CSC 470 – Section 3

Topics in Computer Science: Advanced Browser Technologies

Mark F. Russo, Ph.D.

Spring 2016

Lecture 12: Image Processing with Web Workers

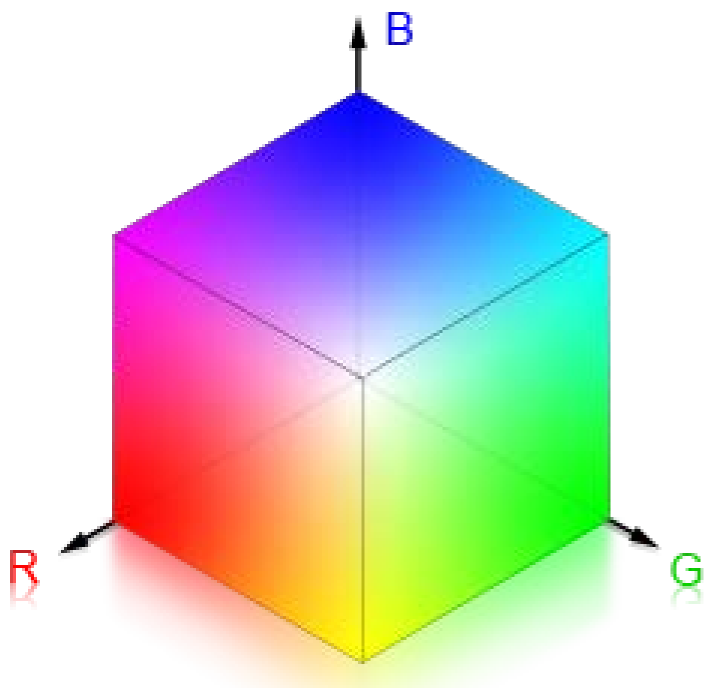
Color Models

Pixel colors are represented using different models

- Popular color models decompose a single color into three orthogonal components
- Dimensionality of the "color space" varies depending upon the model used

Popular Color Models Include:

- RGB – Red, Green, Blue
- HSV – Hue, Saturation, Value
- HSL – Hue, Saturation, Lightness



http://www.nec-display-solutions.com/p/sv/en/colourManagement.xhtml

Color Models - RGB

Most popular is Red-Green-Blue (RGB)

Cartesian coordinate system

Each of three dimensions represents a pure color

- Red, Green, and Blue

The value of each dimension is stored in one byte

- How large of a value can each dimension take on?

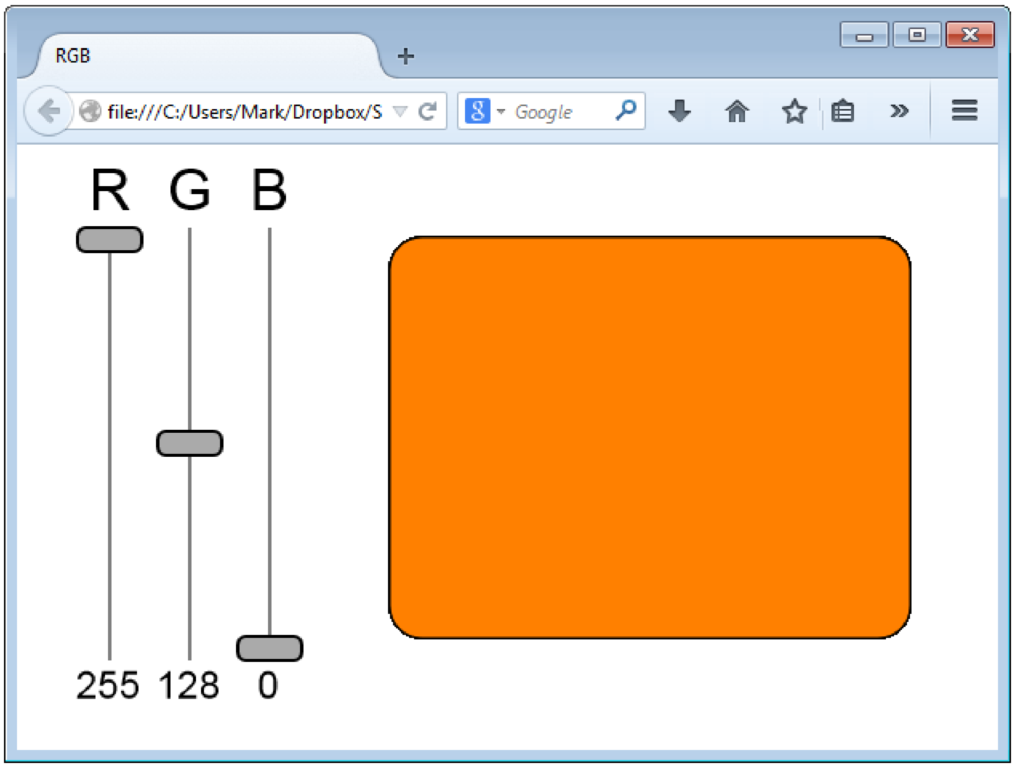
Origin (0, 0, 0) is black

On-axis points are shades of pure colors

Off-axis points represent other colors

(255, 255, 255) is white

Color Models - RGB



Find

- pure red
- pure blue
- pure green
- black
- white
- gray
- yellow
- magenta
- cyan

12/rgb/rgb.html

Color Models - HSV

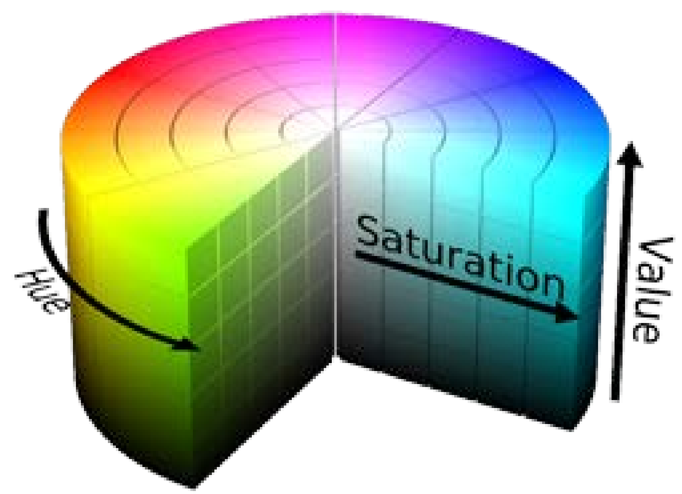
Hue-Saturation-Value (HSV)

- Cylindrical coordinate system
- **Hue** - Pure color value from rainbow (angle)
- **Saturation** - Grayscale to pure color (radius)
- **Value** - Black to color (height)

Useful when intensity or hue is required independent of the other

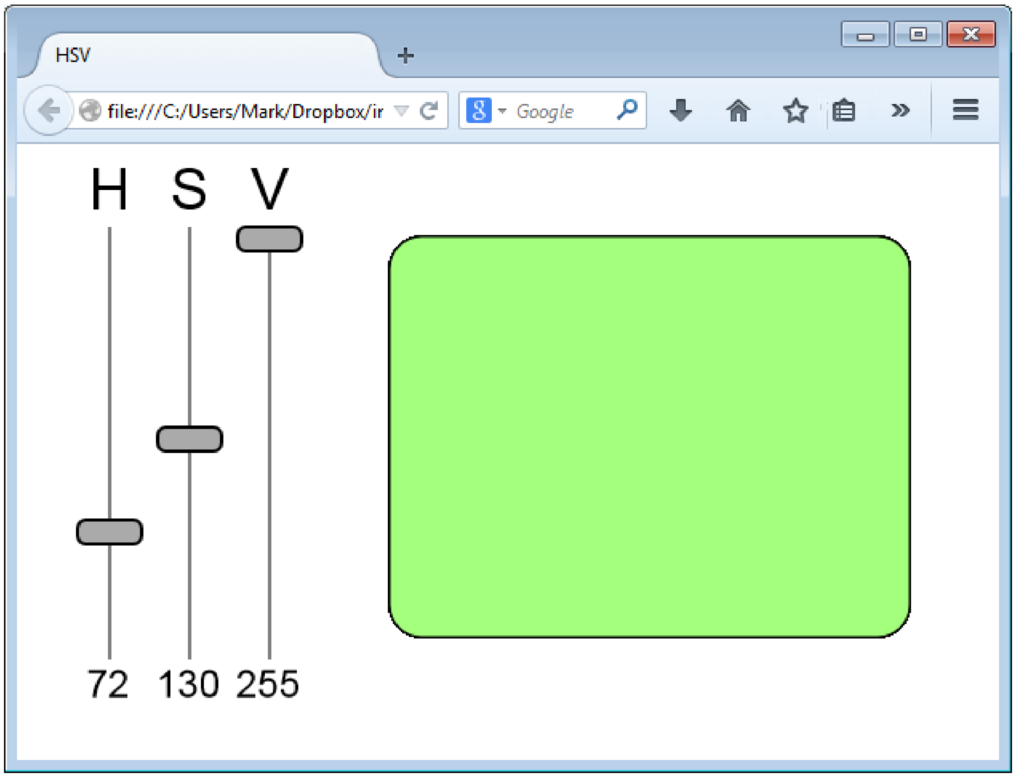
- Eg. Color contour plots

Black:	(any, any, 0)
White:	(any, 0, 255)
Grayscale:	(any, 0, any)



http://chroma.readthedocs.org/en/latest/

Color Models - HSV

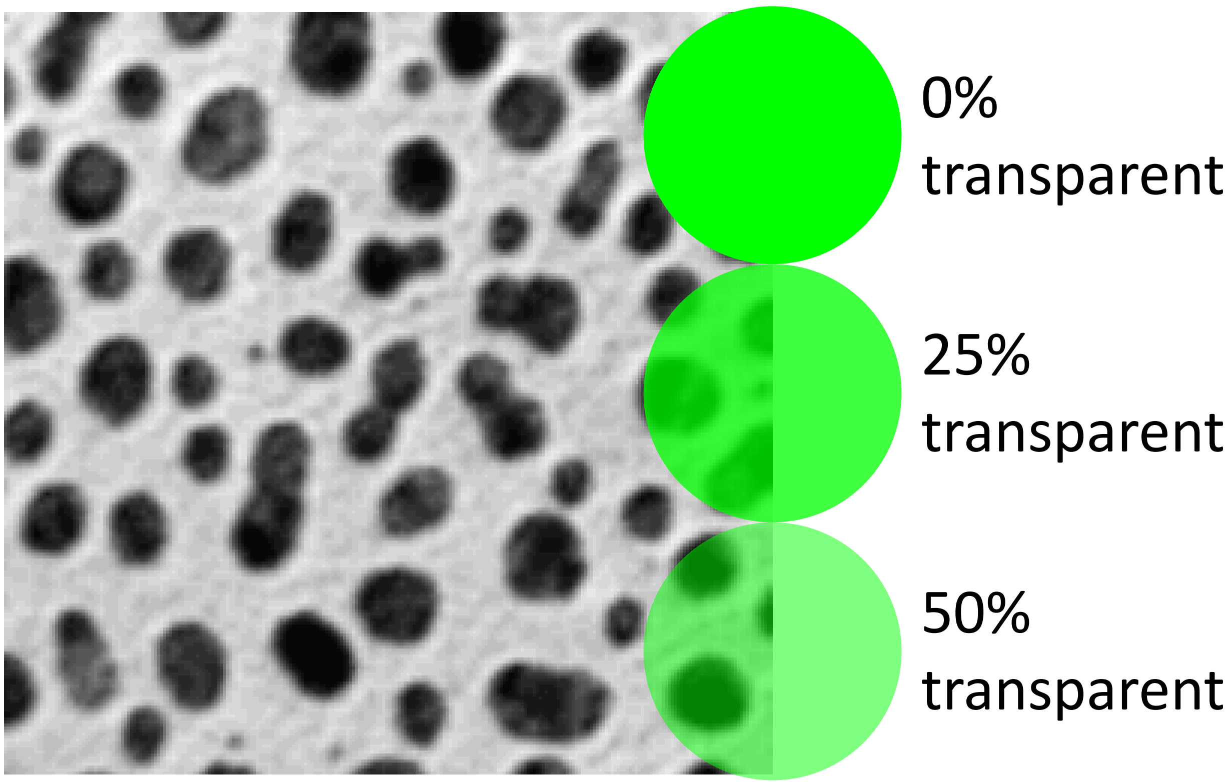


Find

- pure red
- pure blue
- pure green
- black
- white
- gray
- yellow
- magenta
- cyan

Transparency

- The degree to which a pixel allows the color of the pixel "below it" to bleed through



32-bit RGBA Color Model

- Adds a fourth value (called alpha) to a pixel color triple
  - alpha = 255 fully opaque
  - alpha = 0 fully transparent
- If three pixel color values and alpha are each stored in one byte, a pixel will be 32-bits (4 bytes) in total
- RGB**A** color byte example:

$$\underbrace{11111111}_{\mathbf{R}}\underbrace{00000000}_{\mathbf{G}}\underbrace{00000000}_{\mathbf{B}}\underbrace{11111111}_{\mathbf{A}}$$

Image File Formats

- An image file can have a variety of standard formats
- Image files include information such as:

- **Pixel data** : pixel color and location
- **Header information** : non-pixel data
  - date-time image captures
  - camera settings
  - GPS coordinates at which picture was taken
  - …

Image Files - Compression

- Algorithms used to encode information in fewer bytes than the original
- Used in image files to reduce file size/ required disk storage
- May or may not result in data loss

File size depends upon:

- image dimensions
- color-depth
- compression algorithm

"Deflate" Compression

- Used by Zip archives and PNG image files
- Repeated byte patterns are identified in data stream
- When found, a repeat is replaced by the location in the stream it was last seen along with its length
- Patterns are 4-258 bytes in length
- Patterns are further encoded as a (Huffman) tree

http://www.ietf.org/rfc/rfc1951.txt

"Deflate" Compression

Mary had a little lamb, little lamb, little lamb.
Mary had a little lamb, its fleece was white as snow.

