

+/- Message

```
## Loading required package: knitr
```

+/- R Code

+/- R Code

+/- R Code

## Quantitative Big Imaging

author: Kevin Mader date: 5 March 2015 width: 1440 height: 900 css: ../common/template.css transition: rotate

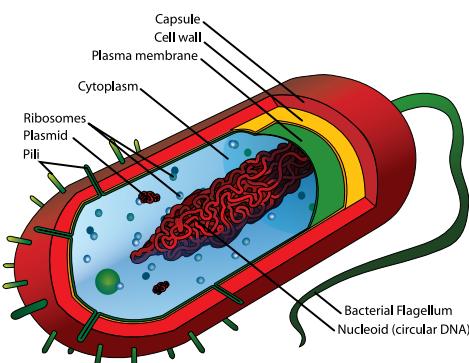
### Basic Segmentation and Discrete Binary Structures

### Course Outline

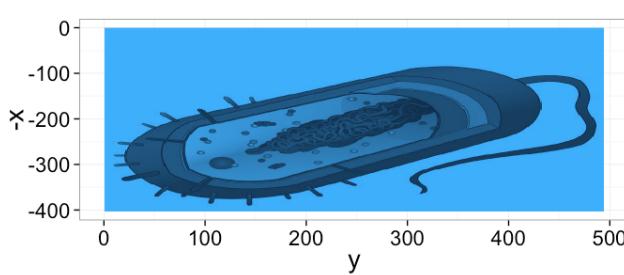
+/- R Code

- 19th February - Introduction and Workflows
- 26th February - Image Enhancement (A. Kaestner)
- 5th March - Basic Segmentation, Discrete Binary Structures
- 12th March - Advanced Segmentation
- 19th March - Machine Learning in Image Processing (M. Jäggi and A. Lucchi)
- 26th March - Analyzing Single Objects
- 2nd April - Analyzing Complex Objects
- 16th April - Spatial Distribution
- 23rd April - Statistics and Reproducibility
- 30th April - Dynamic Experiments
- 7th May - Scaling Up / Big Data
- 21th May - Guest Lecture, Applications in Material Science

- What we are looking at



- What we get from the imaging modality



+/- R Code

## To test a hypothesis

- 28th May - Project Presentations

## Lesson Outline

- Motivation
- The old ways
- Thresholding
  - Other types of images
  - Selecting a good threshold
- Implementation
- Morphology
- Applications

## Literature / Useful References

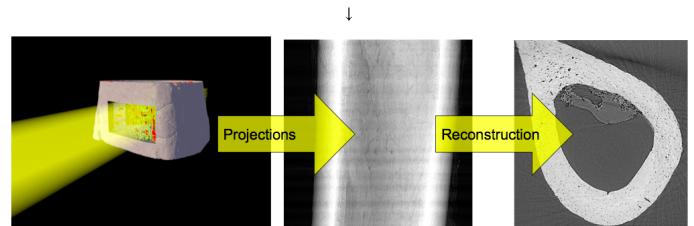
- Jean Claude, Morphometry with R
  - Online (<http://link.springer.com/book/10.1007%2F978-0-387-77789-4>) through ETHZ
  - Buy it (<http://www.amazon.com/Morphometrics-R-Use-Julien-Claude/dp/038777789X>)
- John C. Russ, "The Image Processing Handbook", (Boca Raton, CRC Press)
  - Available online (<http://dx.doi.org/10.1201/9780203881095>) within domain ethz.ch (or proxy.ethz.ch / public VPN)

## Motivation: Why do we do imaging experiments?

#incremental: true

- To get an idea of what is going on
- To test a hypothesis
  - Does temperature affect bubble size?
  - Is this gene important for cell shape and thus mechanosensation in bone?
  - Does higher canal volume make bones weaker?
  - Does the granule shape affect battery life expectancy?

- We perform an experiment bone to see how big the cells are inside the tissue



$2560 \times 2560 \times 2160 \times 32 \text{ bit} = 56\text{GB} / \text{sample}$

- Filtering and Preprocessing!

- 20h of computer time later ...
- 56GB of less noisy data
- Way too much data, we need to reduce

## What did we want in the first place

#incremental: true

### Single number:

- volume fraction,
- cell count,
- average cell stretch,
- cell volume variability

## Why do we perform segmentation?

- In model-based analysis every step we perform, simple or complicated is related to an underlying model of the system we are dealing with
- Occam's Razor ([http://en.wikipedia.org/wiki/Occams\\_Razor](http://en.wikipedia.org/wiki/Occams_Razor)) is very important here : The simplest solution usually the right one
  - Bayesian, neural networks optimized using genetic algorithms with Fuzzy logic has a much larger

parameter space to explore, establish sensitivity in, and must perform much better and be tested much more thoroughly than thresholding to be justified.

- We will cover some of these techniques in the next 2 lectures since they can be very powerful particularly with unknown data

## Review: Filtering and Image Enhancement

incremental: true

- This was a noise process which was added to otherwise clean imaging data

$$I_{measured}(x, y) = I_{sample}(x, y) + \text{Noise}(x, y)$$

- What would the perfect filter be

- $\text{Filter} * I_{sample}(x, y) = I_{sample}(x, y)$

- $\text{Filter} * \text{Noise}(x, y) = 0$

- $\text{Filter} * I_{measured}(x, y) = \text{Filter} * I_{real}(x, y) + \text{Filter} * \text{Noise}(x, y) \rightarrow I_{sample}(x, y)$

- What most filters end up doing

- $\text{Filter} * I_{measured}(x, y) = 90\%I_{real}(x, y) + 10\%\text{Noise}(x, y)$

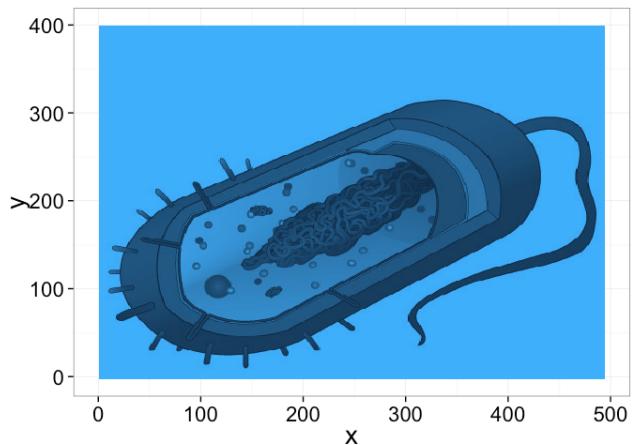
- What bad filters do

- $\text{Filter} * I_{measured}(x, y) = 10\%I_{real}(x, y) + 90\%\text{Noise}(x, y)$

## What did people used to do?

- What comes out of our detector / enhancement process

+/- R Code



- Identify objects by eye
  - Count, describe qualitatively: "many little cilia on surface", "long curly flagellum", "elongated nuclear structure"
- Morphometrics
  - Trace the outline of the object (or sub-structures)
  - Can calculate the area by using equal-weight-paper
  - Employing the "cut-and-weigh" (<http://ion.chem.usu.edu/%7Esbialkow/Classes/361/GC/GC.html>) method

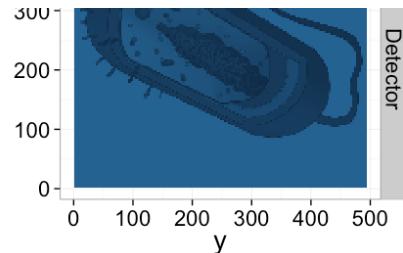
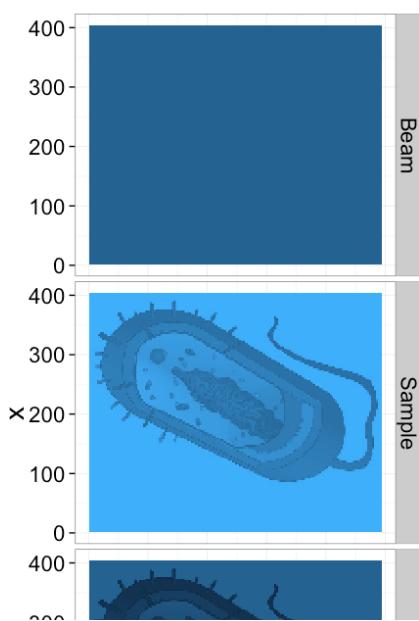
## Model-based Analysis

- Many different imaging modalities (\$\mu\$CT to MRI to Confocal to Light-field to AFM).
- Similar equations
  - different coefficients, units, and mechanism

$$S_{measured}(\vec{x}) = F_{system}(S_{stimulus}(\vec{x}), S_{sample}(\vec{x}))$$

### Direct Imaging (simple)

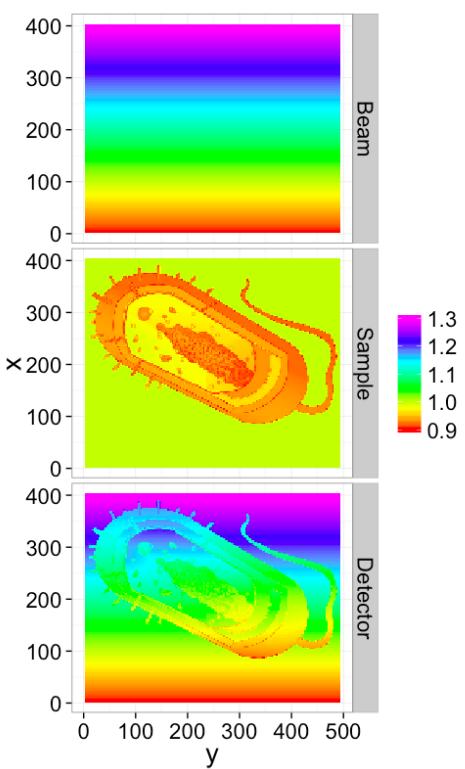
- $F_{system}(a, b) = a * b$
- $S_{stimulus} = \text{Beam}_{profile}$
- $S_{system} = \alpha(\vec{x})$
- $\rightarrow \alpha(\vec{x}) = \frac{S_{measured}}{\text{Beam}_{profile}}$



