



# An Introduction to Images

CS6640/BIOENG6640/ECE6532

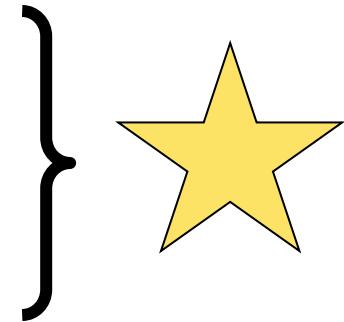
Ross Whitaker, Tolga Tasdizen

SCI Institute, School of Computing,  
Electrical and Computer Engineering

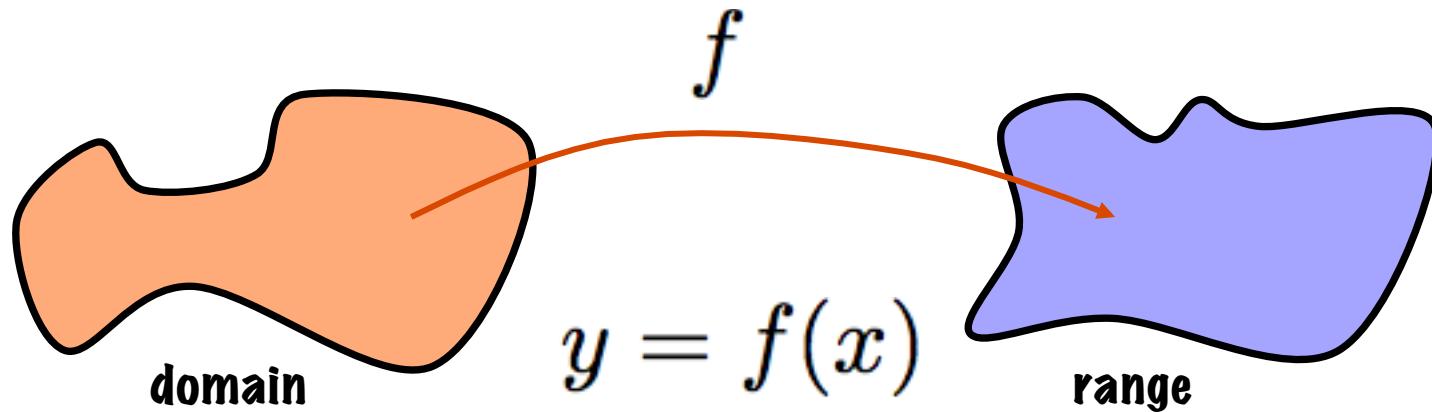
University of Utah

# What Is An *Digital Image*?

- A file you download from the web (e.g. image.jpg)
- What you see on the screen
- An array (regular grid) of data values
- A mapping from one domain to another
  - A discrete sampling (approximation) of a function



