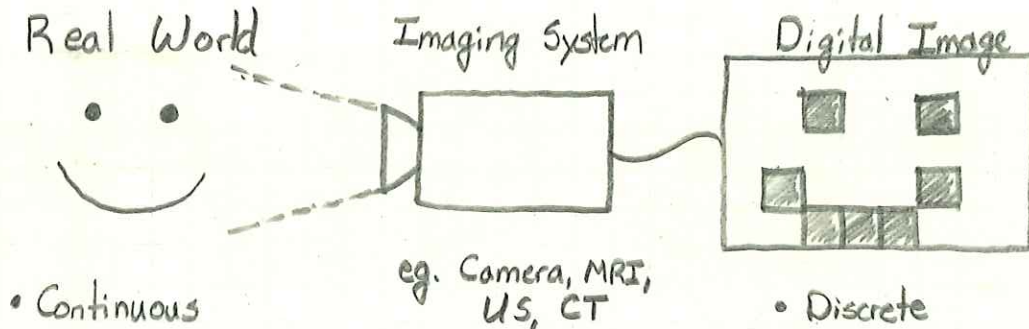


BMEN 509 - Lab #1

Bryce Besler

Objectives

- Representation of images
- Work with images in numpy
- Understand and use fundamental image operations

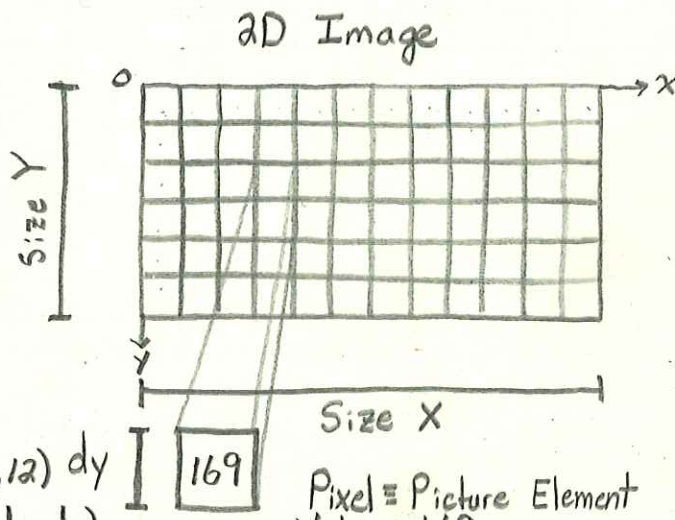
Images as Representations of the Real World

Digital images are a finite, approximate representation of the real, continuous world. How to create these images is the bulk of the course. However, we notice two important properties of digital images:

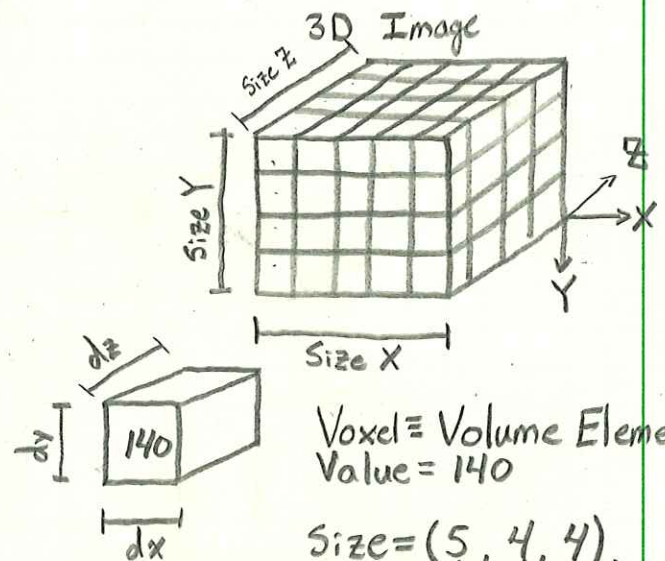
- 1) Discrete in Space (quanta = voxel or pixel)
- 2) Discrete in Intensity (represented by a digital number)

Image Representation

Images are composed of scalars (real numbers,  $\mathbb{R}$ ) arranged in a regular grid. They contain the following properties: spacing (resolution), size, and dimensionality.