## Nonuniform Beam-Profiles

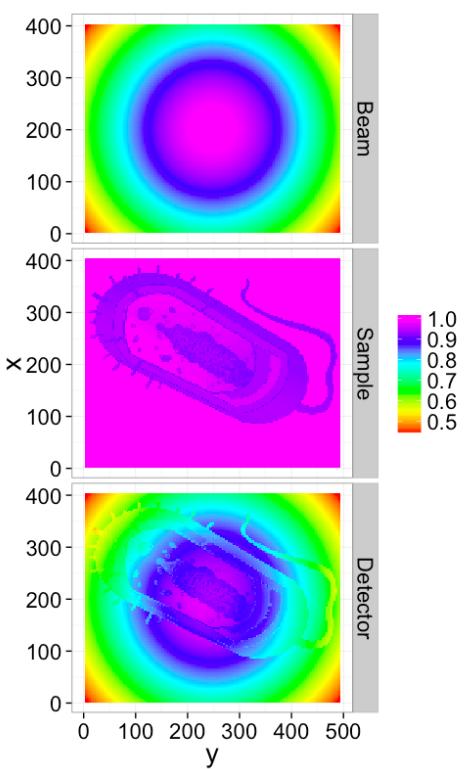
In many setups there is un-even illumination caused by incorrectly adjusted equipment and fluctuations in power and setups

+/- R Code



Frequently there is a fall-off of the beam away from the center (as is the case of a Gaussian beam which frequently shows up for laser systems). This can make extracting detail away from the center challenging

[+/- R Code](#)



## Absorption Imaging (X-ray, Ultrasound, Optical)

- For segmentation this model is:

- there are 2 (or more) distinct components that make up the image
- these components are distinguishable by their values (or vectors, colors, tensors, ...)
- For absorption/attenuation microscopy, Beer-Lambert Law ([http://en.wikipedia.org/wiki/Attenuation\\_coefficient](http://en.wikipedia.org/wiki/Attenuation_coefficient))

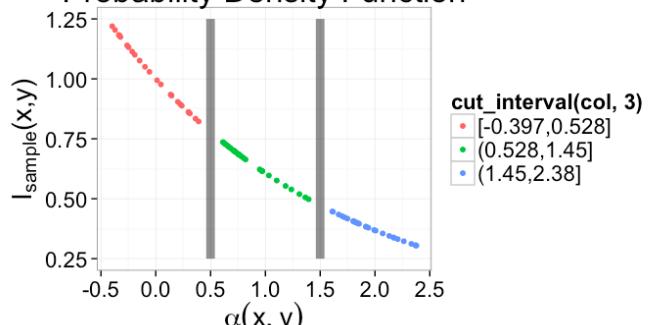
$$I_{\text{detector}} = I_{\text{source}} \exp(-\alpha d)$$

- Different components have a different  $\alpha$  based on the strength of the interaction between the light and the chemical / nuclear structure of the material

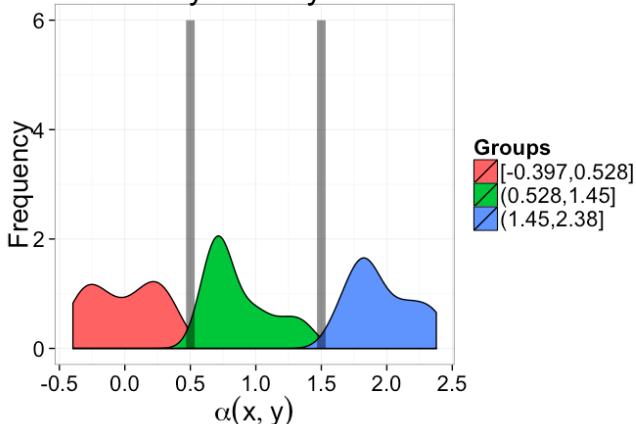
$$I_{\text{sample}}(x, y) = I_{\text{source}} \exp(-\alpha(x, y)d)$$

$$\alpha = f(N, Z, \sigma, \dots)$$

## Probability Density Function



## Probability Density Function



+/- R Code

## Where does segmentation get us?

incremental: true

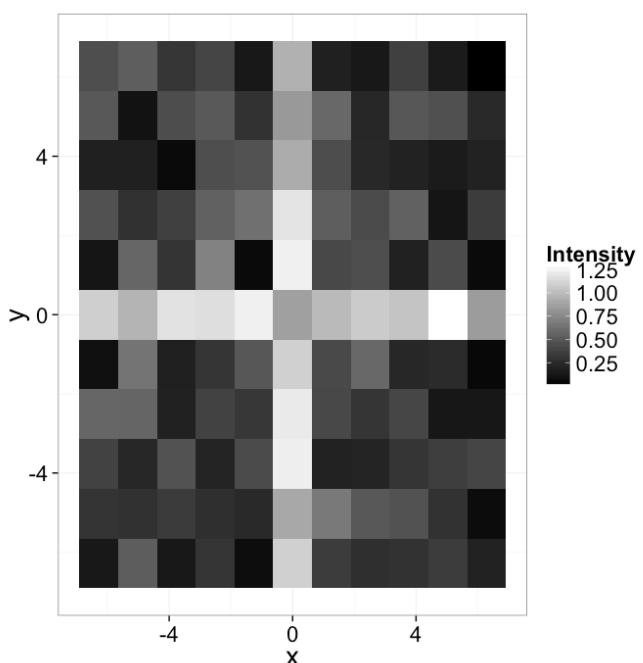
- We convert a decimal value (or something even more complicated like 3 values for RGB images, a spectrum for hyperspectral imaging, or a vector / tensor in a mechanical stress field)
- to a single, discrete value (usually true or false, but for images with phases it would be each phase, e.g. bone, air, cellular tissue)
- $2560 \times 2560 \times 2160 \times 32 \text{ bit} = 56\text{GB} / \text{sample}$
- $2560 \times 2560 \times 2160 \times 1 \text{ bit} = 1.75\text{GB} / \text{sample}$

↓

## Applying a threshold to an image

Start out with a simple image of a cross with added noise

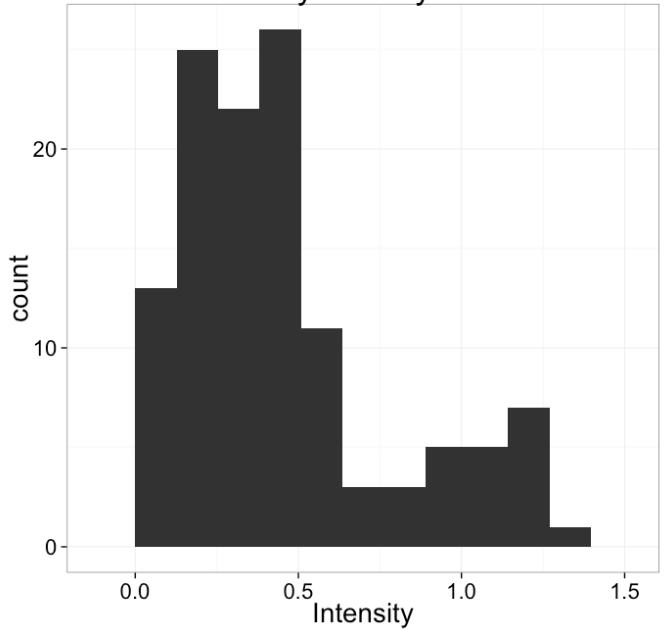
$$I(x, y) = f(x, y)$$



The intensity can be described with a probability density function

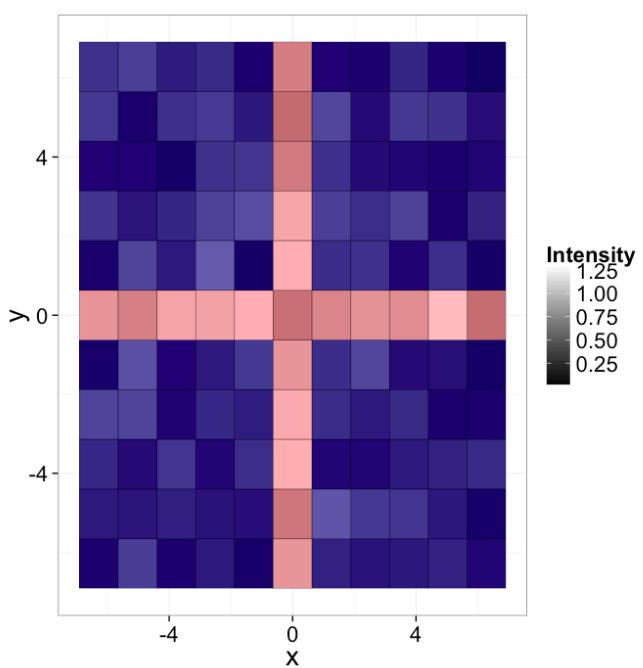
$$P_f(x, y)$$

## Probability Density Function



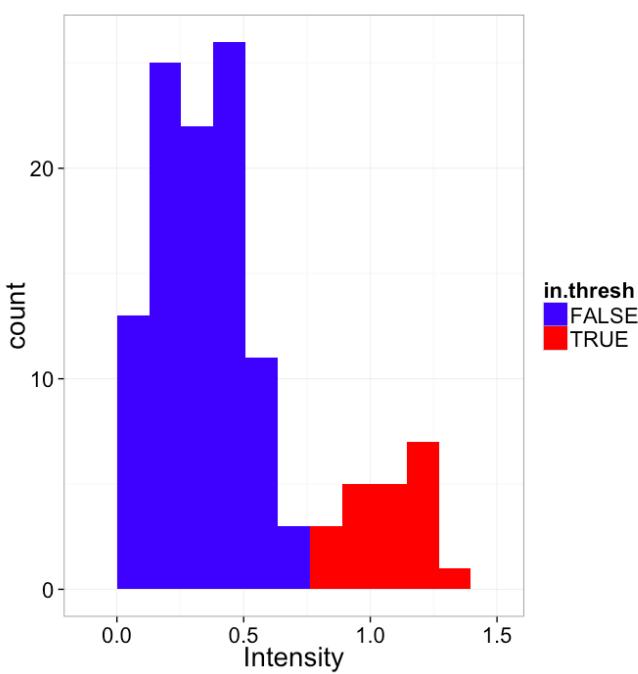
## Applying a threshold to an image

By examining the image and probability distribution function, we can deduce that the underlying model is a whitish phase that makes up the cross and the darkish background