# Image As A Mapping (Function)



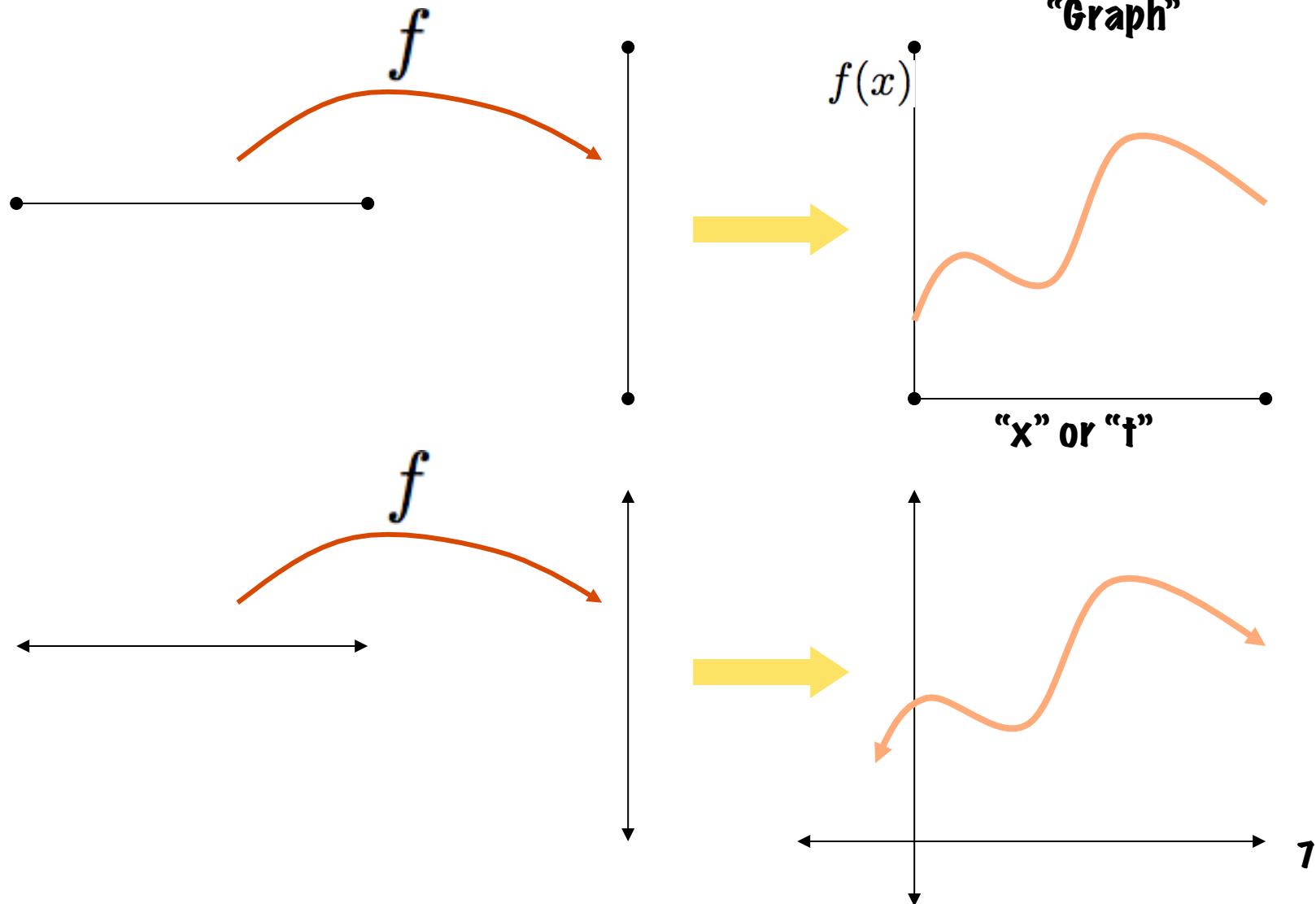
$$f : \mathcal{D} \mapsto \mathcal{R}$$

$$\mathcal{D} \subset \mathbb{R}^n \text{ and } \mathcal{R} \subset \mathbb{R}^m$$