Size = (6, 12) dy  
Spacing = (dx, dy)  
Area = dx · dy (of pixel)

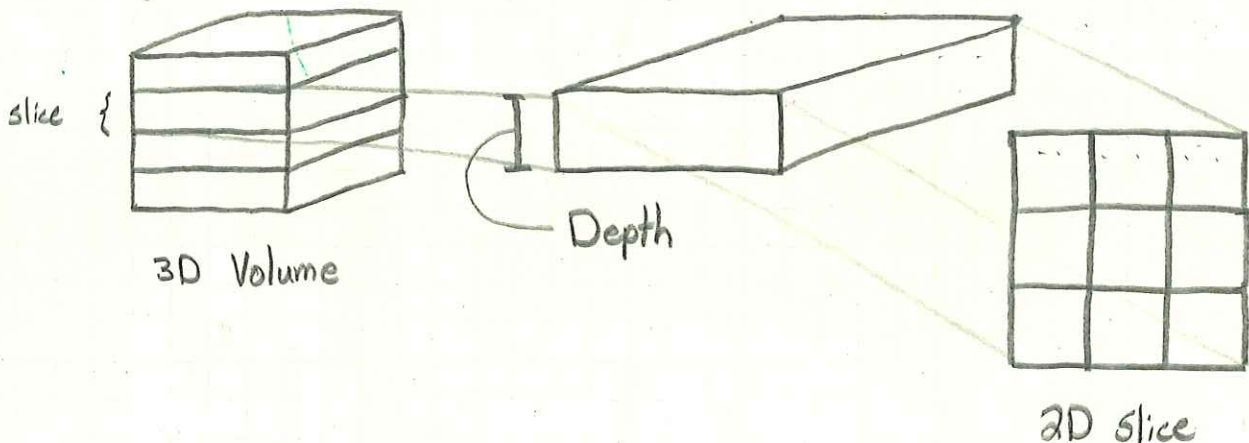


Size = (5, 4, 4)  
Spacing = (dx, dy, dz)  
Volume = dx · dy · dz

See Example 1 for application of the image basics

## 2D vs 3D Imaging

Historically, medical imaging has been dominated by two dimensional images. This includes X-ray and original MR and CT images. With technology advances, 3D and even 4D images exist. The 4th dimension is often time, such as live imaging of the heart. We can conceptualize 3D images as a stack of 2D images.



## Image Statistics

Often, we need to compare images. Qualitative images as tone, texture, quality, and structure are useful but vary between individuals. Quantitative metrics are important to provide objective measures.

$$I = \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

$$\text{mean}(I) = \frac{0+0+0+0+1+0+0+0+0}{9} = 0.11$$

$$\text{std}(I) = \sqrt{\frac{1}{n-1} \sum_{j=1}^n (I(j) - \text{mean}(I))^2} = 0.33$$

In general, we will be interested in the signal-to-noise ratio.

$$\text{SNR}(I) = \frac{\text{mean}(I)}{\text{std}(I)} = \frac{\mu}{\sigma}$$

We may also be interested in the CNR.

$$\begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

A points to the center cell (1), B points to a corner cell (0).

$$\text{Contrast}(A, B) = |I(A) - I(B)| = |1 - 0| = 1$$

$$\text{CNR}(A, B) = \frac{\text{Contrast}(A, B)}{\text{std}(I)} = \frac{1}{0.33} = 3$$



Operating on Images

We can operate on images just like we operate on images.