Mary had a little lamb, (•,13)(•,11). (•,24) *its fleece was white as snow.*

(•,13) ≡ Jump to identified location and replay 13 characters

Image Files – Lossy vs. Lossless

Lossless File Formats

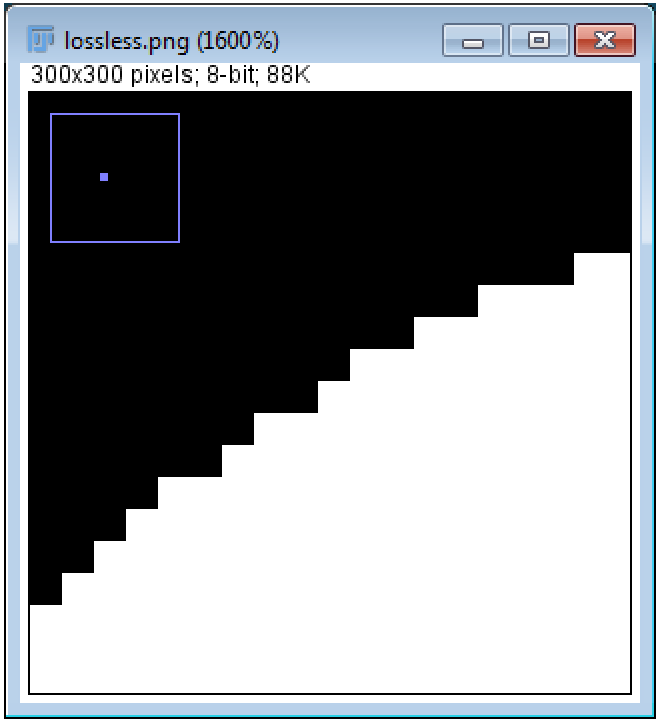
- No pixel information is lost, in spite of compression
- Image can be restored exactly as captured
- Ex: TIFF, PNG, GIF, BMP, …

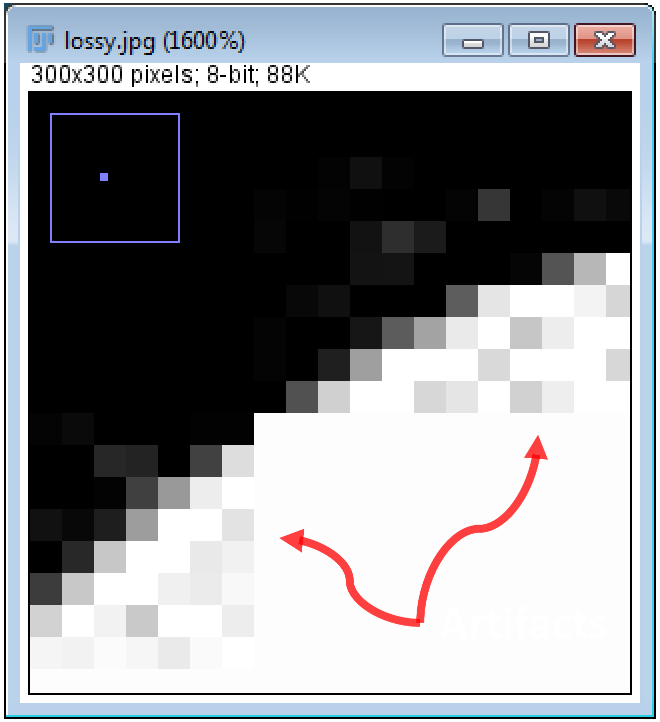
Lossy File Formats

- Compression algorithm approximates pixel values
- Image cannot be restored to original
- Greater compression ratio = smaller files
- Ex: JPEG

Lossy vs. Lossless

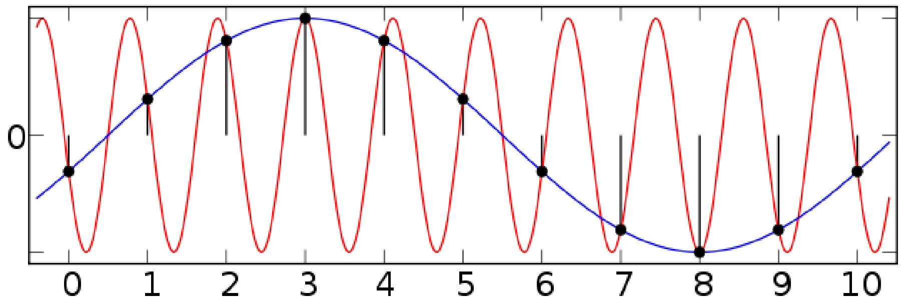
Artifacts in the JPEG file, which are introduced by the lossy compression algorithm





Sampling Theorem

If a function x(t) contains no frequencies higher than W cycles-per-second (hertz), it is completely determined by sampling at a series of points spaced 1/2W seconds apart.



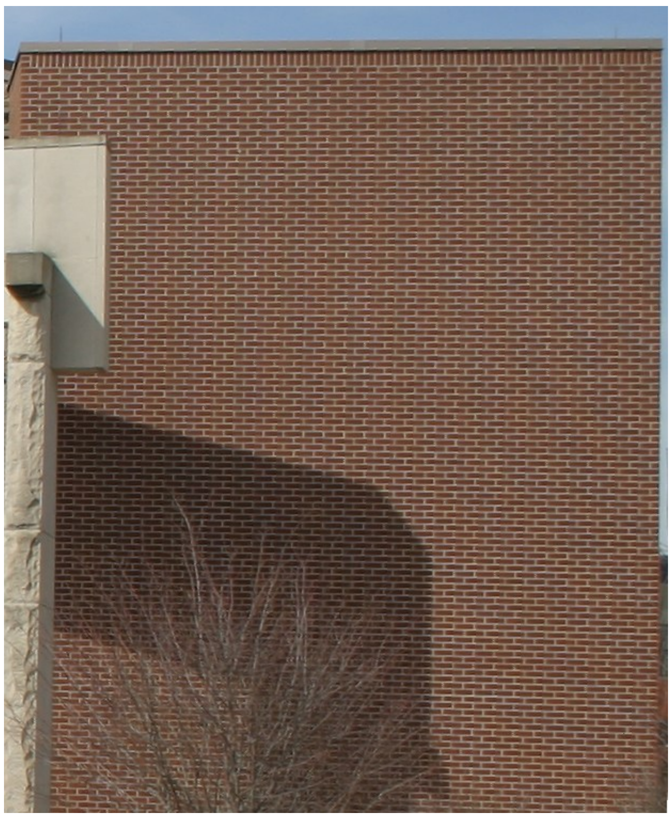
Aliasing occurs when different signals are indistinguishable due to inadequate sampling rate

- For adequate spatial resolution of image features there must be at least 2 pixels per resolvable unit
- Credited to Harry Nyquist and Claude Shannon

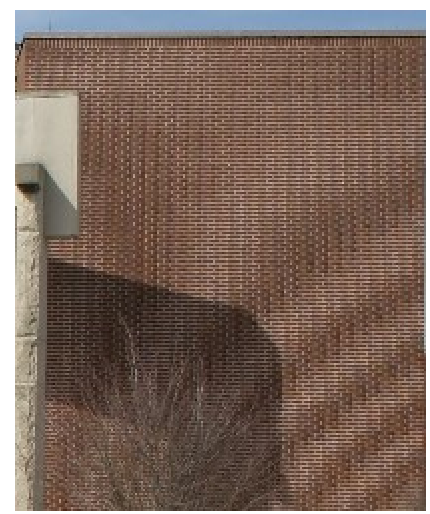
http://en.wikipedia.org/wiki/Sampling_theorem

Moiré Pattern

- When the Sampling Theorem's pixel density requirements are not satisfied a *Moiré Pattern* can result - a kind of aliasing



Downsampling



http://en.wikipedia.org/wiki/Sampling_theorem

Image Processing

Three broad algorithm groups:

1. <u>Intensity Transformation Filters</u> – Point Operations
   - Output image pixel color depends on corresponding input pixel color <u>only</u>

2. <u>Local Filtering/Convolution</u> – Pixel Region Operations
   - Kernel application
   - Output image pixel color depend on pixels near corresponding input pixel

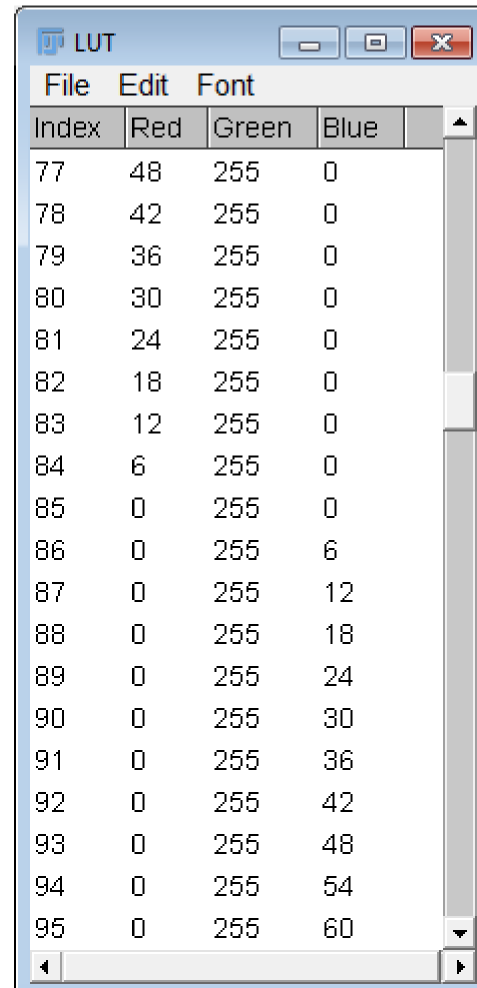
3. <u>Whole Image Algorithms</u>
   - Algorithms that make use of the entire image to perform processing

Point Operations

- Lookup Tables
- Invert
- Grayscale
- Sepia
- Brighten (add an offset to all pixels)
- Saturate
- Posterize
- Threshold
- Image Math (add images, subtract images, …)

Lookup Tables (LUTs)

- Maps 8-bit grayscale pixel values to RGB colors
- Used to transform an image by "looking up" gray scale values and replacing with corresponding color
- A LUT is chosen strategically to highlight features of interest
  - E.g. Use hue to highlight subtle grayscale features