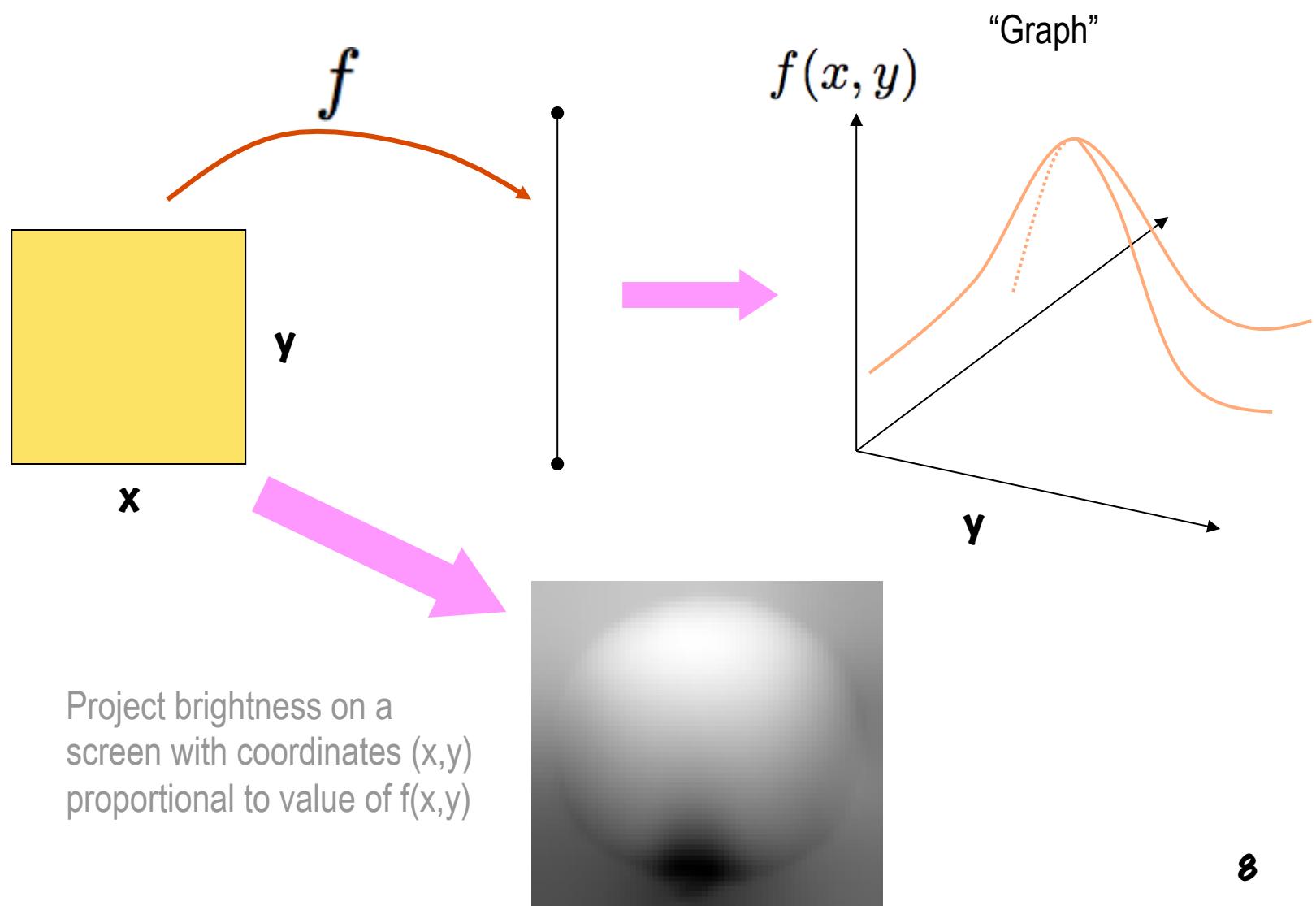
# Image As A Mapping: Issues

- Dimensionality of domain ( $n = ?$ )
- Dimensionality of range ( $m = ?$ )
- Typically use shorthand of  $R^n$  or  $R^m$
- Discrete or continuous
  - Discrete reasoning/math
  - Continuous math (calculus)  $\rightarrow$  discrete approximation
  - Issues for both domain and range

