplay();
```



Colour Access

- RGB pixel data stored in 32-bit integer
  - ARGB format
- Use shifting and masking to get individual channels

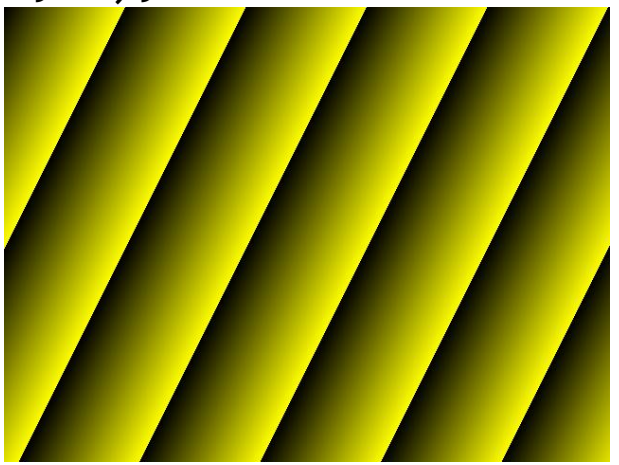
```
v = getPixel(x, y);
red = (v>>16) & 0xFF;  // extract red byte (bits 23-17)
green = (v>>8) & 0xFF; // extract green byte (bits 15-8)
blue = v & 0xFF;       // extract blue byte (bits 7-0)
```

Macro Example

```
width = 640;
height = 480;
// 8-bit per channel RGB image
newImage("Test RGB Image", "RGB", width, height, 1);
for (y=0; y<height; y++) {
   for (x=0; x<width; x++) {
        v = (2*x+y) % 255;

        red = (v<<16);
        green = (v<<8);
        blue = 0;

        setPixel(x, y, red | green | blue); // "bitwise or" operator,
   also works with red+green+blue
   }
}
updateDisplay();
```



Rounding

- Remember to apply rounding before storing integer pixel value
  - Variables internally uses best fit representation
    - floating points = double, integers = int
  - Runtime automatically truncates floating points to integers, gives slightly incorrect result
  - setPixel(x, y, round(v)); // for greyscale

Arrays

- Manipulate with functions
- No 2D array, use manual indexing

```
myArray = newArray(256);
for (i = 0; i < 256; i++) {
    myArray[i] = i % 30;
}
```

Macro Example

```
width = getWidth();
height = getHeight();

histo = newArray(256);

for (y = 0; y < height; y++) {
    for (x = 0; x < width; x++) {
        v = getPixel(x, y);
        histo[v]++;
    }
}

for (i = 0; i < 256; i++) {
    print(i, histo[i], "\n"); // equivalent to print(i + " " + histo[i] + " " +
"\n");
}
```

Other Examples

- For more explore the built-in macros in
  <ImageJ>\macros
  <ImageJ>\plugins\Examples\-Macros