1	2	3
4	5	6
7	8	9

 $\times 2 =$ 

2	4	6
8	10	12
14	16	18

$$I \cdot 2 = J$$

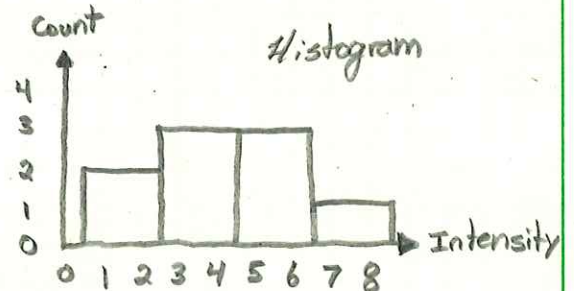
Image Histograms

It wouldn't be uncommon to see images of size  $512 \times 512$ . That is 262 144 pixels! Mammograms can be  $2394 \times 2394$ ! High resolution histology can be  $10000 \times 10000$ ! We can't just print these images. We need tools to throw away uninteresting data.

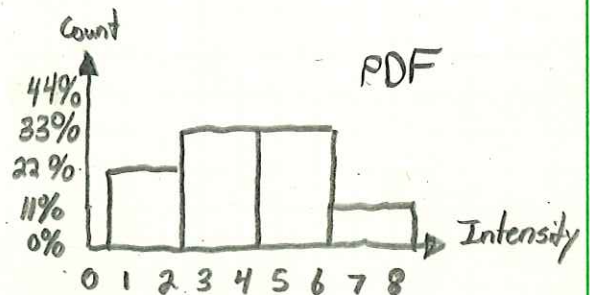
Histograms have many uses. Here, I will present them as a tool for understanding and comparing images. Histograms take the grey values in an image, quantize them in bins, and plot the number of elements in each bin.

1	2	3
4	5	6
4	6	7

Bin	Count
1-2	2
3-4	3
5-6	3
7-8	1



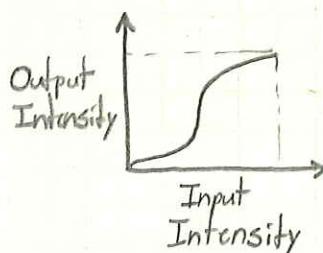
We can create a probability density function (PDF) by dividing each count by the number of elements



How do you think you would get a cumulative distribution function (CDF) from this data?

## Image Transfer Functions

We often want to apply a function pixel-wise to an image. This can be done to enhance structures of interest, visualize tissues better, or equalize contrast. These pixelwise functions are called transfer functions.



$$\text{Output} = f(\text{Input})$$

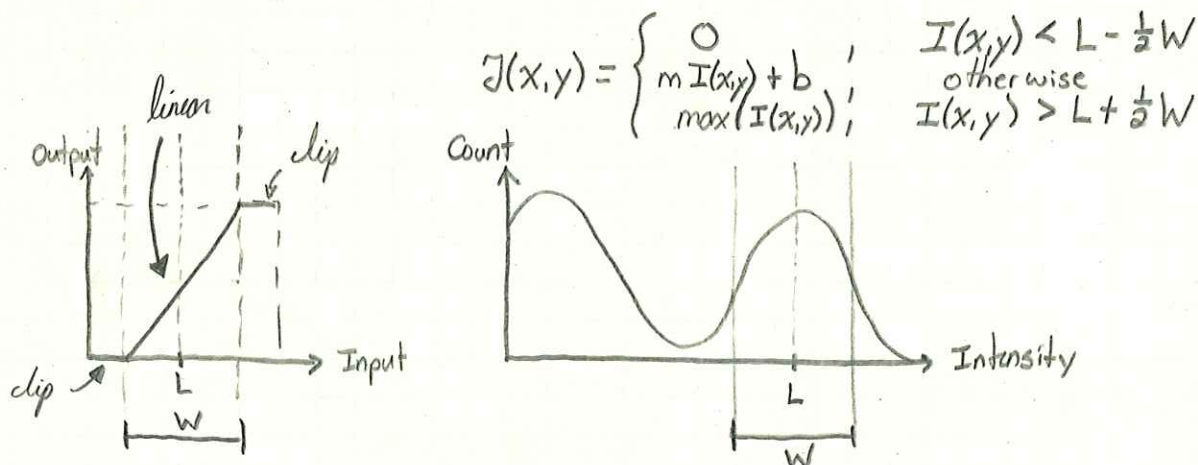
Possible functions include CDFs, linear equations, sigmoid, tanh, log, and anything else.