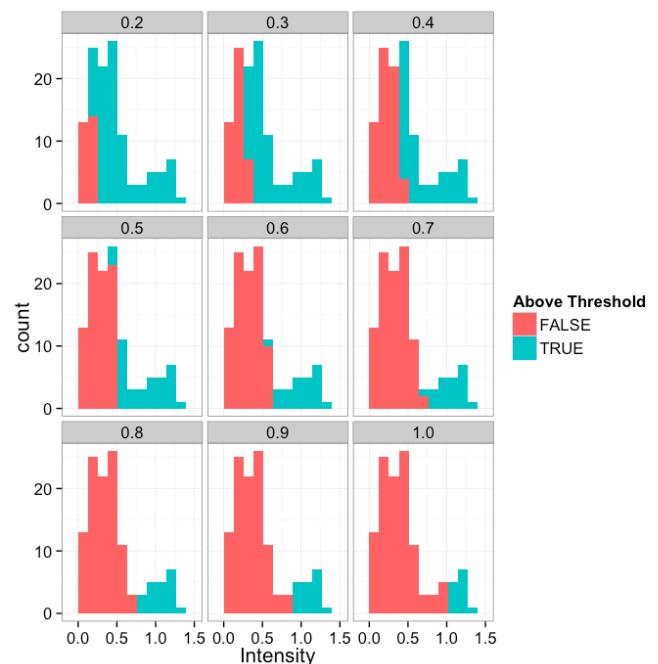


Applying the threshold is a deceptively simple operation

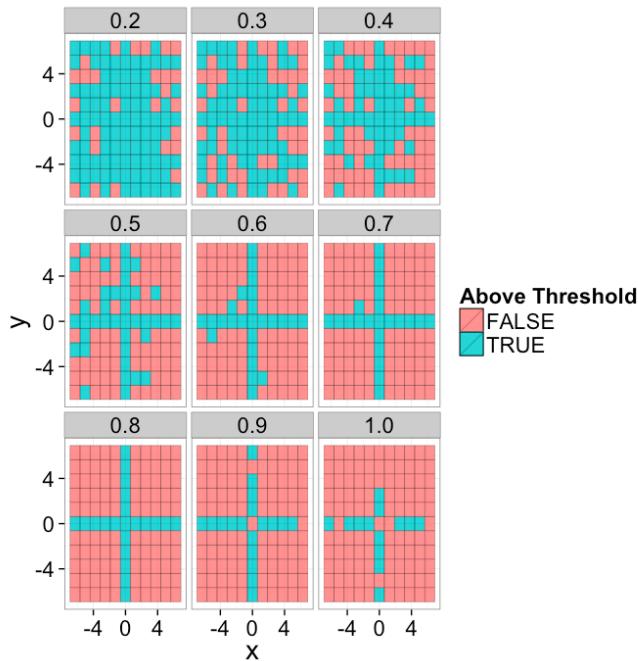
$$I(x, y) = \begin{cases} 1, & f(x, y) \geq 0.5 \\ 0, & f(x, y) < 0.5 \end{cases}$$



## Various Thresholds

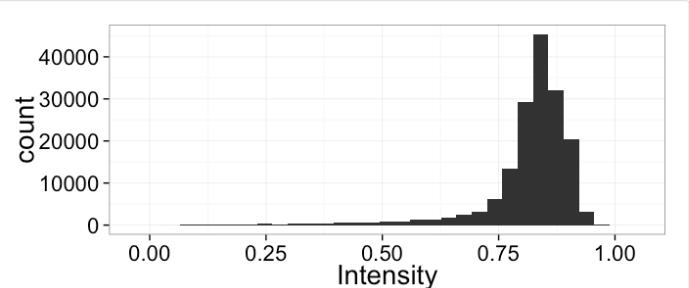
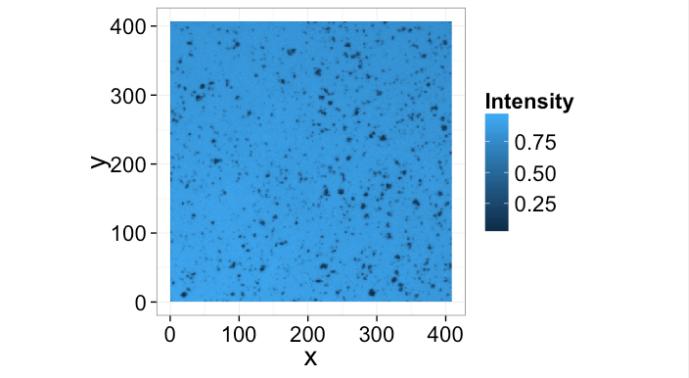


+/- R Code



## Segmenting Cells

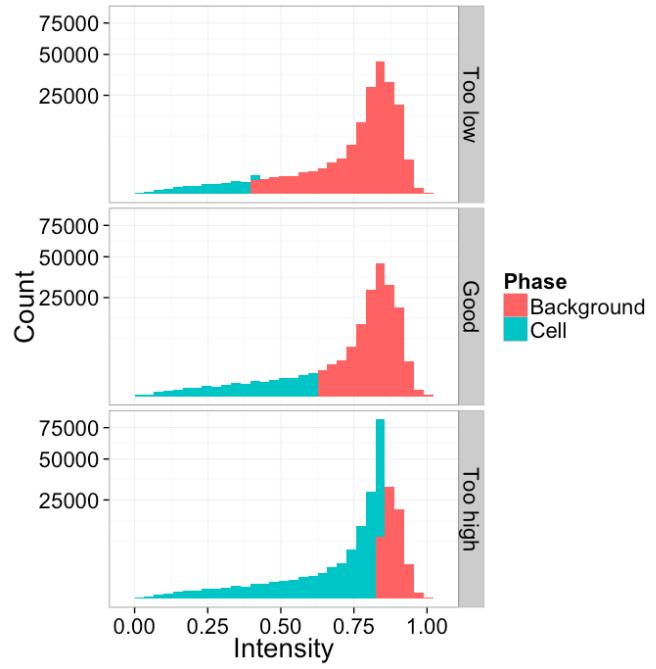
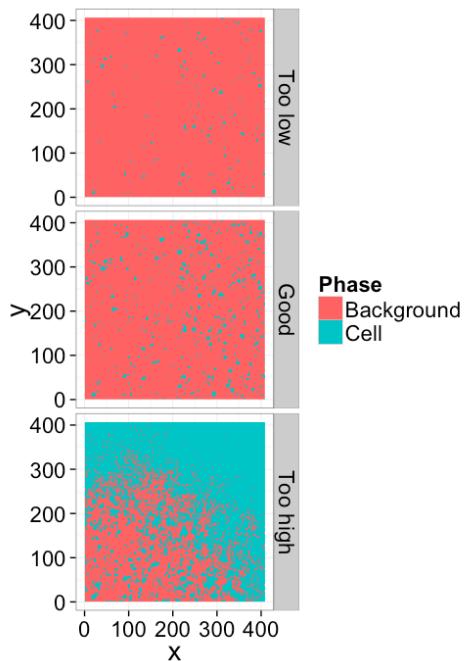
+/- R Code



- We can perform the same sort of analysis with this image of cells
- This time we can derive the model from the basic physics of the system
  - The field is illuminated by white light of nearly uniform brightness

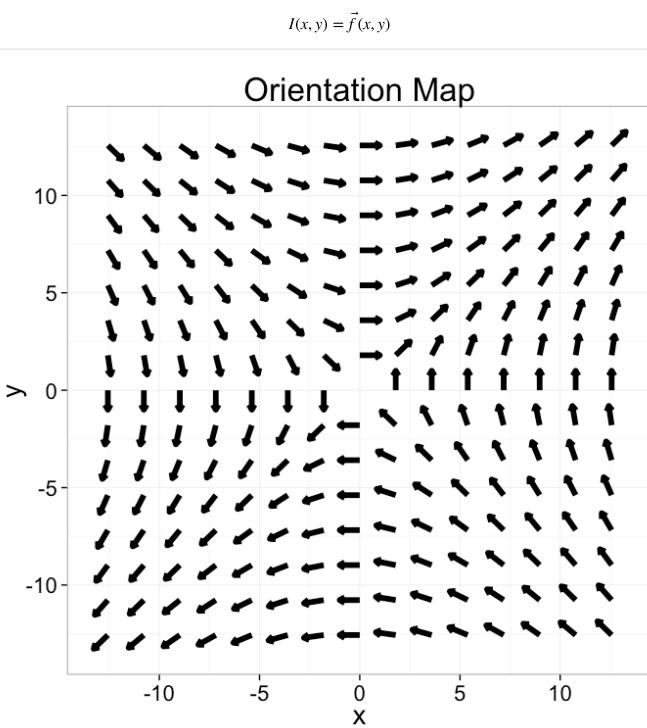
- Cells absorb light causing darker regions to appear in the image
- Lighter regions have no cells
- Darker regions have cells