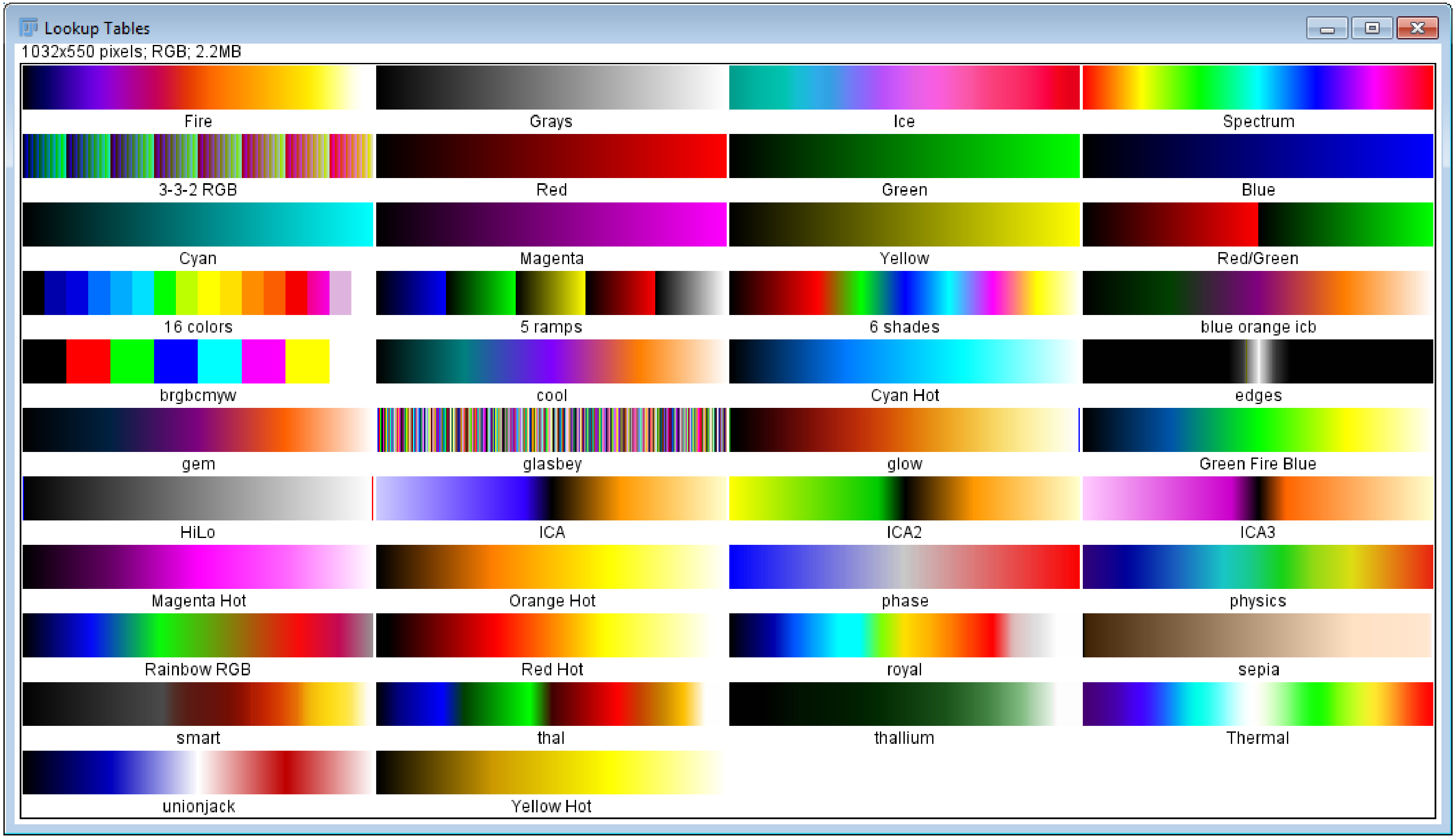


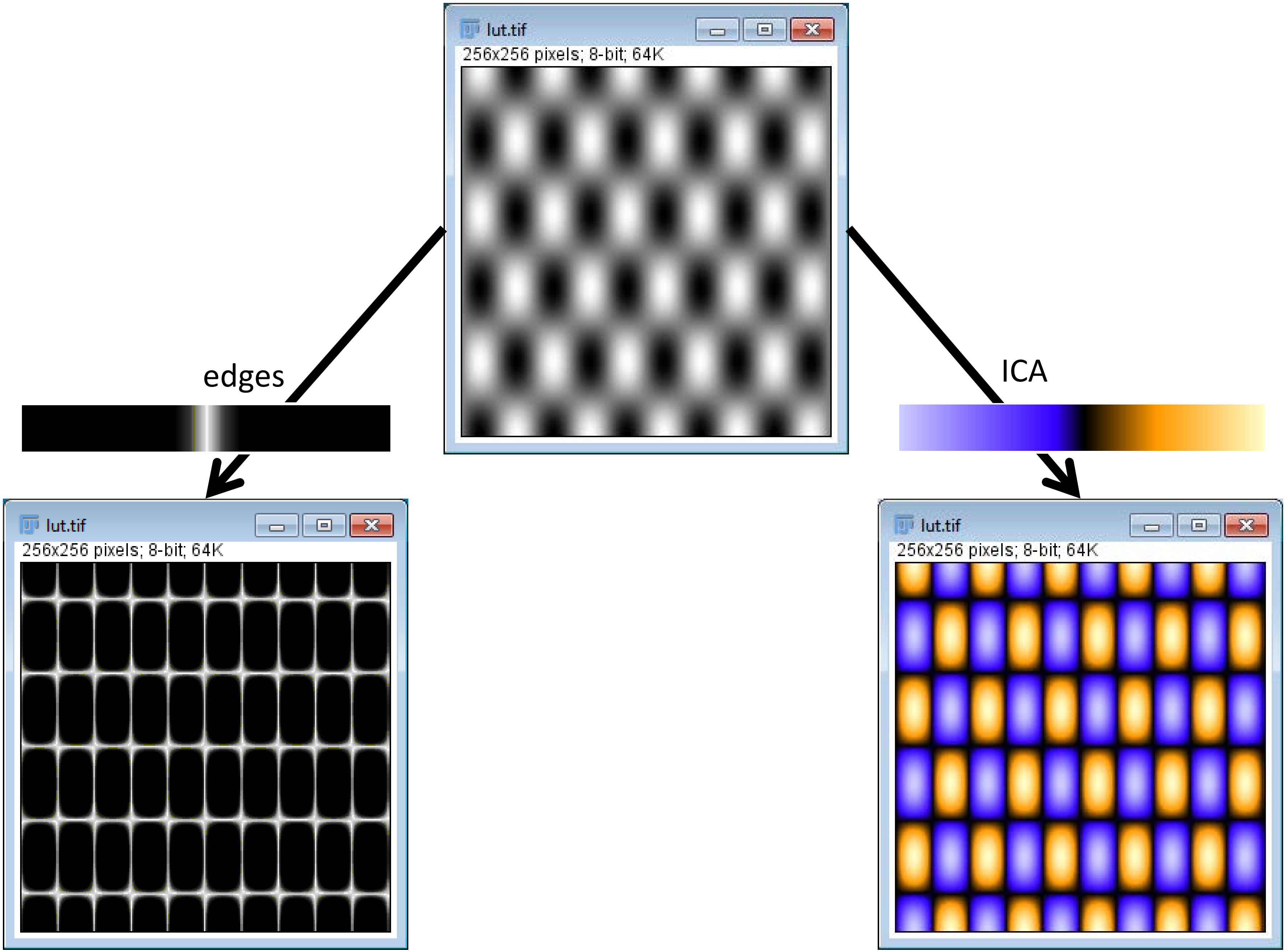
LUT

File Edit Font

Index	Red	Green	Blue
77	48	255	0
78	42	255	0
79	36	255	0
80	30	255	0
81	24	255	0
82	18	255	0
83	12	255	0
84	6	255	0
85	0	255	0
86	0	255	6
87	0	255	12
88	0	255	18
89	0	255	24
90	0	255	30
91	0	255	36
92	0	255	42
93	0	255	48
94	0	255	54
95	0	255	60

Predefined (Named) LUTs




lut.tif
256x256 pixels; 8-bit; 64K
edges
ICA
lut.tif
256x256 pixels; 8-bit; 64K
lut.tif
256x256 pixels; 8-bit; 64K

Point Operations

grayscale:

```
r = g = b = 0.299*r + 0.587*g + 0.114*b;
```

sepia:

```
r = (r * 0.393) + (g * 0.769) + (b * 0.189);
g = (r * 0.349) + (g * 0.686) + (b * 0.168);
b = (r * 0.272) + (g * 0.534) + (b * 0.131);
```

invert:

```
r = 255 - r;
g = 255 - g;
b = 255 - b;
```

brighten (p):

```
r = clamp(r + p);
g = clamp(g + p);
b = clamp(b + p);
```

saturate: (p in [0.0, 2.0])

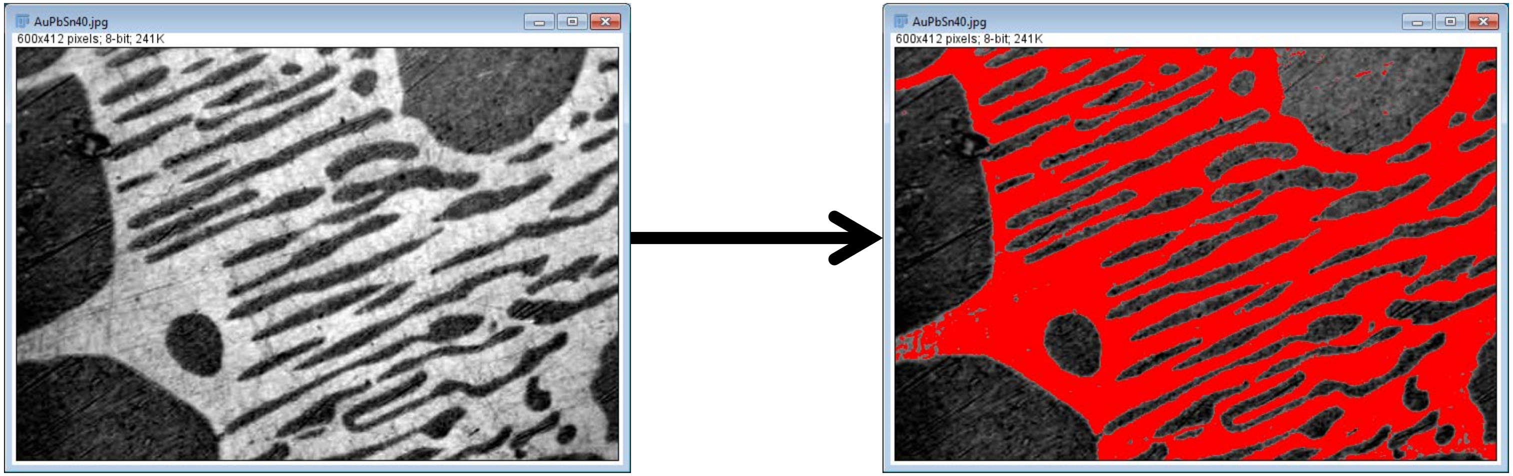
```
var avg = (r + g + b) / 3;
r = avg + p * (r - avg);
g = avg + p * (g - avg);
b = avg + p * (b - avg);
```

posterize: (p in [1, 255])

```
var step = Math.floor(255 / p);
r = Math.floor(r / step) * step;
g = Math.floor(g / step) * step;
b = Math.floor(b / step) * step;
```

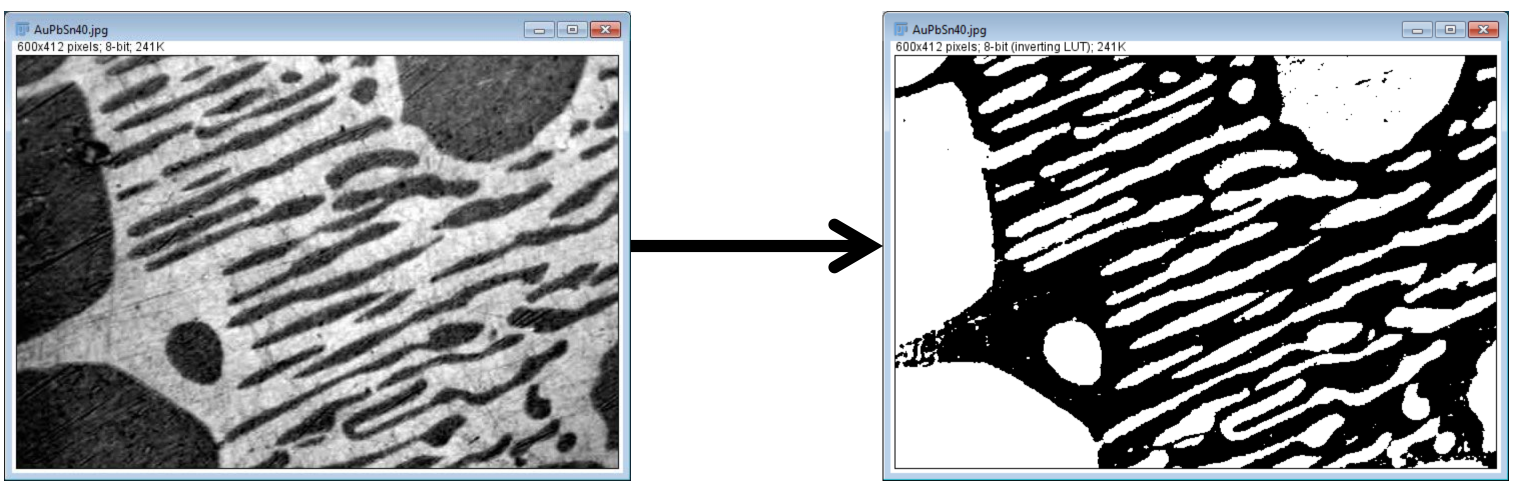
Image Segmentation

- The process of partitioning a digital image into multiple segments of interest
- Example:
  - Highlight the "interesting" part of an alloy image in preparation for analysis

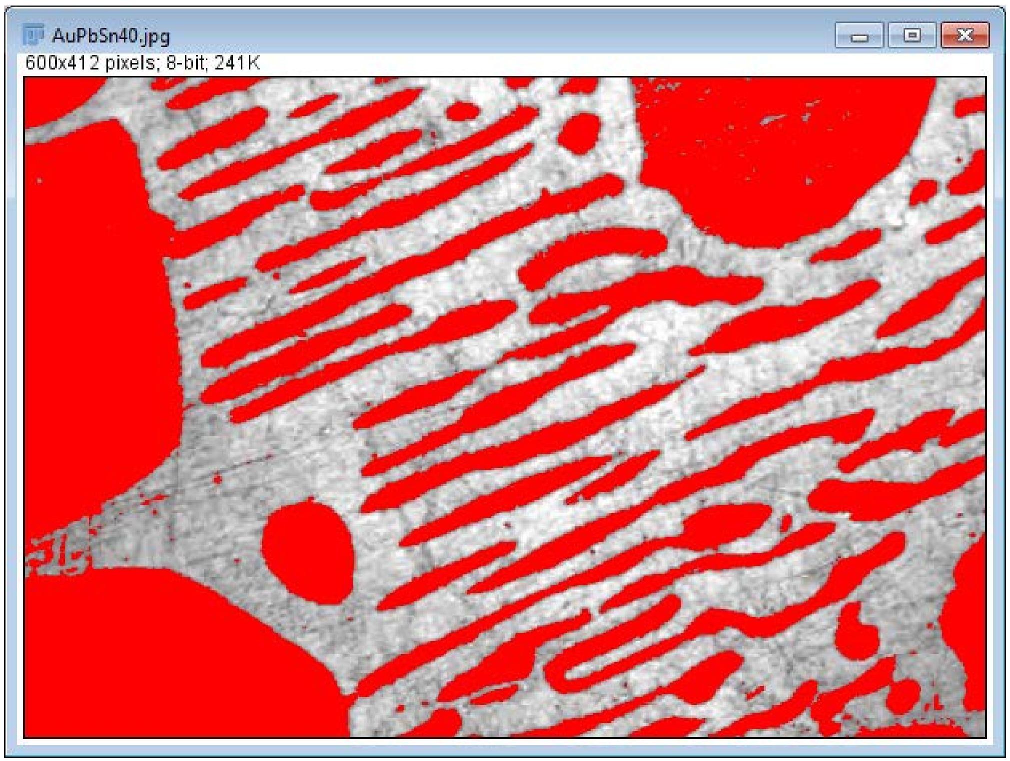


Thresholding

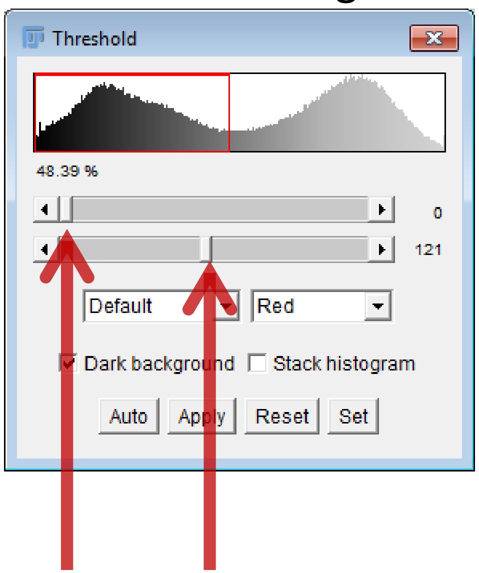
- Simplest method of image segmentation
- Convert an image into a binary format
    - Parts of interest converted to one color
    - Everything else converted to another color
- Further processing on pixels with color of interest