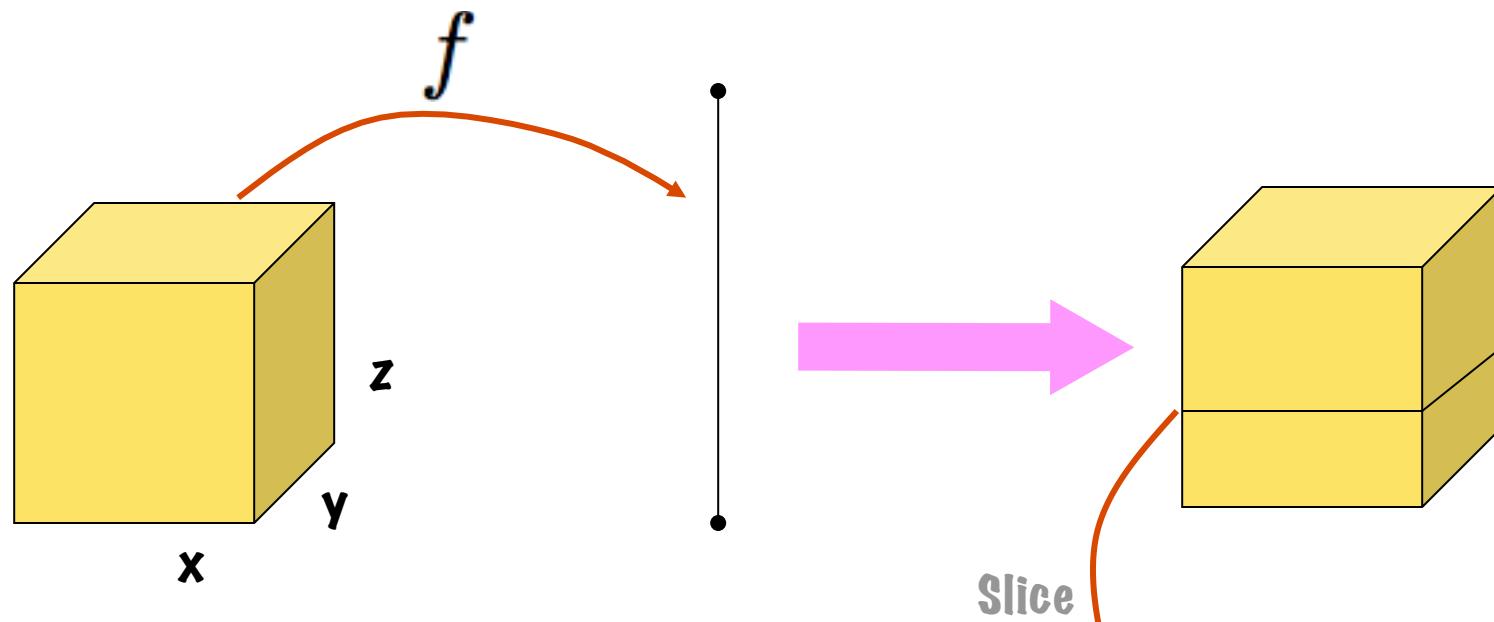
# Examples of “Images” as Functions



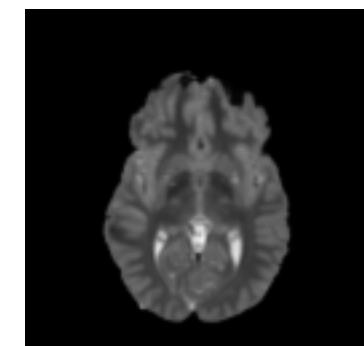
# Images As 2D Functions



# 3D Images – Volumes

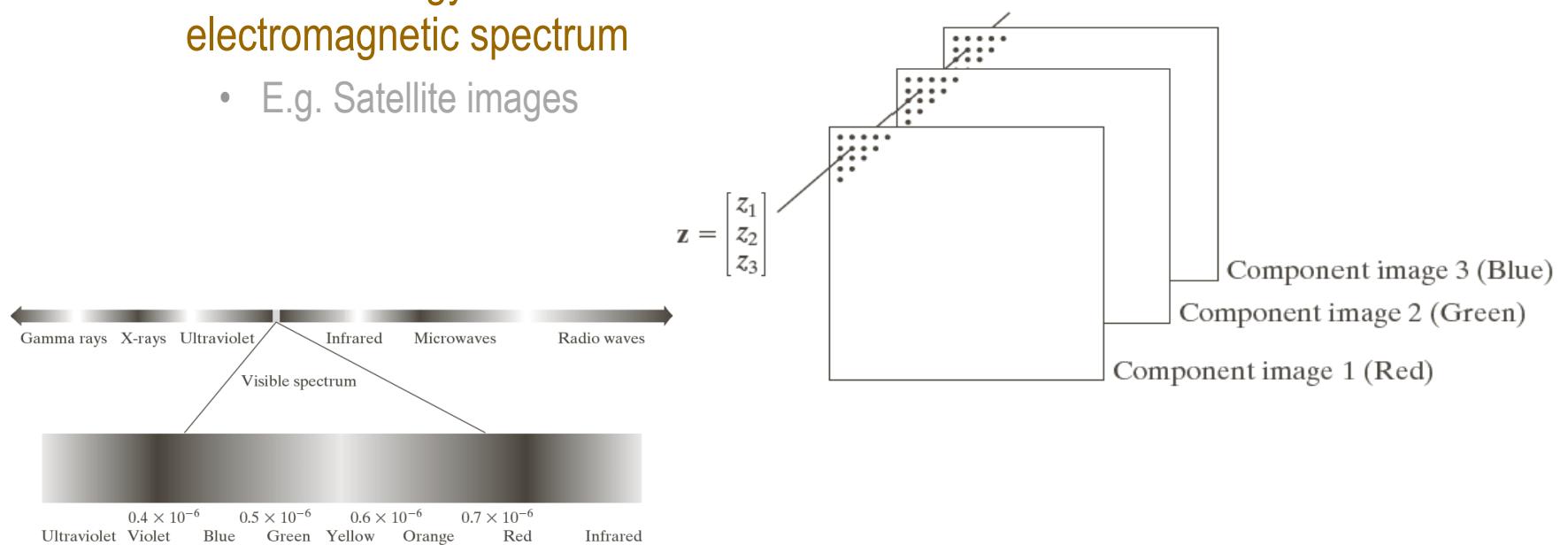


Impossible to graph:  
Slicing  
Volume rendering  
Scientific visualization



# Multivalued Images

- Color images: mappings to some subset of R3
  - Color spaces: RGB, HSV, etc.
- Spectral imagery
  - Measure energy at different bands within the electromagnetic spectrum
    - E.g. Satellite images