## Different Threshold Values

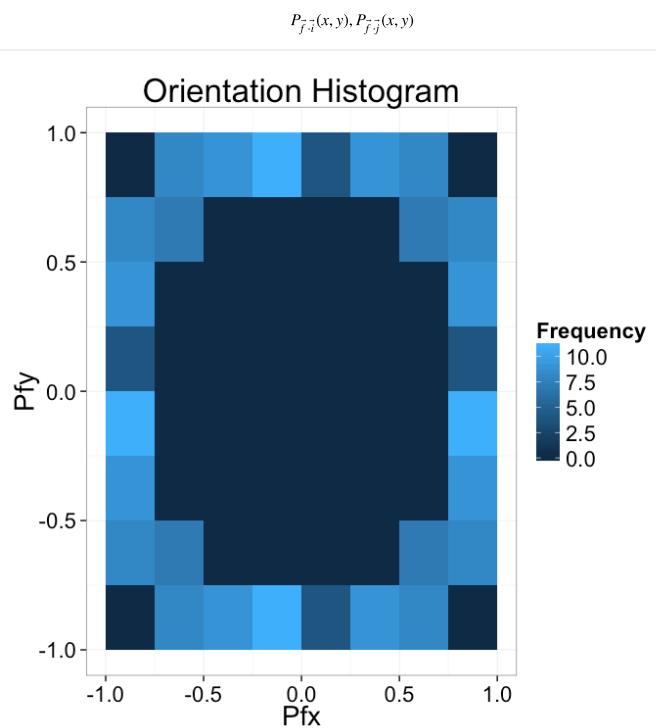


## Other Image Types

While scalar images are easiest, it is possible for any type of image



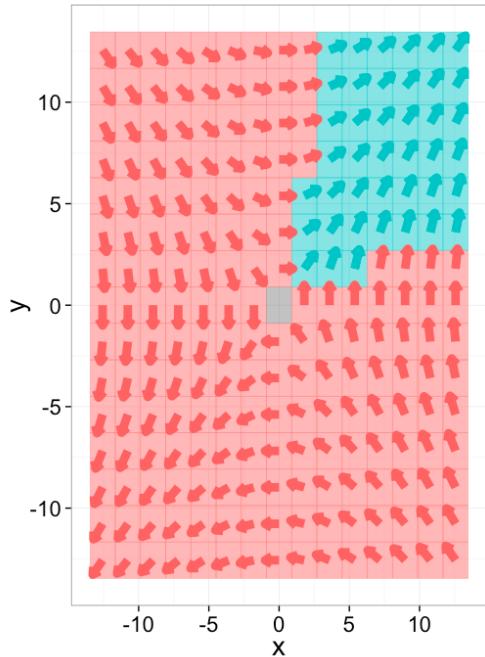
The intensity can be described with two separate or a single joint probability density function



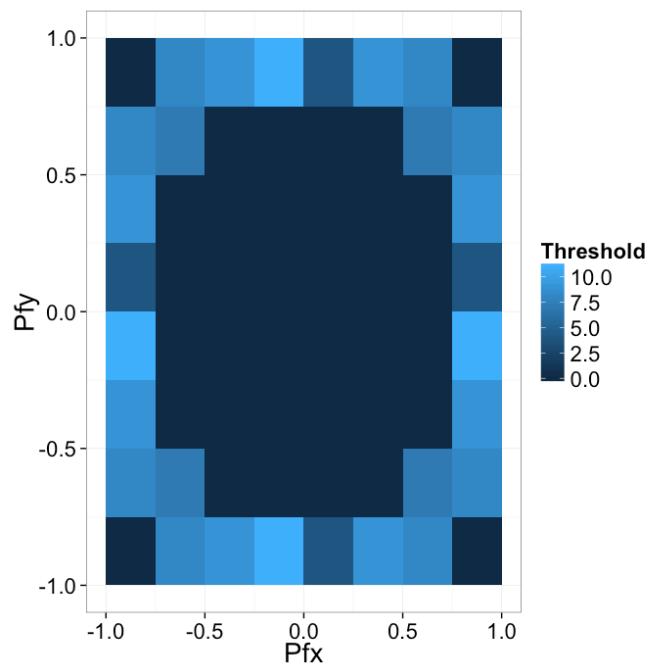
## Applying a threshold

A threshold is now more difficult to apply since there are now two distinct variables to deal with. The standard approach can be applied to both

$$I(x, y) = \begin{cases} 1, & \tilde{f}_x(x, y) \geq 0.25 \text{ and} \\ & \tilde{f}_y(x, y) \geq 0.25 \\ 0, & \text{otherwise} \end{cases}$$



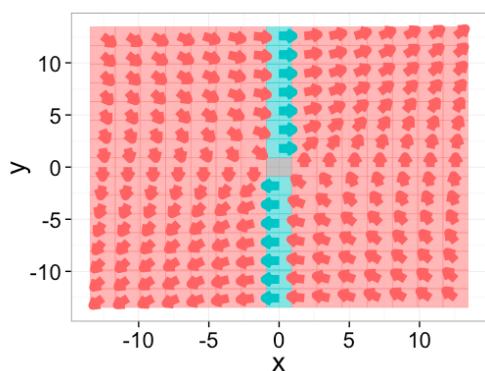
This can also be shown on the joint probability distribution as



## Applying a threshold

Given the presence of two variables; however, more advanced approaches can also be investigated. For example we can keep only components parallel to the x axis by using the dot product.

$$I(x, y) = \begin{cases} 1, & \tilde{f}(x, y) \cdot \vec{i} = 1 \\ 0, & \text{otherwise} \end{cases}$$

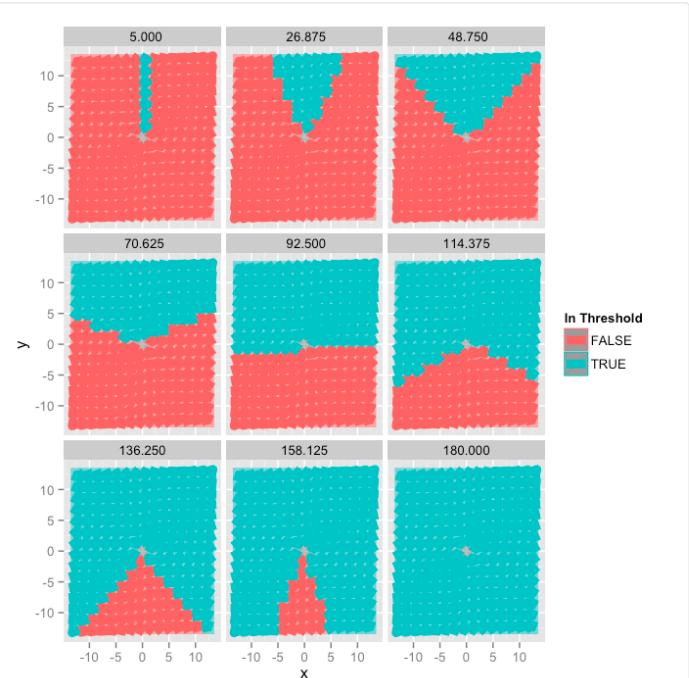


## Looking at Orientations

We can tune the angular acceptance by using the fact

$$\vec{x} \cdot \vec{y} = \|\vec{x}\| \|\vec{y}\| \cos(\theta_{x \rightarrow y})$$

$$I(x, y) = \begin{cases} 1, & \cos^{-1}(\tilde{f}(x, y) \cdot \vec{i}) \leq \theta^* \\ 0, & \text{otherwise} \end{cases}$$

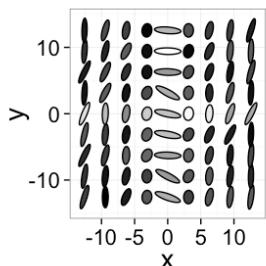


+/- R Code

## Other Image Types

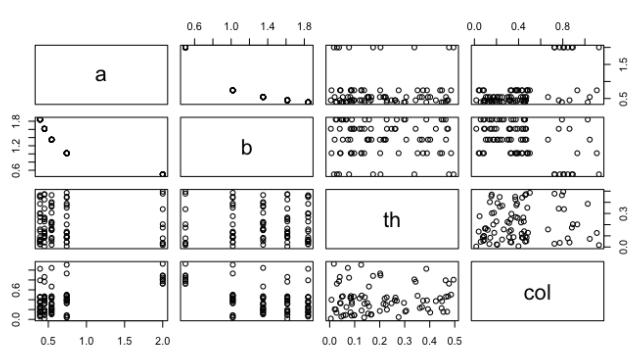
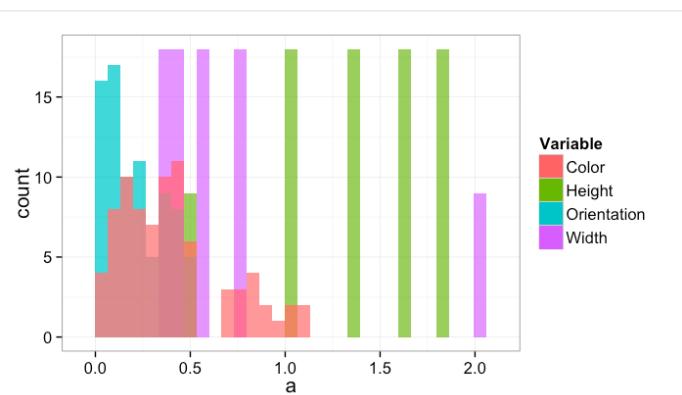
Going beyond vector the possibilities for images are limitless. The following shows a tensor plot with the tensor represented as an ellipse.

$$I(x, y) = \hat{f}(x, y)$$



Once the variable count is above 2, individual density functions and a series of cross plots are easier to interpret than some multidimensional density hypervolume.