Thresholding



Pixel value histogram

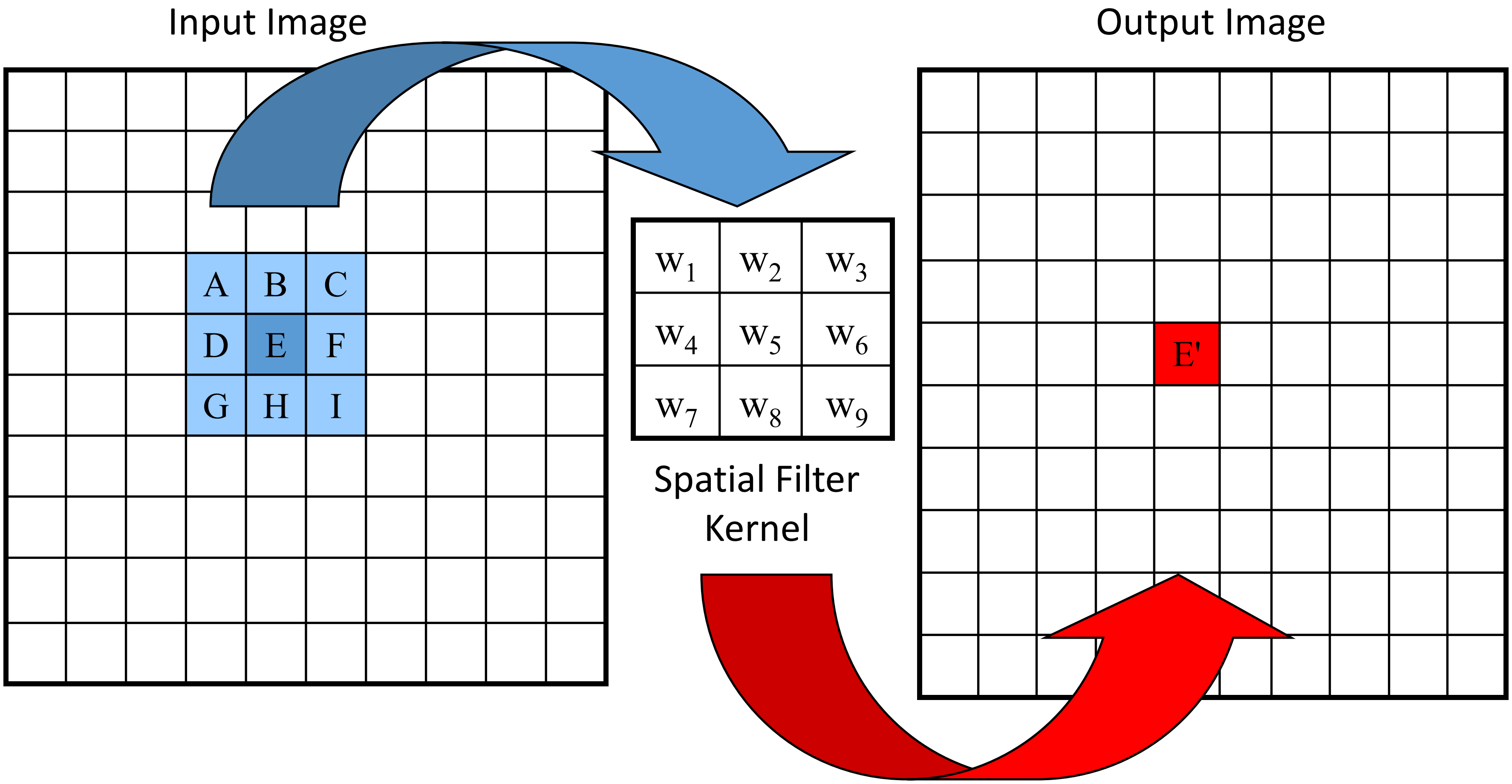


Local Filtering

Examples:

- Image Convolution
  - Identity
  - Smooth (Average)
  - Box Blur
  - Gaussian Blur
  - Sharpen
  - Edge Detection
- Median Filter
- Dilation
- Erosion
- Opening
- Closing

Local Filtering – Convolution Kernel



$$E' = w_1A+w_2B+w_3C+w_4D+w_5E+w_6F+w_7G+w_8H+w_9I$$

Convolution – Identity Kernel

- No change

0	0	0
0	1	0
0	0	0

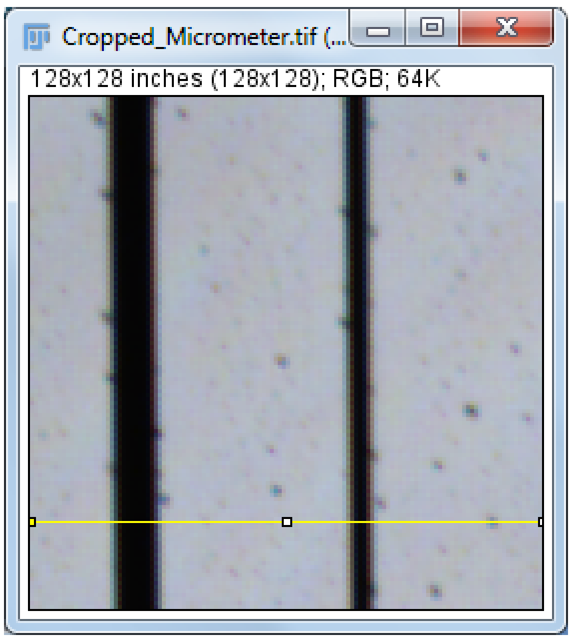
Convolution – Box Blur

- Set pixel to the average of all colors in the neighborhood
- Smooths out areas of sharp changes.

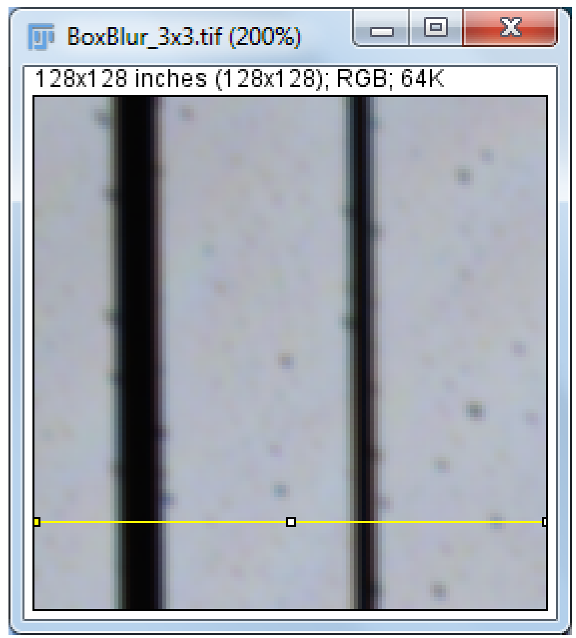
1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25

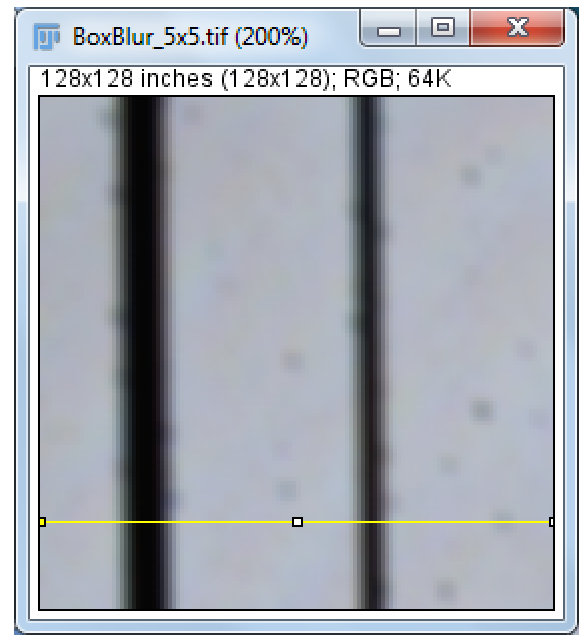
Box Blur Example



Original

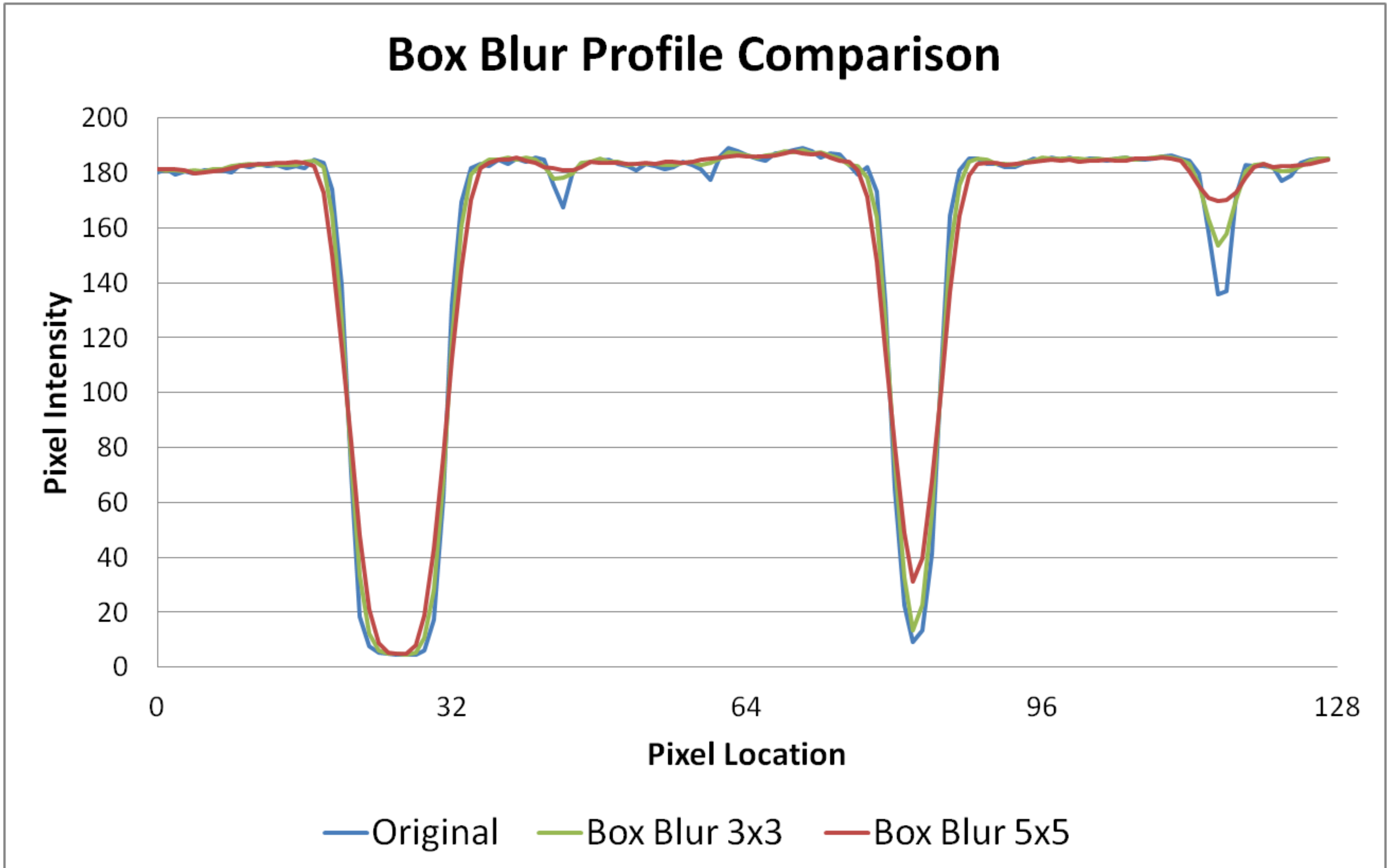


3x3



5x5

Box Blur Profile Comparison



Convolution – Sharpen

- Enhances the difference between neighboring pixels
- The greater the difference, the more change in the output pixel