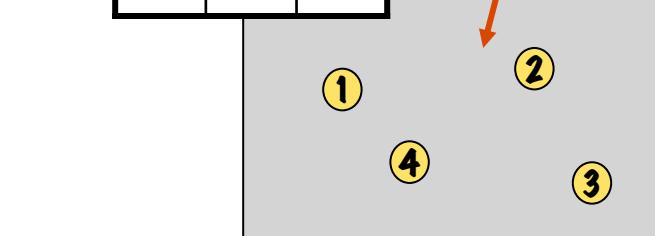


# Image As Grid of Values

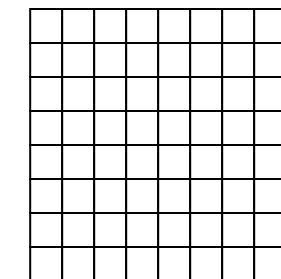
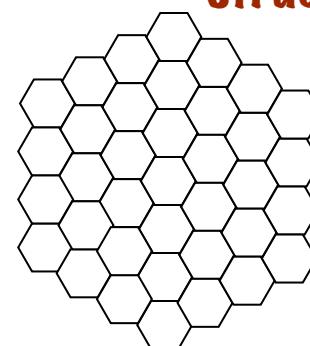
- Two views
  - Domain is a discrete set of samples
  - Samples are points from an underlying continuous function
- How is the grid organized?
  - Unstructured
    - Points specified by position and value
  - Structured grids
    - Position inferred from structure/index
    - 1D, 2D, 3D, ....
    - Sizes w, w x h, w x h x d

Unstructured grid

f1	x1	y1
f2	x2	y2
f3	x3	y3
...	...	...
...	...	...