+/- R Code

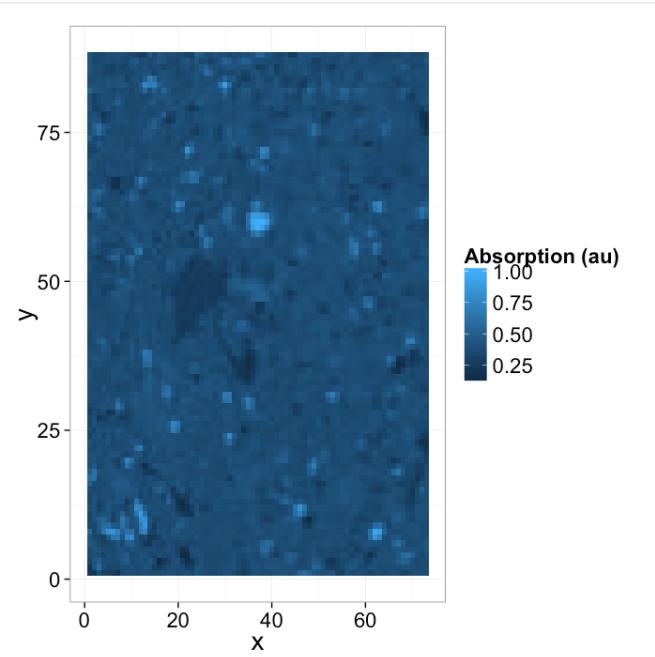


## Multiple Phases: Segmenting Shale

% Shale provided from Kanitpanyacharoen, W. (2012). Synchrotron X-ray Applications Toward an Understanding of Elastic Anisotropy.

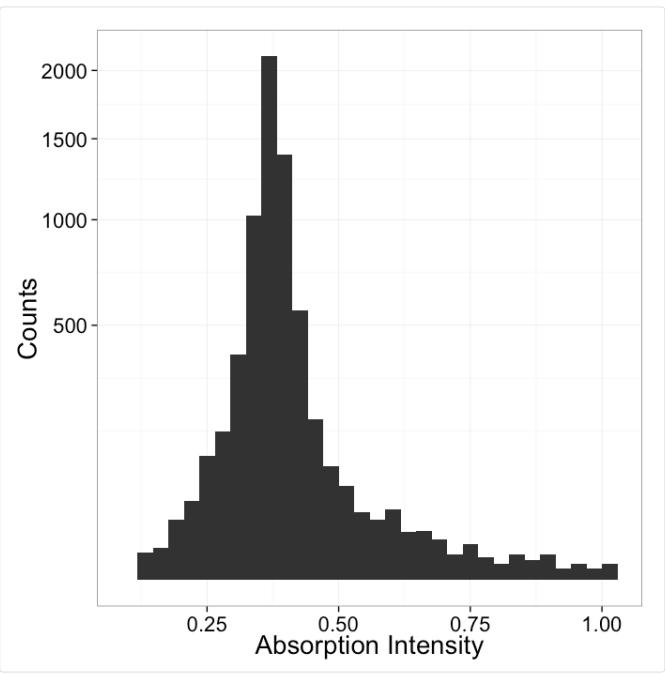
- Here we have a shale sample measured with X-ray tomography with three different phases inside (clay, rock, and air).
- The model is that because the chemical composition and density of each phase is different they will absorb different amounts of x-rays and appear as different brightnesses in the image

+/- R Code



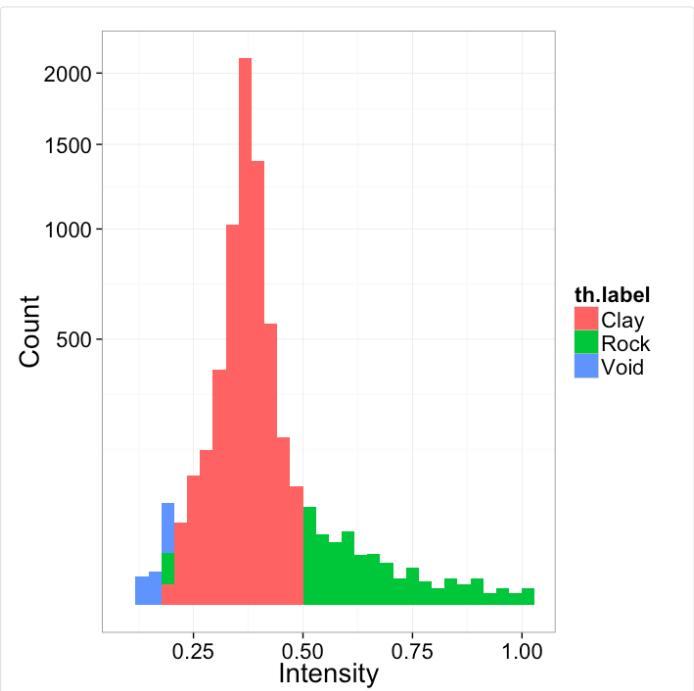
Ideally we would derive 3 values for the thresholds based on a model for the composition of each phase and how much it absorbs, but that is not always possible or practical.

- While there are 3 phases clearly visible in the image, the histogram is less telling (even after being re-scaled).



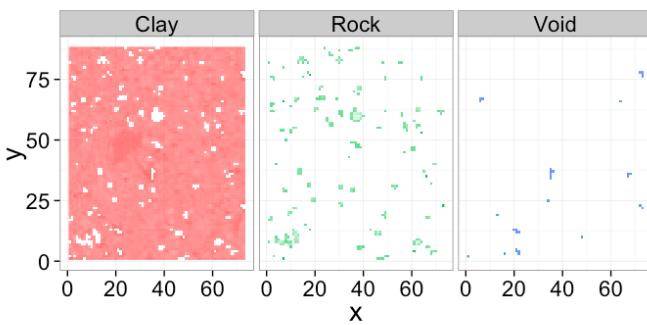
## Multiple Segmentations

For this exercise we choose arbitrarily 3 ranges for the different phases and perform visual inspection



The relation can explicitly be written out as

$$I(x) = \begin{cases} \text{Void}, & 0 \leq x \leq 0.2 \\ \text{Clay}, & 0.2 < x \leq 0.5 \\ \text{Rock}, & 0.5 < x \end{cases}$$



## Implementation

The implementations of basic thresholds and segmentations is very easy since it is a unary operation of a single image

$$f(I(\vec{x}))$$

In mathematical terms this is called a map and since it does not require information from neighboring voxels or images it can be calculated for each point independently (*parallel*). Filters on the other hand almost always depend on neighboring voxels and thus the calculations are not as easy to separate.

## Implementation Code

### Matlab

The simplest is a single threshold in Matlab:

```
thresh_img = gray_img > thresh
```

A more complicated threshold:

```
thresh_img = (gray_img > thresh_a) & (gray_img > thresh_b)
```

### Python (numpy)

```
thresh_img = gray_img > thresh
```

### Python

```
thresh_img = map(lambda gray_val: gray_val>thresh,
                 gray_img)
```

### Java

```
boolean[] thresh_img = new boolean[x_size*y_size*z_size];
for(int x=x_min;x<x_max;x++)
    for(int y=y_min;y<y_max;y++)
        for(int z=z_min;z<z_max;z++) {
            int offset=(z*y_size+y)*x_size+x;
            thresh_img[offset]=gray_img[offset]>thresh;
        }
    }
```

### In C/C++

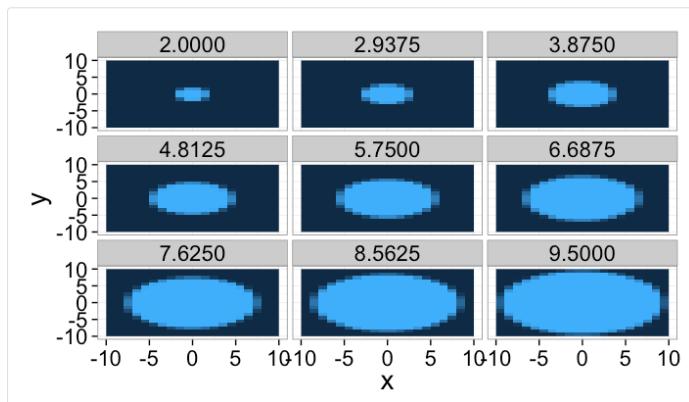
```
bool* thresh_img = malloc(x_size*y_size*z_size * sizeof (bool));
for(int x=x_min;x<x_max;x++)
    for(int y=y_min;y<y_max;y++)
        for(int z=z_min;z<z_max;z++) {
            int offset=(z*y_size+y)*x_size+x;
            thresh_img[offset]=gray_img[offset]>thresh;
        }
    }
```

## Pitfalls with Segmentation

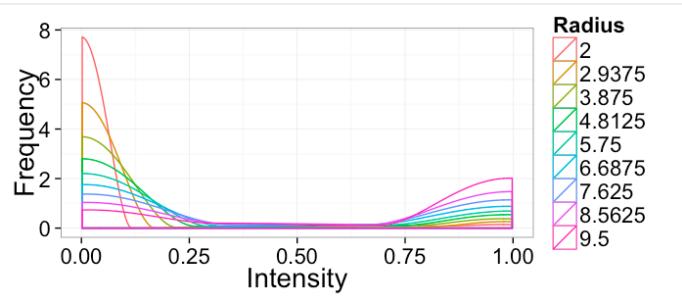
type: alert

## Partial Volume Effect

- The partial volume effect (<http://bit.ly/1mW7kdP>) is the name for the effect of discretization on the image into pixels or voxels.
- Surfaces are complicated, voxels are simple boxes which make poor representations
- Many voxels are only partially filled, but only the voxels on the surface
- Removing the first layer alleviates issue



+/- R Code



- Shown as a function of radius

+/- R Code

Radius	Mean Intensity	Sd Intensity
2.0000	0.0311250	0.1623387
2.9375	0.0677250	0.2369905
3.8750	0.1177250	0.3061084
4.8125	0.1822250	0.3687992
5.7500	0.2601250	0.4196009
6.6875	0.3511250	0.4567461
7.6250	0.4569250	0.4762676
8.5625	0.5761250	0.4690413
9.5000	0.7072841	0.4258651

## Morphology

Returning to the original image of a cross

Like filtering the assumption behind morphological operations are

- nearby voxels in **real** images are related / strongly correlated with one another
- noise and imaging artifacts are less spatially correlated.