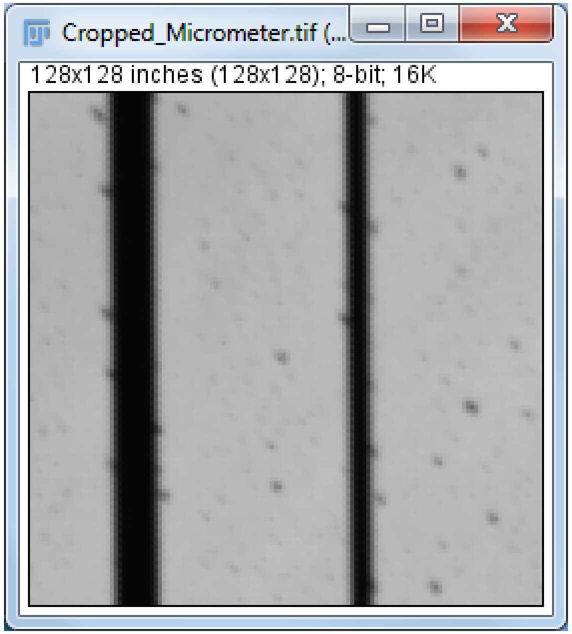
-1/9	-1/9	-1/9
-1/9	1	-1/9
-1/9	-1/9	-1/9

Sharpen 1

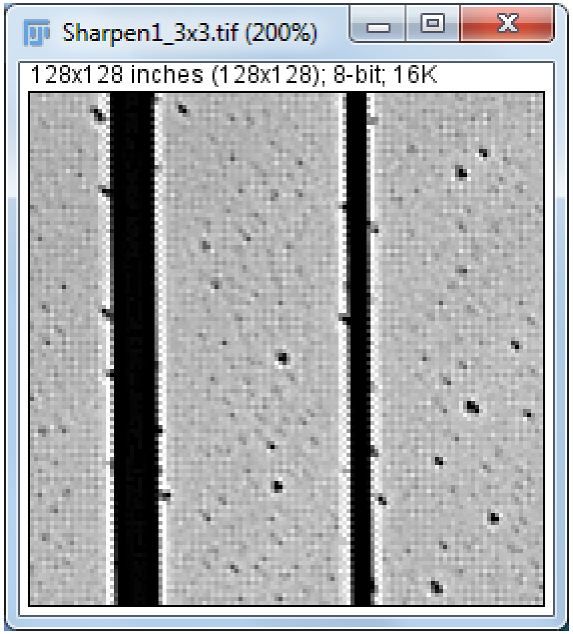
0	-2/3	0
-2/3	11/3	-2/3
0	-2/3	0

Sharpen 2

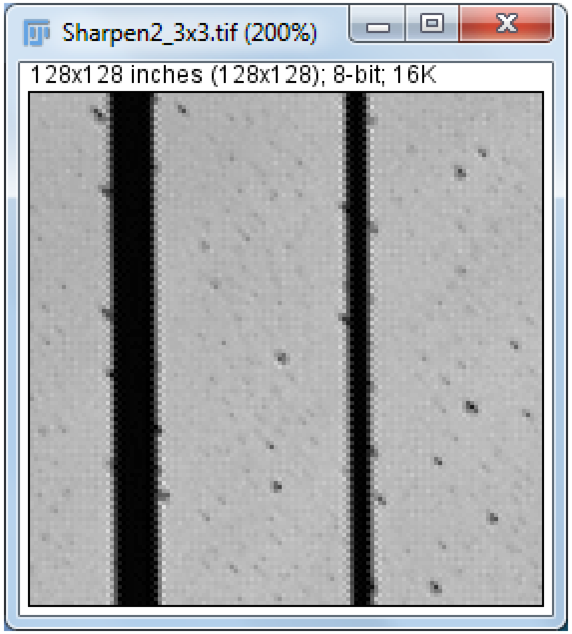
Sharpen Example



Original

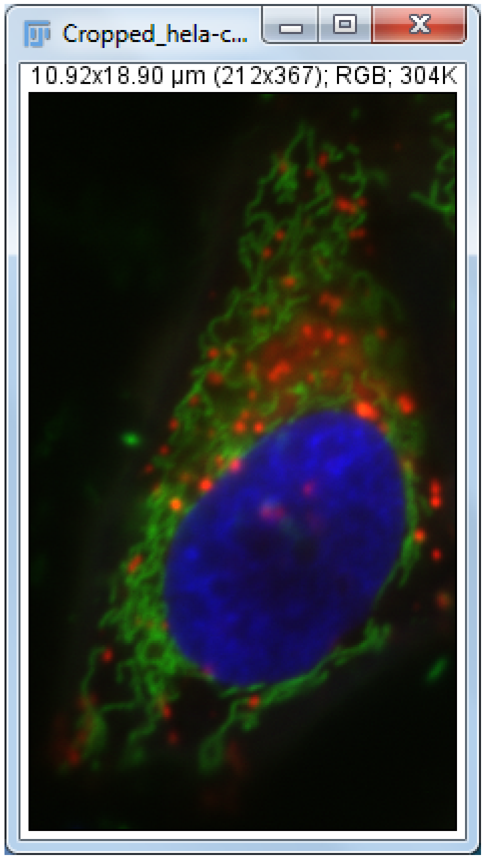


Sharpen 1

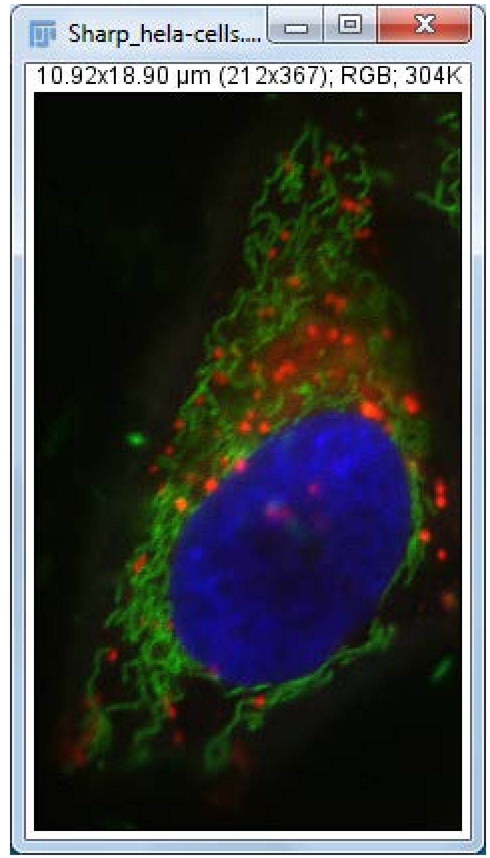


Sharpen 2

Color Sharpen Example

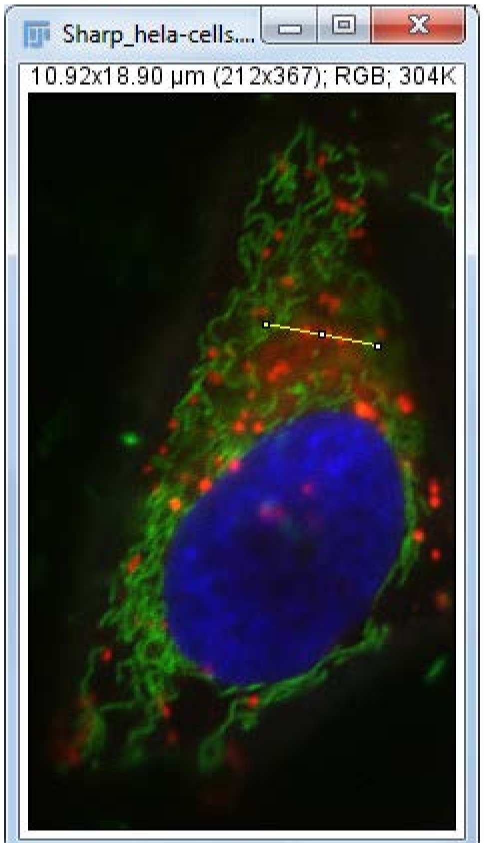


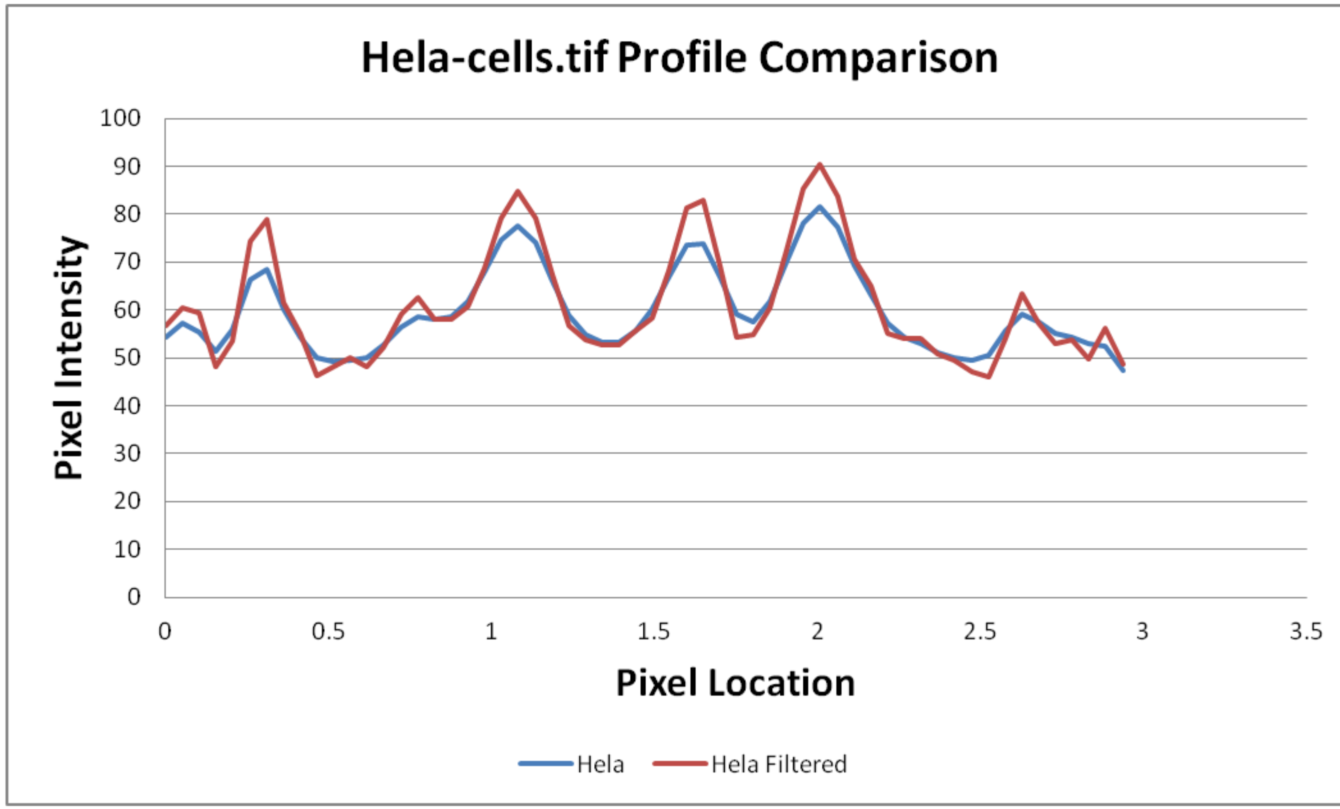
Original



Sharpened

Profile Comparison





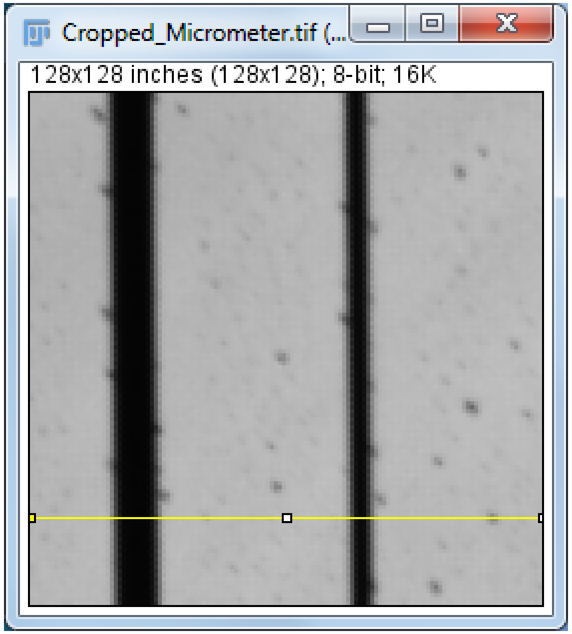
Edge Detection – Sobel Operators

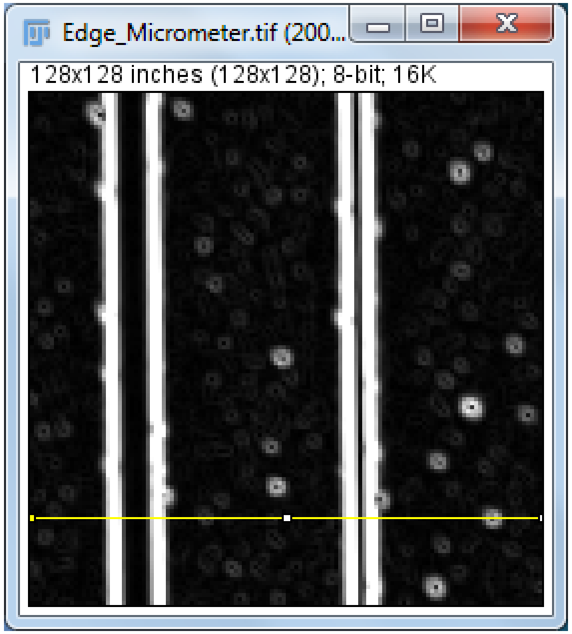
- Two 3×3 kernels are convolved with the original image
- Calculates approximations of the derivatives
  - one for horizontal changes, one for vertical changes
- Final Image produced by taking square root of sum of square of resulting images
- Same method as "Process > Find Edges…"