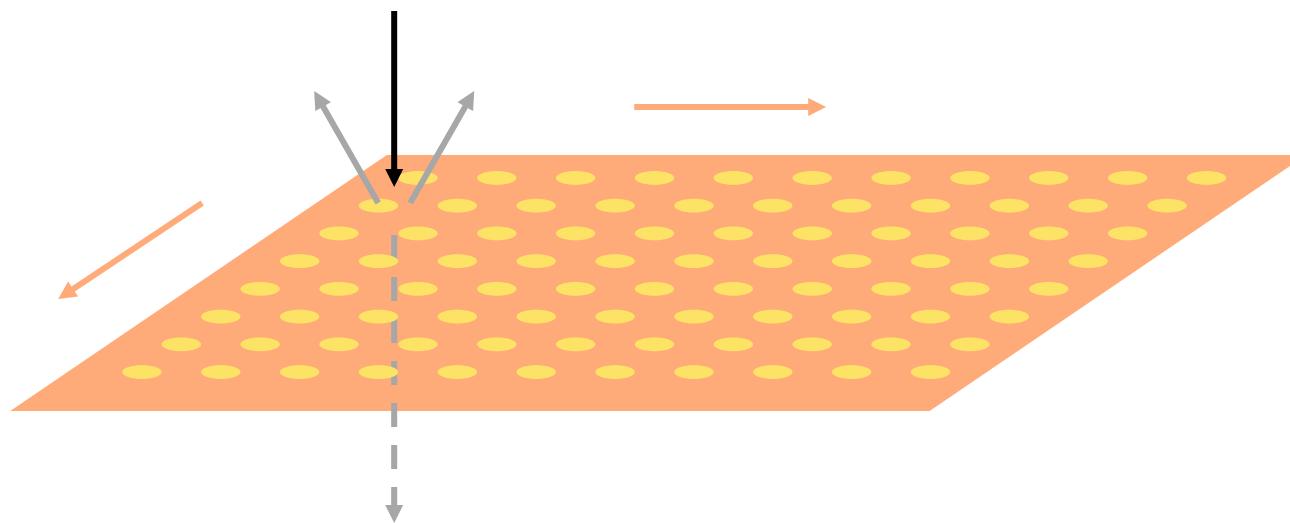


Structured grids

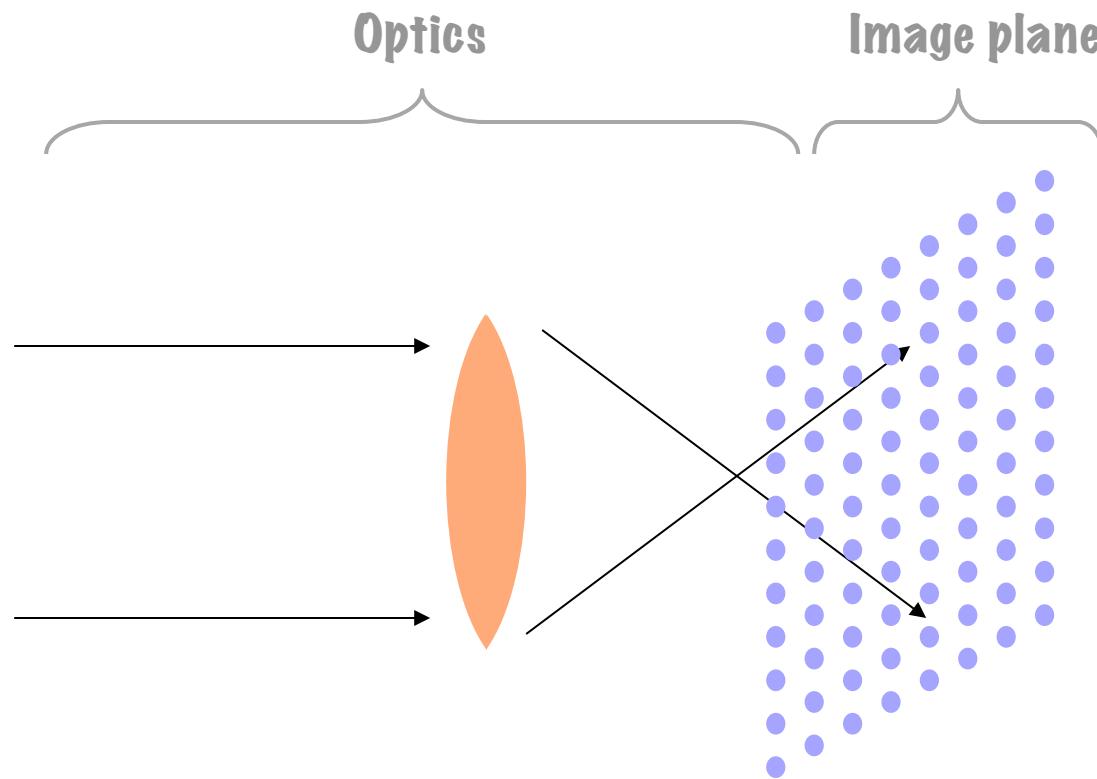


# Where Do Digital Images Come From?