Therefore these imaging problems can be alleviated by adjusting the balance between local and neighborhood values.

## Fundamentals: Neighborhood

A neighborhood consists of the pixels or voxels which are of sufficient proximity to a given point. There are a number of possible definitions which largely affect the result when it is invoked.

- A large neighborhood performs operations over larger areas / volumes
  - Computationally intensive
  - Can smooth out features
- A small neighborhood performs operations over small areas / volumes
  - Computationally cheaper
  - Struggles with large noise / filling large holes

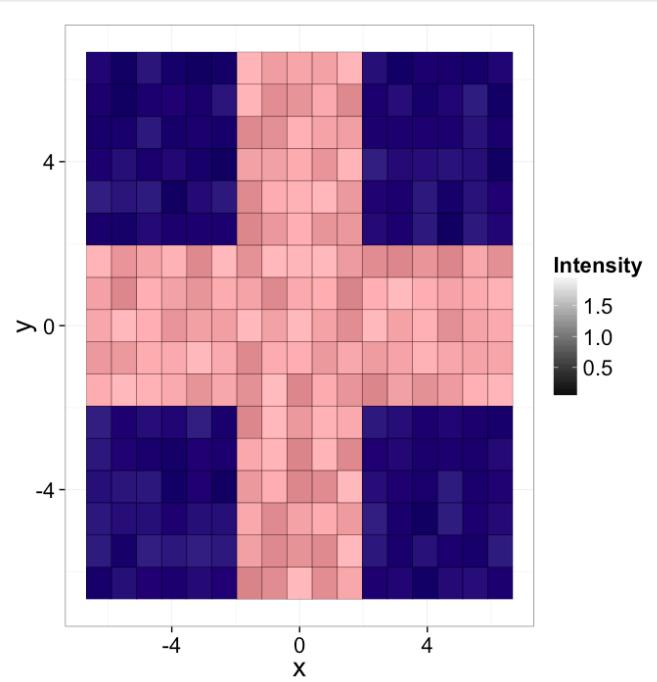
The neighborhood is important for a large number of image and other (communication, mapping, networking) processing operations:

- filtering
- morphological operations
- component labeling
- distance maps
- image correlation based tracking methods

## Fundamentals: Neighbors in 2D

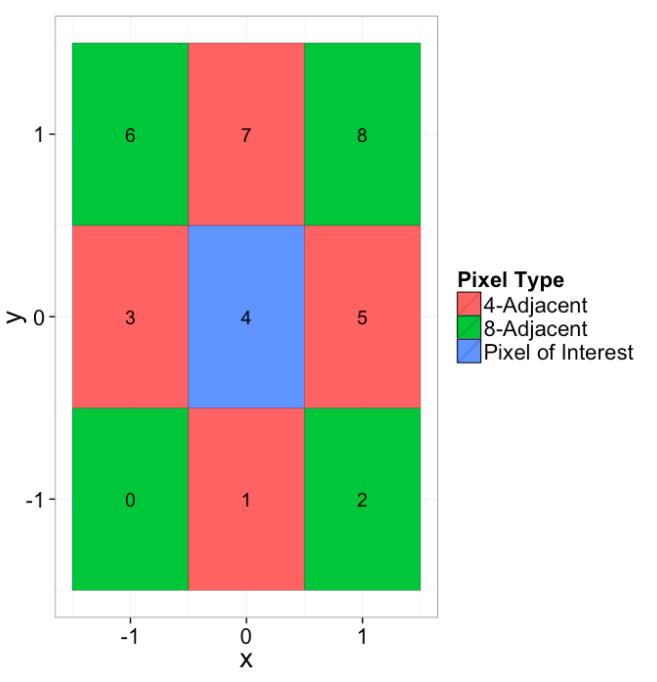
For standard image operations there are two definitions of neighborhood. The 4 and 8 adjacent neighbors shown below. Given the blue pixel in the center the red are the 4-adjacent and the red and green make up the 8-adjacent.

+/- R Code



+/- R Code

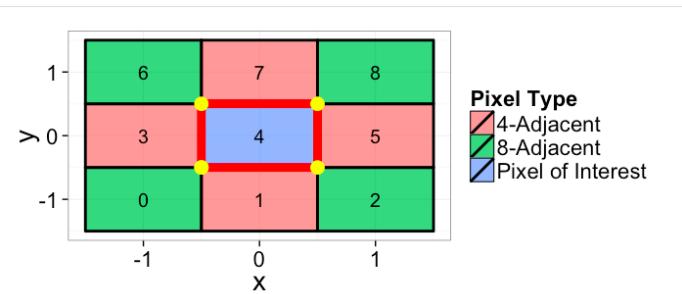
We can now utilize information from neighborhood voxels to improve the results. These steps are called morphological operations.



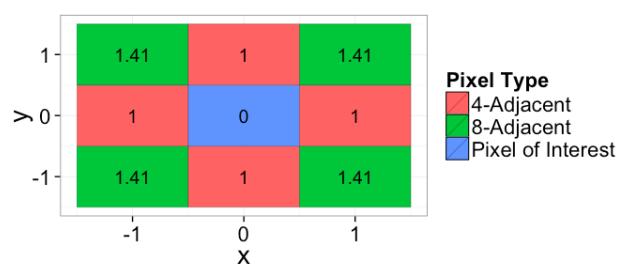
## Formal Neighborhood Definitions

Formally these have two effectively equivalent, but different definitions.

- Pixels which share a face (red line) with the pixel
- Pixels which share a vertex (yellow dot) with the pixel



- Pixels which are distance 1 from the pixel
- Pixels which are distance  $\sqrt{2}$  from the pixel

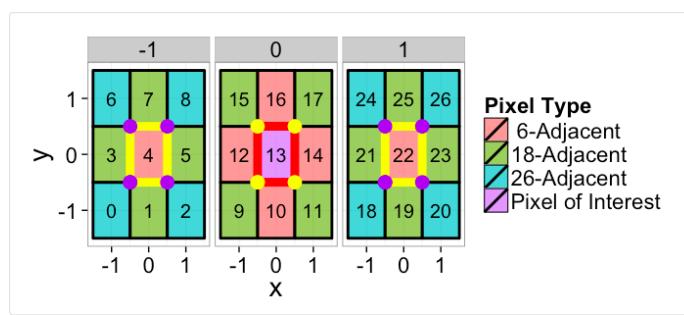


+/- R Code

## Formal Neighborhood Definitions in 3D

In 3D there is now an additional group to consider because of the extra-dimension

- Voxels which share a face (red line) with the voxel (6-adjacent)
- Voxels which share an edge (yellow dot) with the voxel (18-adjacent)
- Voxels which share a vertex (purple dot) with the voxel (26-adjacent)



+/- R Code

## Erosion and Dilation

### Erosion

If any of the voxels in the neighborhood are 0/false than the voxel will be set to 0

- Has the effect of peeling the surface layer off of an object

### Dilation

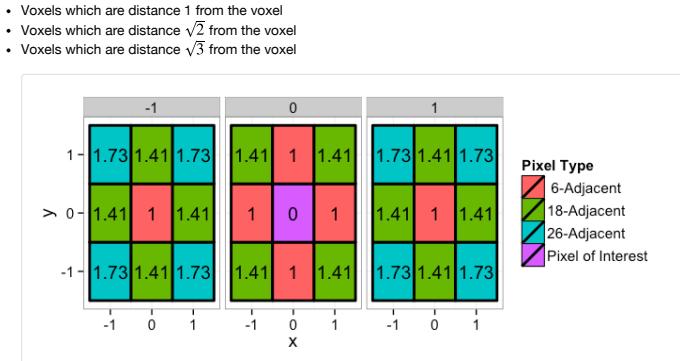
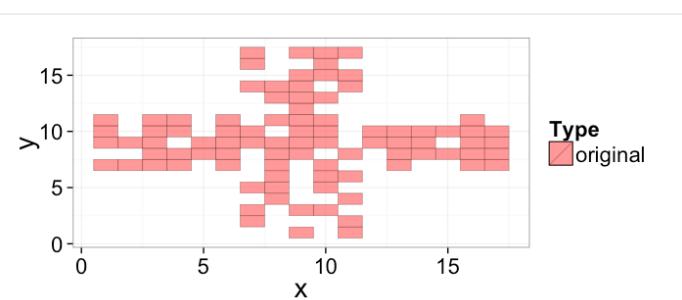
If any of the voxels in the neighborhood are 1/true then the voxel will be set to 1

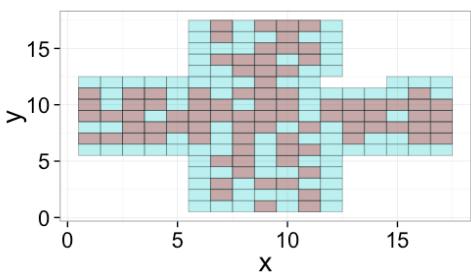
- Has the effect of adding a layer onto an object (dunking a strawberry in chocolate, adding a coat of paint to a car)