-1/2	-1	-1/2
0	0	0
+1/2	+1	+1/2

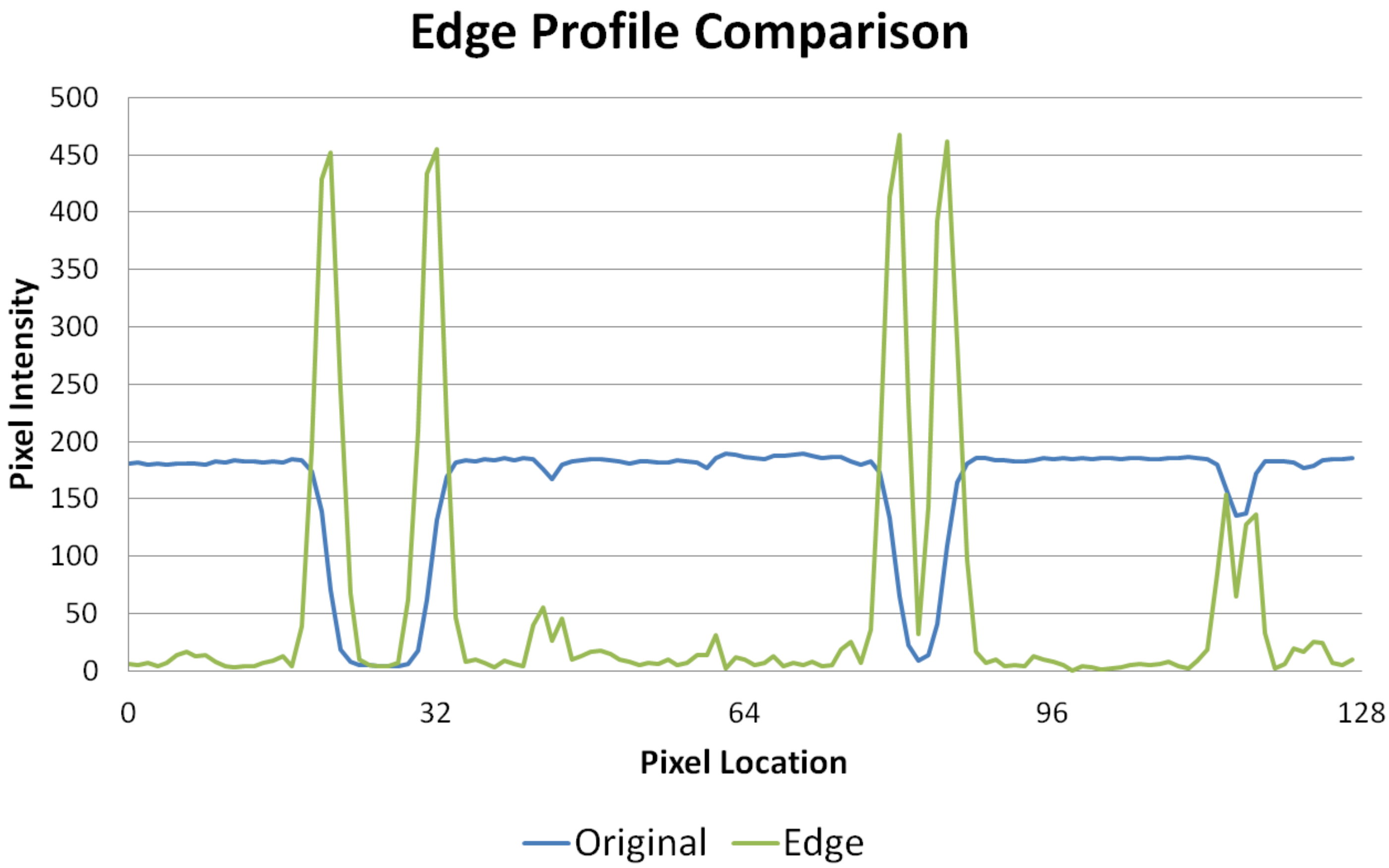
-1/2	0	+1/2
-1	0	+1
-1/2	0	+1/2

Edge Detection – Examples





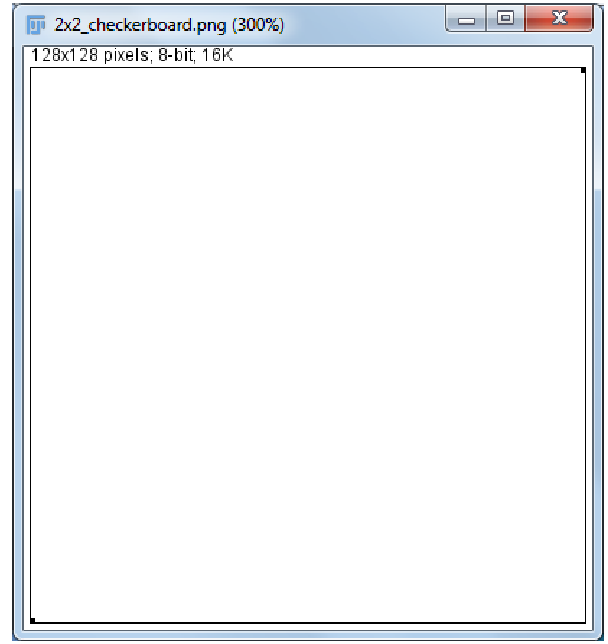
Edge Detection Profile



Dilation

- Set pixel to the <u>maximum color</u> value within a 3x3 window around the pixel
- Causes objects to grow in size.
- Brightens and fills in small holes

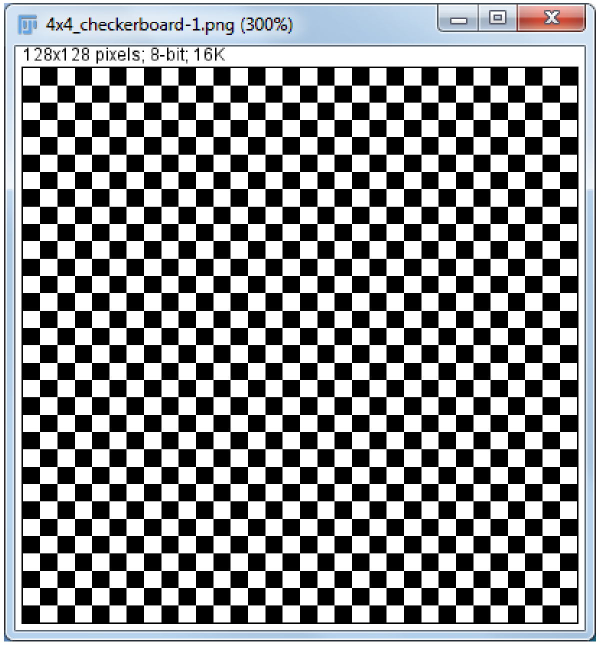
Dilation Examples – 2x2 Checkerboard

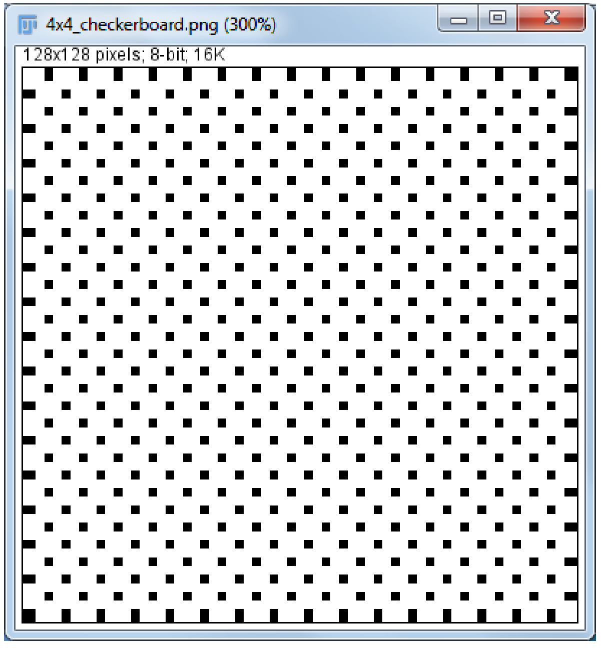


255	255	0
255	255	0
0	0	255

	255	

Dilation Examples – 4x4 Checkerboard





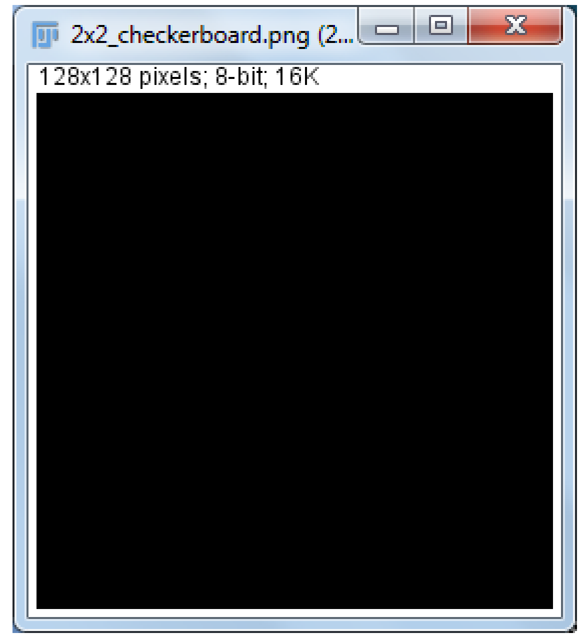
Only 1 location where 4x4 zeros overlay with 3x3, resulting in single pixel of zero

Erosion

- Set pixel to the minimum color value within a 3x3 window around the pixel
- Causes objects to shrink.
- Darkens and removes small objects

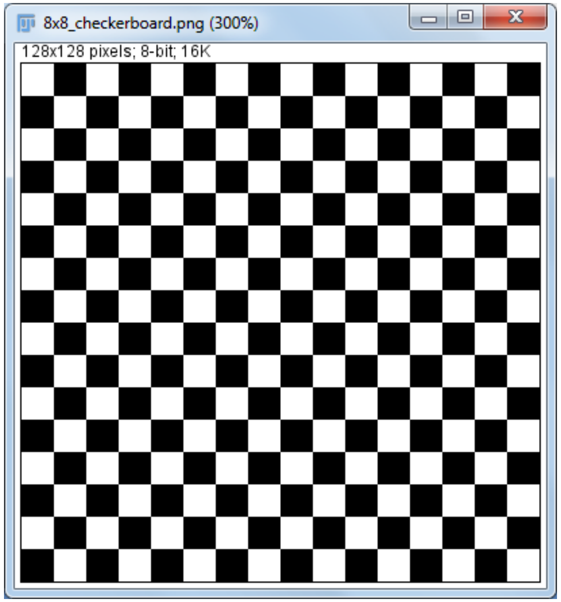
Erosion Examples – 2x2 Checkerboard

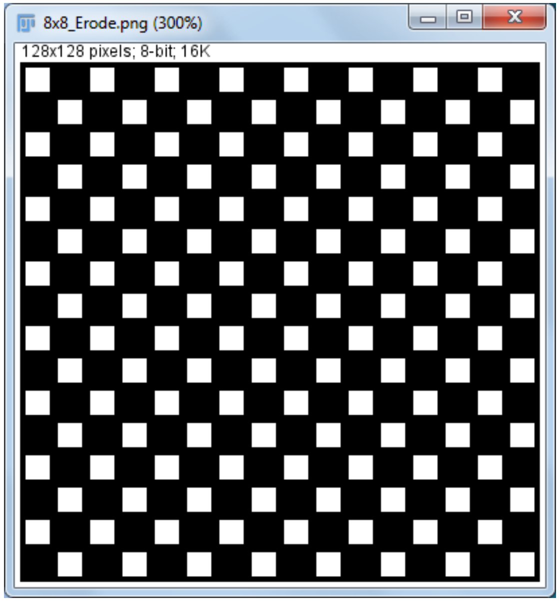
255	255	0
255	255	0
0	0	255



	0	

Erosion Examples – 8x8 Checkerboard





Opening (Erosion -> Dilation)

CT Image

Binary Image

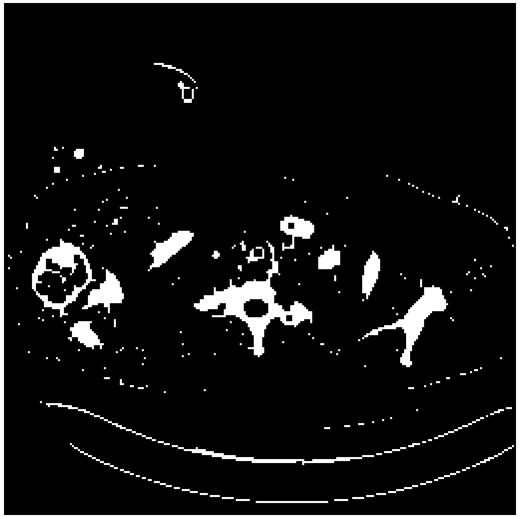
Open Result

Note: Using Grays LUT for before and after comparison.