See Example 3 for histograms and transfer functions.

## Window/Level

A grayscale image is usually stored in a 16 bit unsigned integer. That means the image has  $2^{16} = 65,536$  distinct grayvalues. However, the human eye can only distinguish about 100 different grayvalues. To overcome this limitation, virtually all medical visualization software allows the user to "window and level" the image.

The transfer function for W/L is defined as:



We only visualize pixels between  $L - \frac{1}{2}W$  and  $L + \frac{1}{2}W$

$$m = \frac{\text{max} - \text{min}}{W}$$

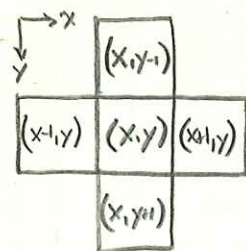
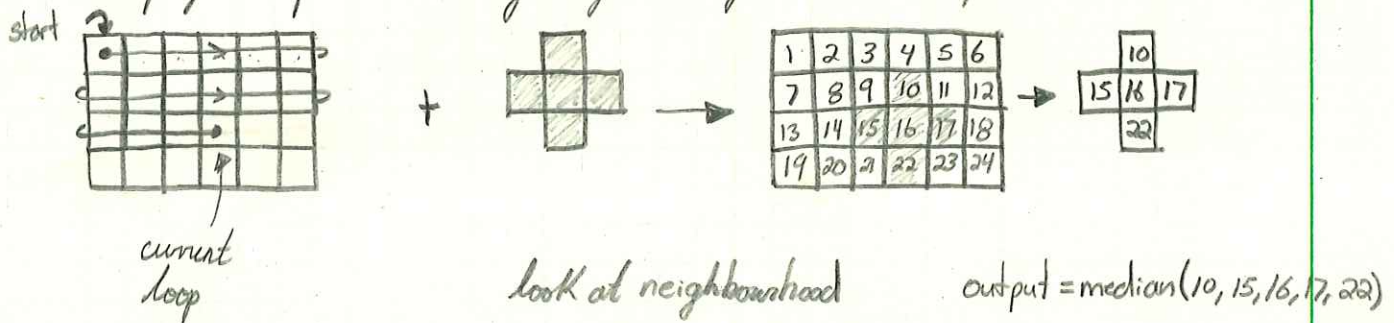
$$c = \text{max} - \frac{(\text{max} - \text{min})}{W} (L + \frac{1}{2}W)$$

You will be implementing this in lab 1!

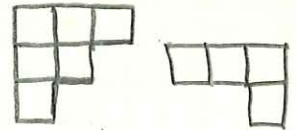


## Neighborhood Operators

An exceptional number of algorithms operate on the concept of neighborhoods. Similar to pixel-wise operators, neighborhood operators loop over an image but perform operation using neighbours of the current pixel.



This is a Kernel. In general, it can be any shape.



The indexing guides the implementation.

## Convolution

If the filter is a linear, time invariant system, it is known as convolution. To perform convolution, shift the kernel through the image applying point-wise multiplication and summation.

$$\begin{bmatrix} 8 & 8 & 9 \\ 8 & 9 & 7 \\ 2 & 8 & 9 \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = 8 \cdot 1 + 8 \cdot 1 + 9 \cdot 1 + 8 \cdot 1 + 9 \cdot 1 + 7 \cdot 1 + 2 \cdot 1 + 8 \cdot 1 + 9 \cdot 1 = 68$$

Average

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Gaussian

$$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Edge

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

See Exercise 4 for more details.


Non-Linear Filters

Many filters which are not LTI can be implemented in a similar fashion.

5	6	4	6	6	4
5	5	6	4	4	6
5	3	2	100	4	6
5	3	6	4	4	4

noise!

+



→ output = median (

	4	
2	100	4
	4	

)= 4

We removed the 100 from noise!