## Applied Erosion and Dilation

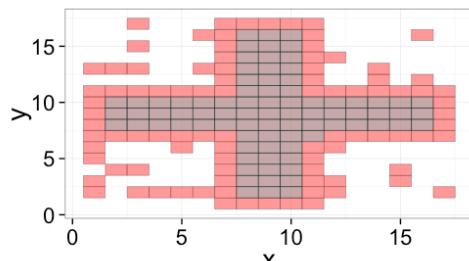
### Dilation

Taking an image of the cross at a too-high threshold, we show how dilation can be used to recover some of the missing pixels





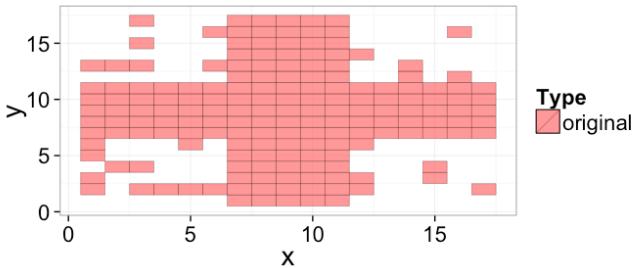
Type  
original  
post dilation



Type  
original  
post erosion

## Erosion

Taking an image of the cross at a too-low threshold, we show how erosion can be used to remove some of the extra pixels



Type  
original

+/- R Code

## Opening and Closing

### Opening

An erosion followed by a dilation operation

- Peels a layer off and adds a layer on
- Very small objects and connections are deleted in the erosion and do not return in the dilation
- A cube larger than several voxels will have the exact same volume after (conservative)

### Closing

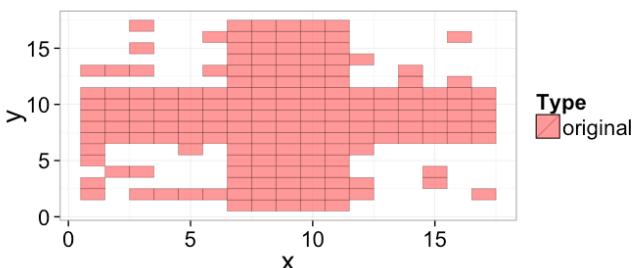
A dilation followed by an erosion operation

- Adds a layer and then peels a layer off
- Objects that are very close are connected when the layer is added and stay connected when the layer is removed thus the image is closed
- A cube larger than one voxel will have the exact same volume after (conservative)

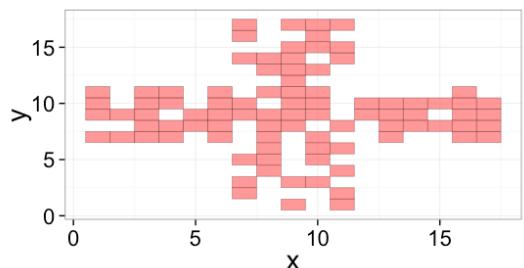
## Applied Opening and Closing

### Opening

Taking an image of the cross at a too-low threshold, we show how opening can be used to remove some of the extra pixels

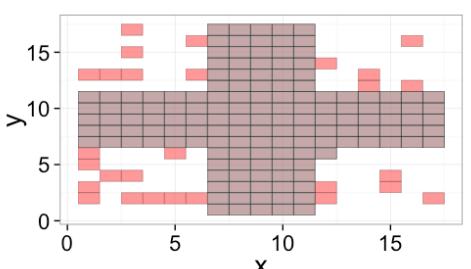


Type  
original

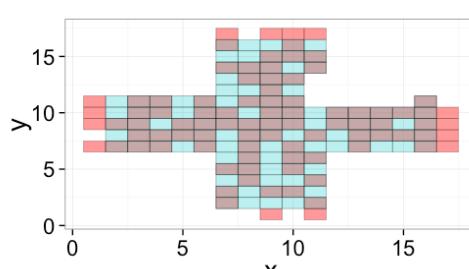


Type  
original

+/- R Code



Type  
original  
post opening



Type  
original  
post closing

## Closing

Taking an image of the cross at a too-high threshold, we show how closing can be used to recover some of the missing pixels

## Applying Morphological Tools

For many applications morphology appears to be the same as filter

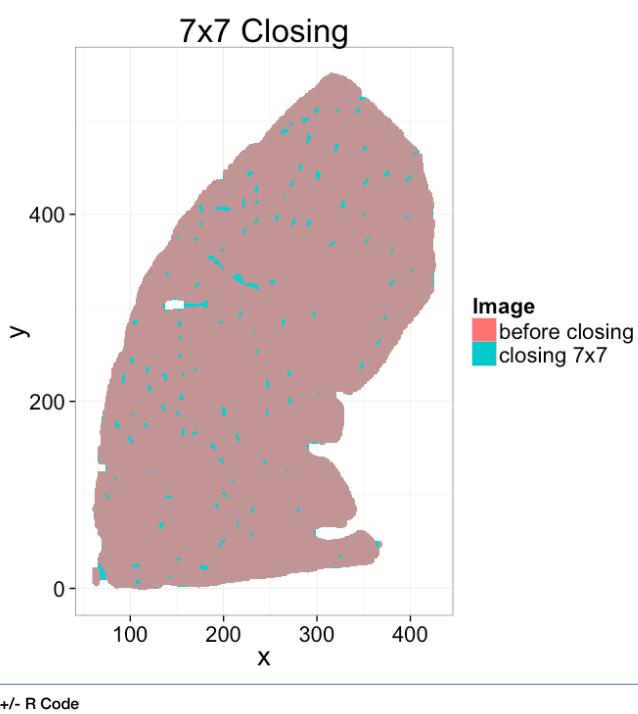
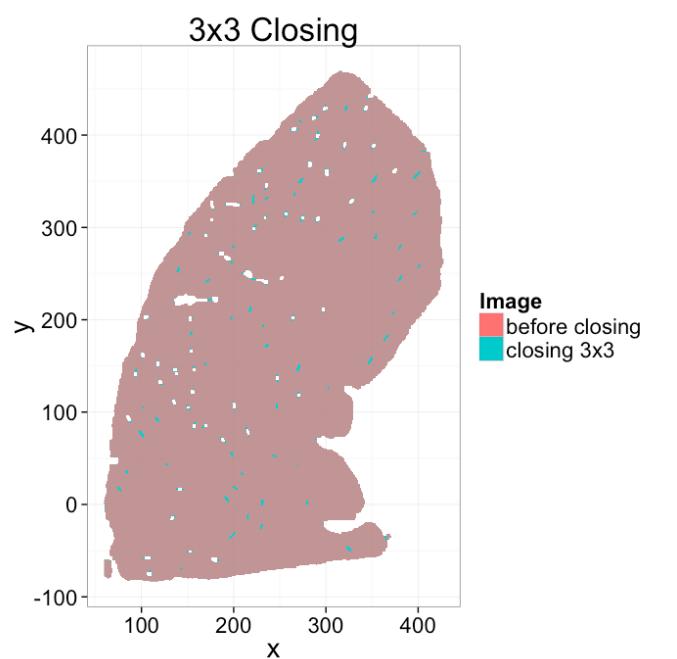
- The key difference is the avalanche or non-linear nature of the results
  - A single off voxel can turn its entire neighborhood off (erosion)
  - A single on voxel can turn its entire neighborhood on (dilation)

- The effects of this are most pronounced when performed iteratively
- This is very useful for filling holes, connecting objects, creating masks

A segmented slice taken from a cortical bone sample. The dark is the calcified bone tissue and the white are holes in the image

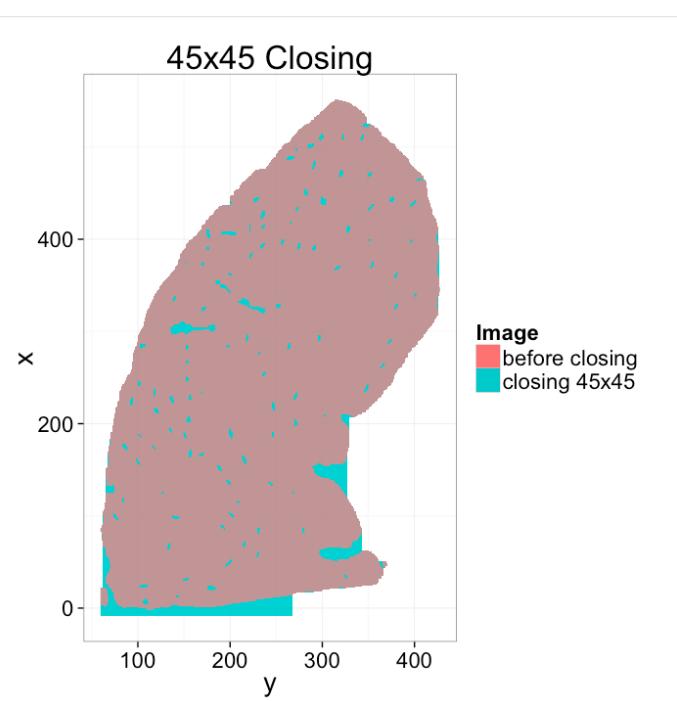


## Filling Holes / Creating Masks

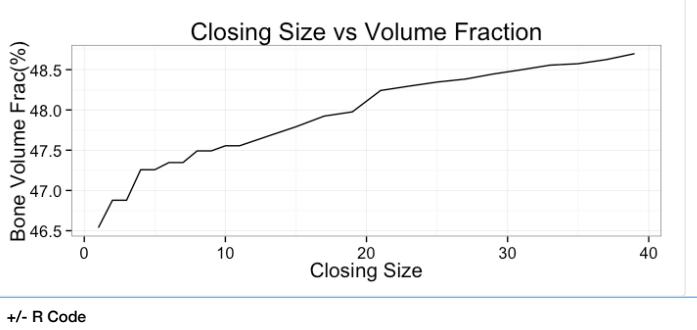


## Filling Holes / Creating Masks

Applying the closing with even larger windows will close everything but being to destroy the underlying structure of the mask