Close (Dilation -> Erosion)

CT Image

Binary Image



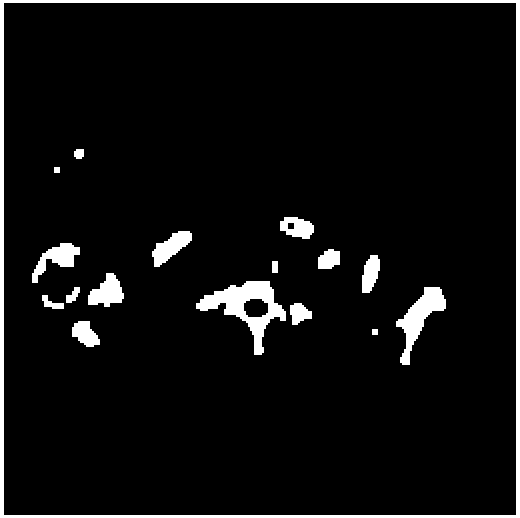
Close Result

Note: Using Grays LUT for before and after comparison.

Close and Open

CT Image

Binary Image



Close/Open Result

Note: Using Grays LUT for before and after comparison.

Median Filter

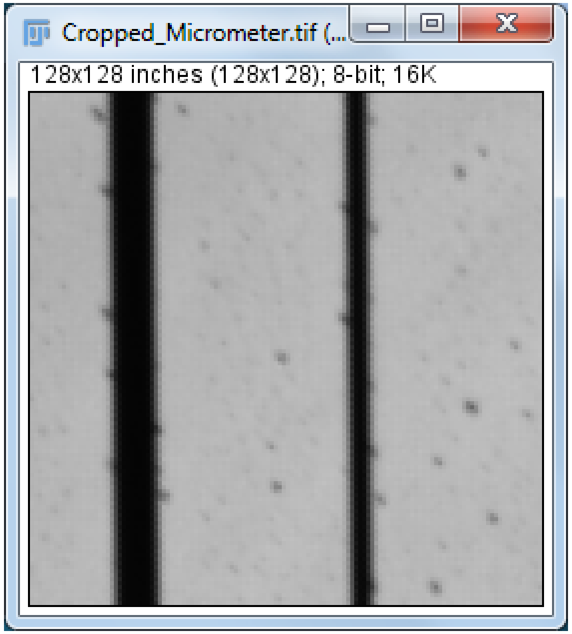
- Replace each pixel in the output image with the median of neighboring pixels in the input image
  - sort 9 input pixel values
  - select middle value (index 4)

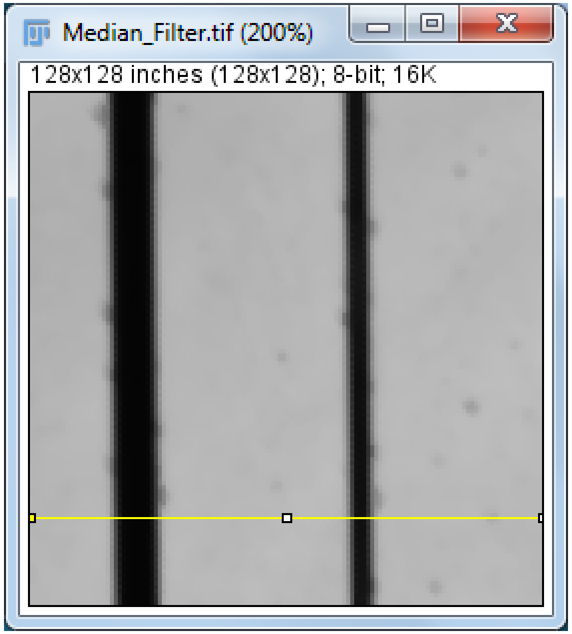
[103, 107, 110, 112, $\boxed{118,}$ 123, 125, 135, 153]

123	107	153
110	125	<u>**118**</u>
135	112	103

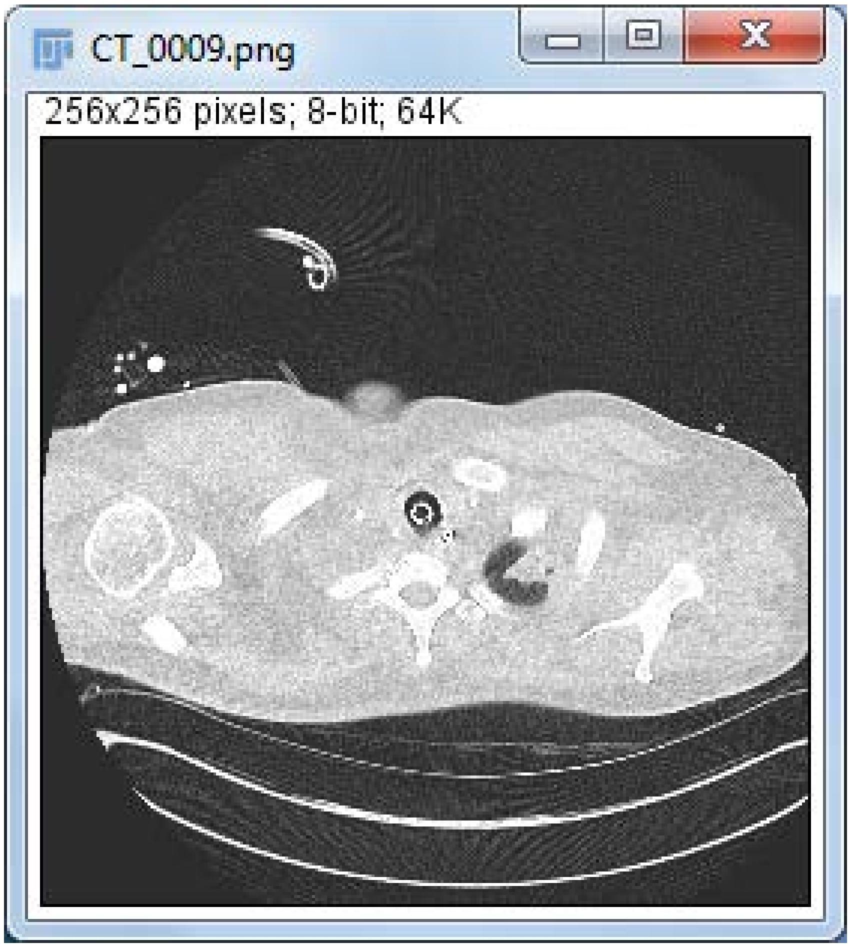
	118	

Median Filter - Example

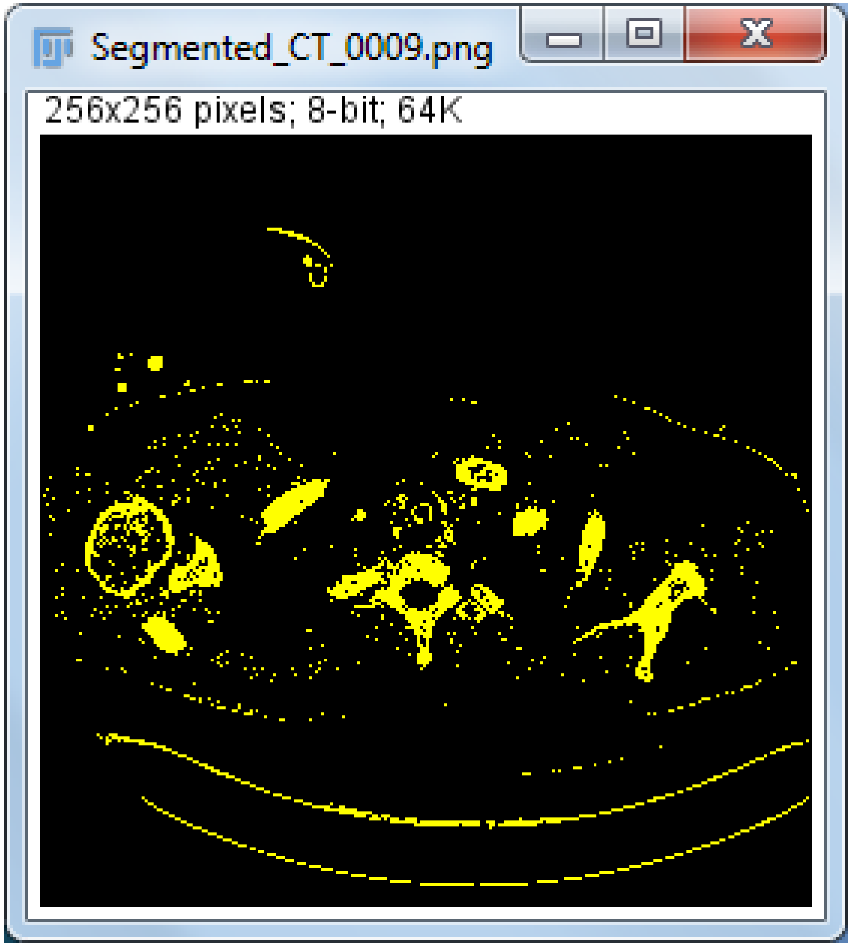


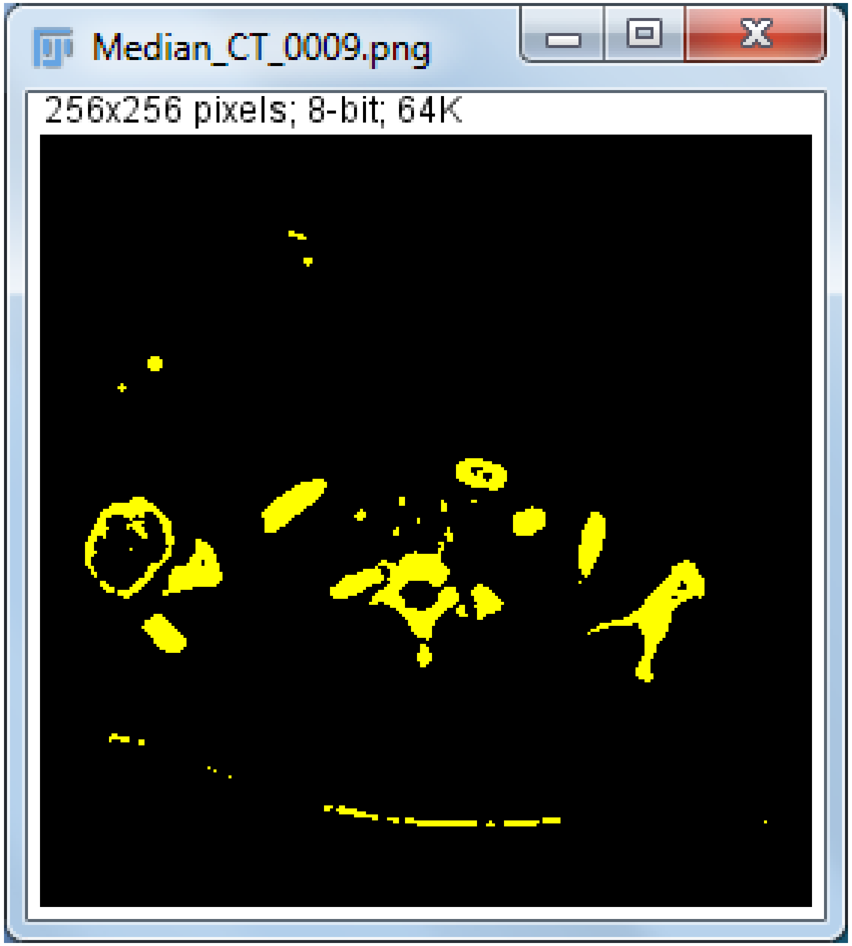


Median Filter – Example 2



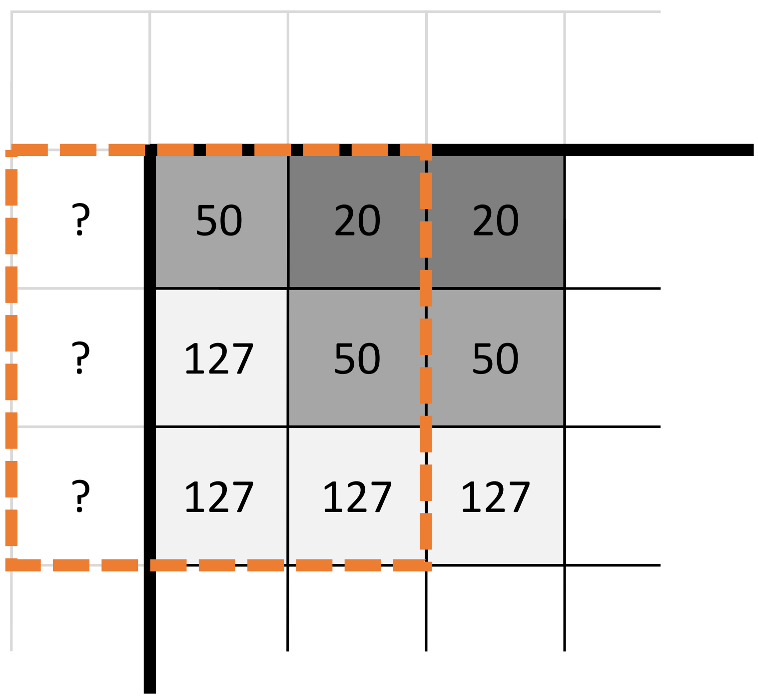
Threshold -> Median Filter





What do we do with edge pixels?