We can characterize this by examining the dependency of the closing size and the volume fraction (note scale)



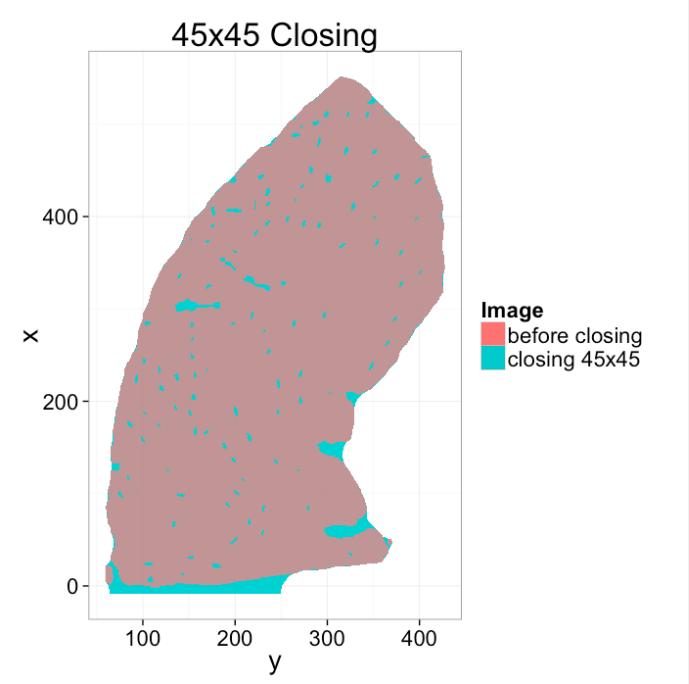
+/- R Code

- Be careful when using large area opening / closing operations (always check the images)
- Noise and defects in images can be amplified with these larger operations

## Different Kernels

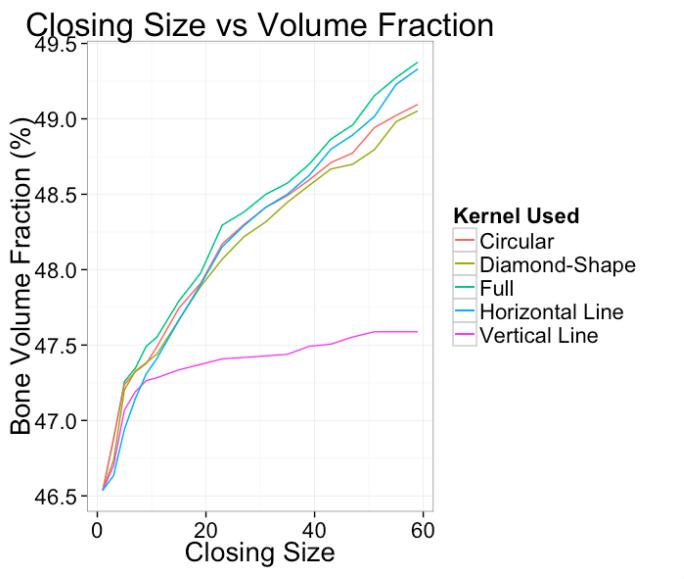
Using a better kernel shape (circular, diamond shaped, etc) the artifacts from morphological operations can be reduced

+/- R Code



We can characterize this by examining the dependency of the closing size and the volume fraction (note scale)

+/- R Code



- Alternative techniques
  - Convex Hull ([http://en.wikipedia.org/wiki/Convex\\_hull](http://en.wikipedia.org/wiki/Convex_hull)) - assuming convex shapes
  - Flood Filling ([http://en.wikipedia.org/wiki/Flood\\_fill](http://en.wikipedia.org/wiki/Flood_fill)) - fill connected pixels like in Microsoft Paint

## Segmenting Fossils

Taken from the BBC Documentary First Life  
(<http://www.bbc.co.uk/programmes/b00vw49d>) by David Attenborough

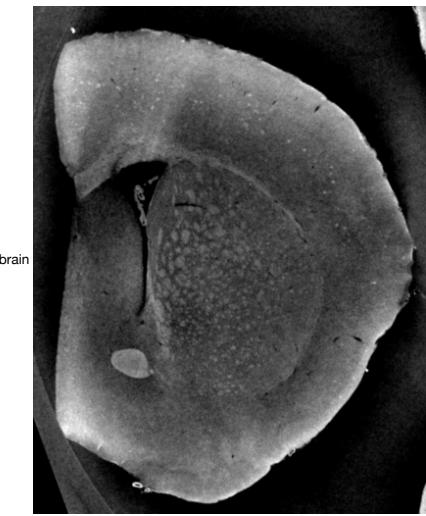


- Gut Data
  - Slice 182 you can make out the intestine track
- Teeth Data
  - Explore the sample and apply a threshold to locate the teeth using the 3D viewer

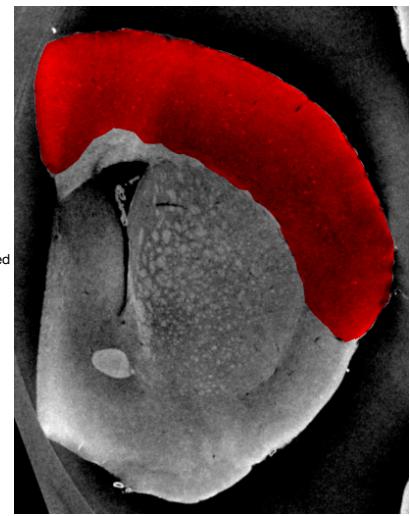
## Quantifying Alzheimers: Segment the Cortex

Courtesy of Alberto Alfonso

- Understand the progress of Alzheimer's disease as it relates to plaque formation in different regions of the



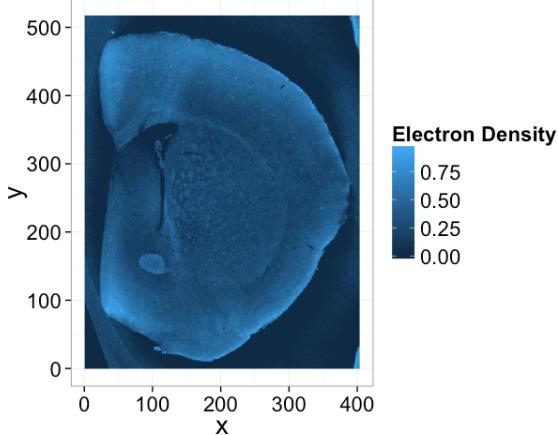
- First identify different regions



## Quantifying Alzheimers: Segment the Cortex

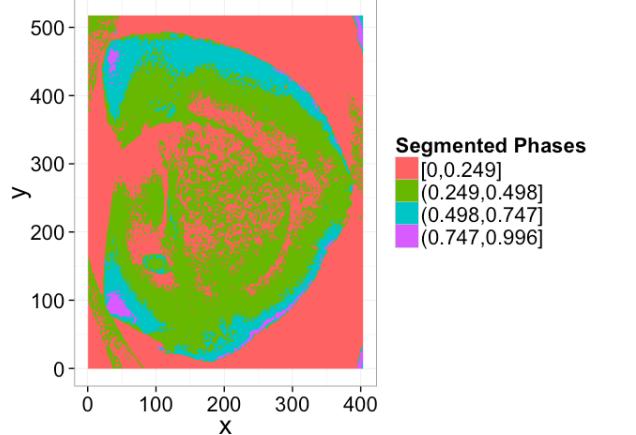
- The cortex is barely visible to the human eye
- Tiny structures hint at where cortex is located

[+/- R Code](#)



- A simple threshold is insufficient to finding the cortical structures
- Other filtering techniques are unlikely to magically fix this problem

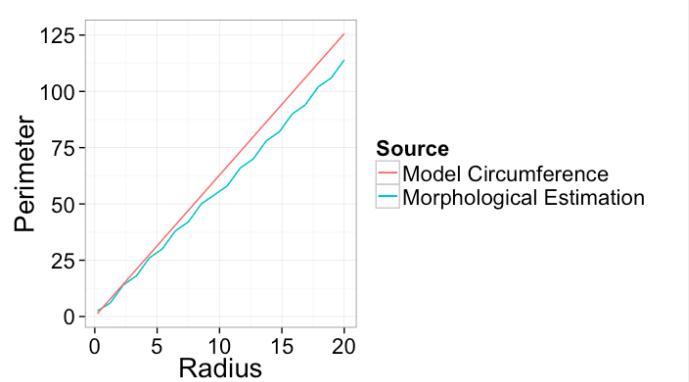
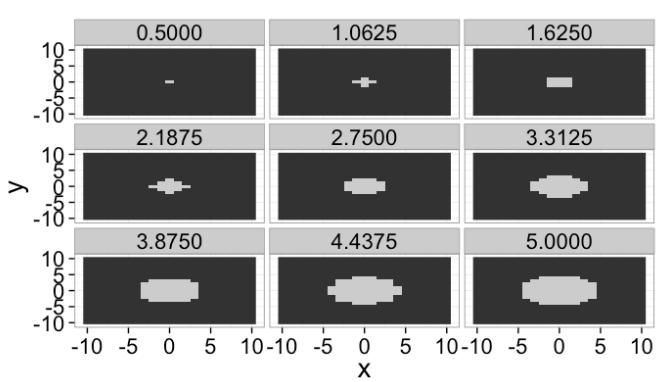
[+/- R Code](#)



## Surface Area / Perimeter

We see that the dilation and erosion affects are strongly related to the surface area of an object: the more surface area the larger the affect of a single dilation or erosion step.

[+/- R Code](#)



+/- R Code