1. Wrap the image
2. Ignore edge pixels and only compute for those pixels with all neighbors
3. Duplicate edge pixels



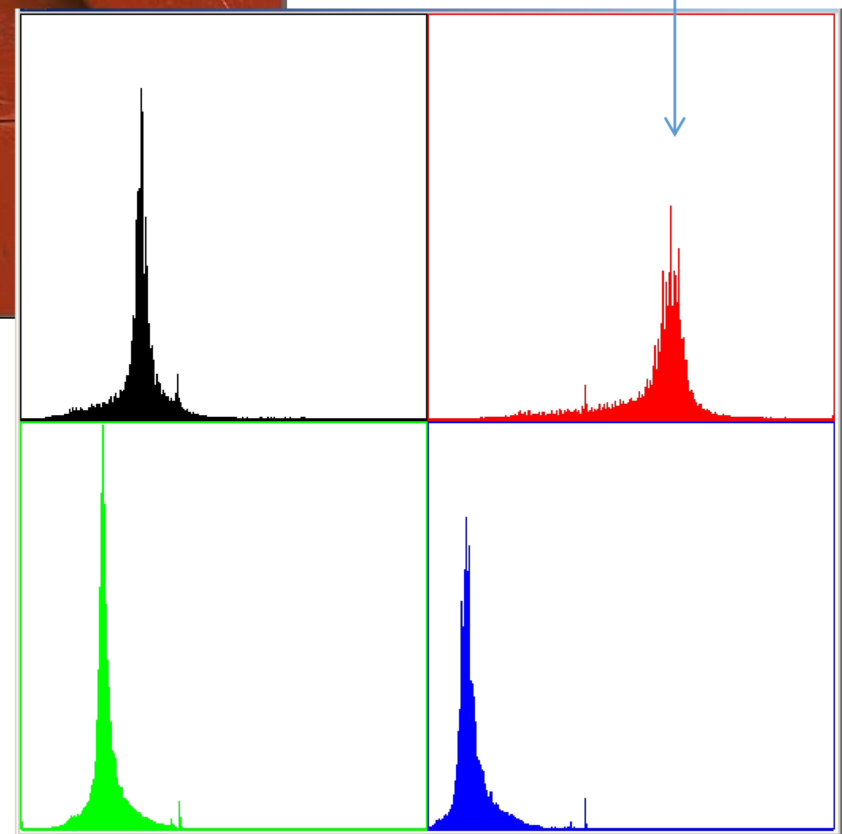
Whole-Image Algorithms

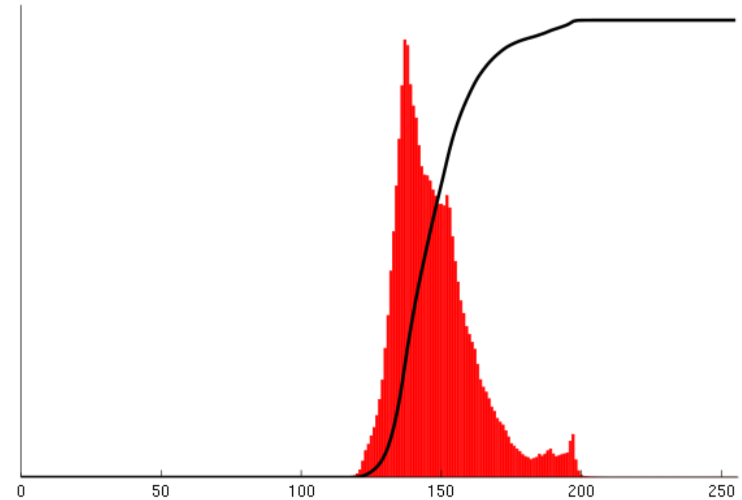
- Histogram Equalization (aka Contrast Stretching)
- Watershed Segmentation
- Rolling Ball Background Subtraction
- …

Histogram Equalization

- Increase the global contrast of images
- Intensities are better distributed
- Reveals more detail in images that are over or under exposed

Shift to the right
implies brighter reds



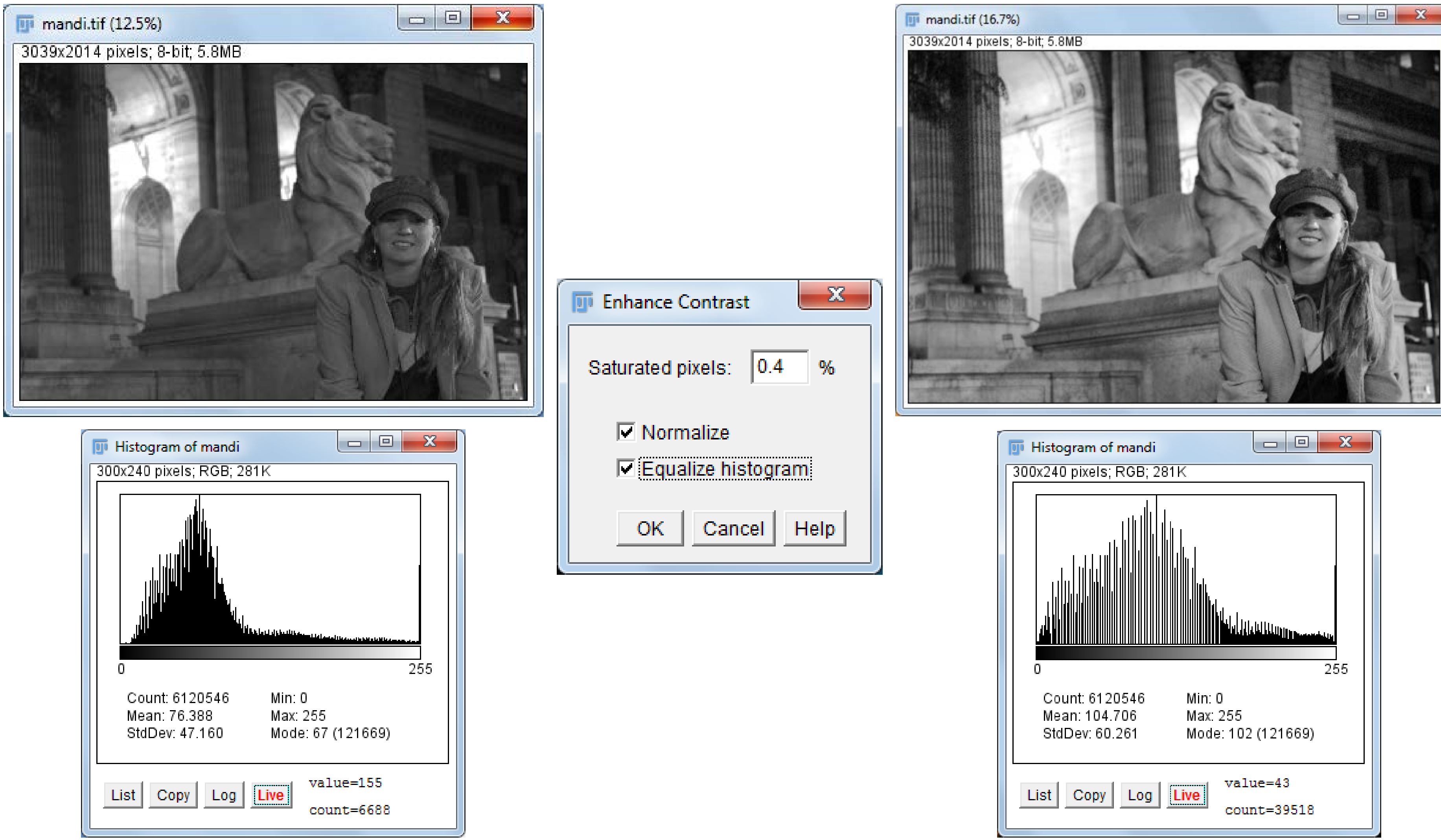

0
50
100
150
200
250


0
50
100
150
200
250

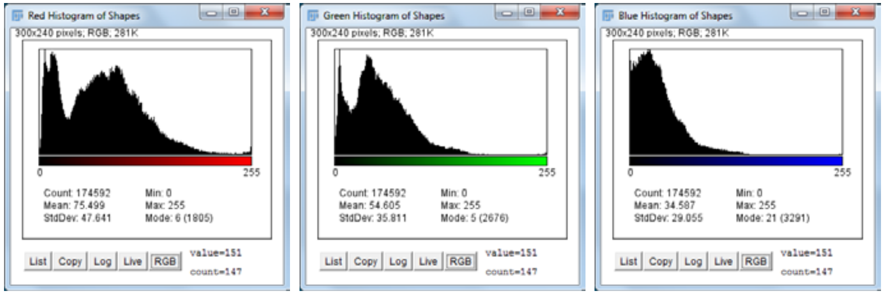
Histogram Equalization

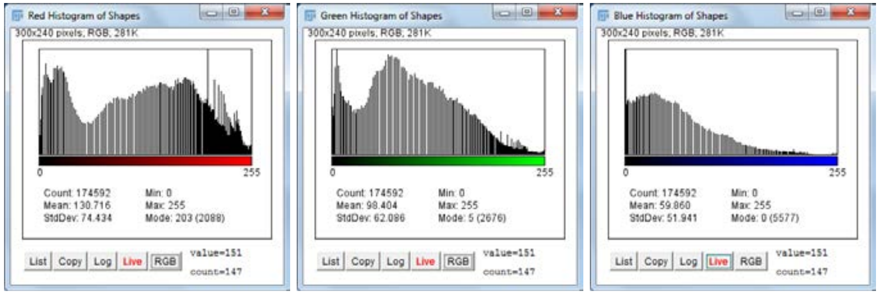
- Calculate color frequencies - count the number of times each pixel color appear in the image
- Calculate the cumulative distribution function (cdf) for each pixel color – the number of times all smaller color values appear in the image
- Normalize over (0, 255)

Histogram Equalization Example

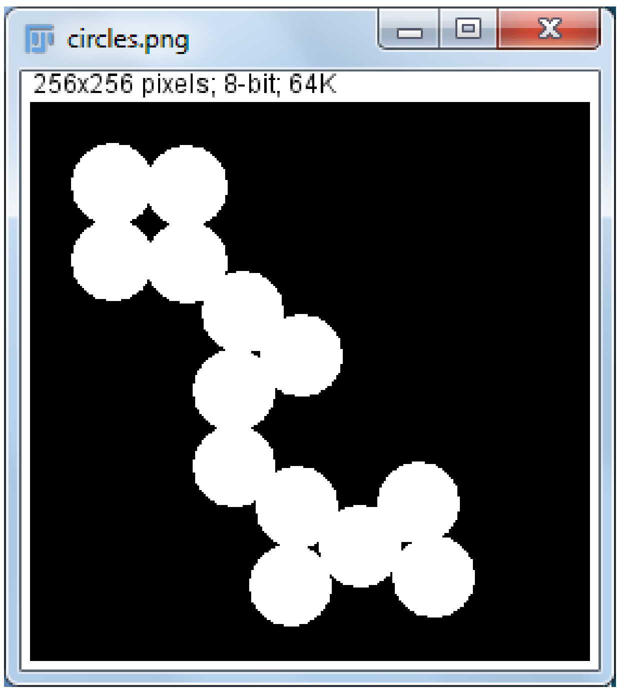


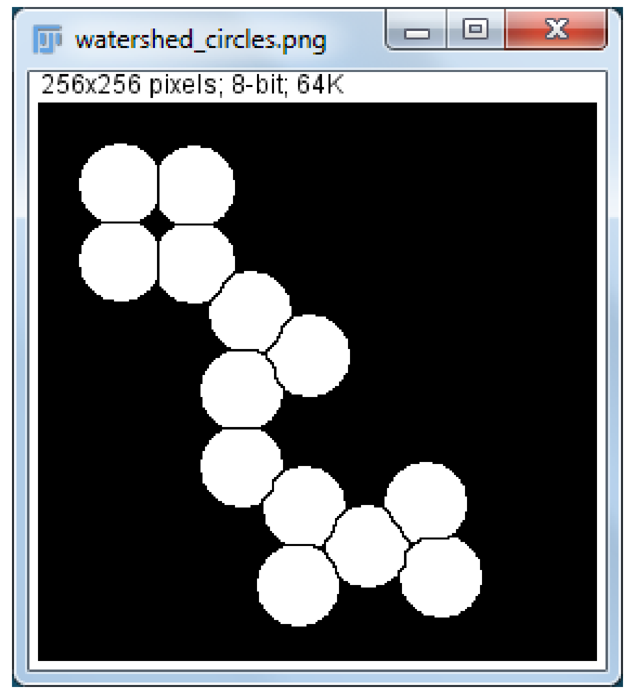
Color Histogram Equalization





Watershed Segmentation





Rolling Ball Background Subtraction

- A local background value is determined for every pixel by averaging over a very large ball around the pixel.
- This value is hereafter subtracted from the original image, (hopefully) removing large spatial variations of the background intensities.
- The radius should be set to at least the size of the largest object that is not part of the background.

Rolling Ball Background Subtraction