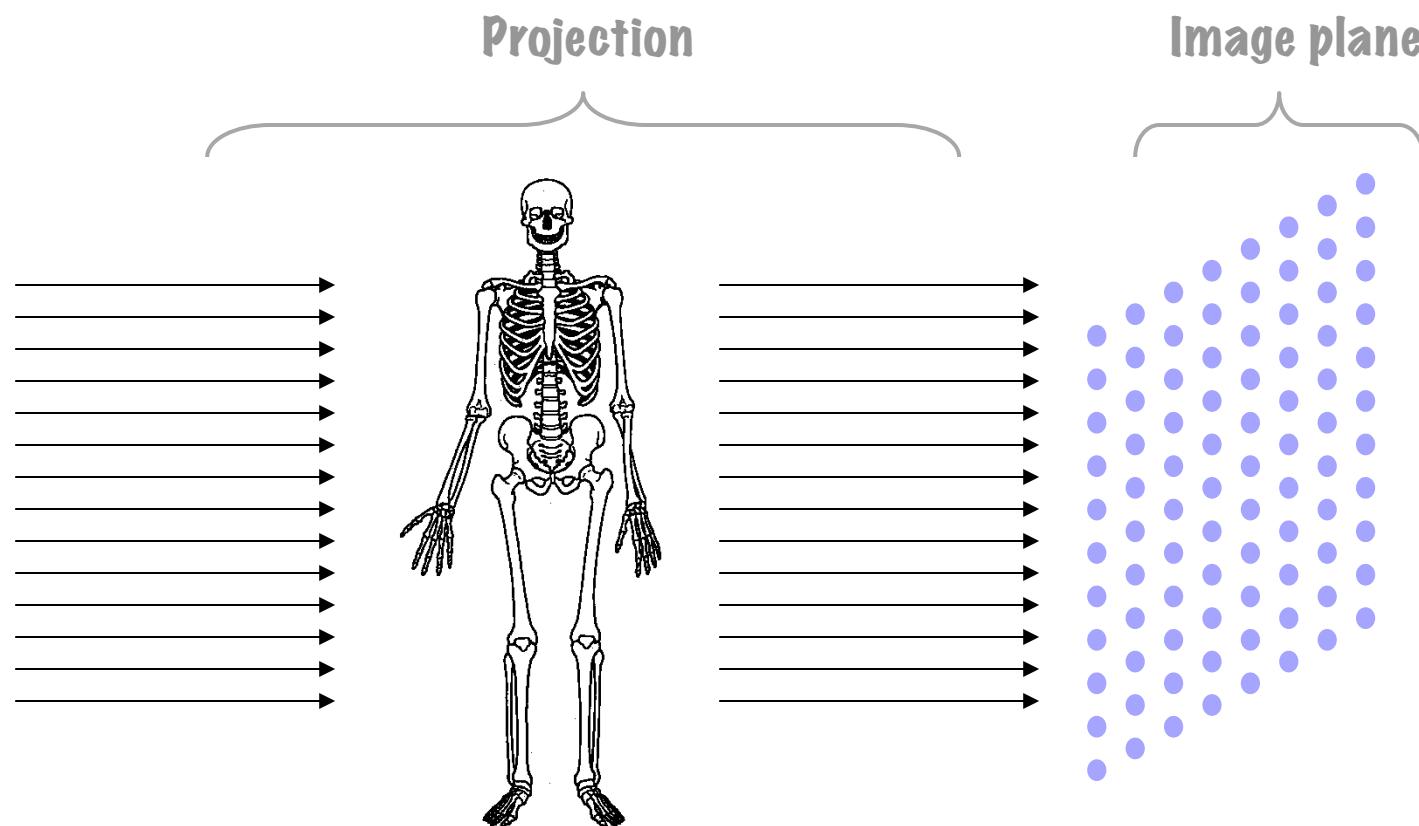
- Digitizing film or paper
  - Rasterize, sample reflectance/transmission on grid



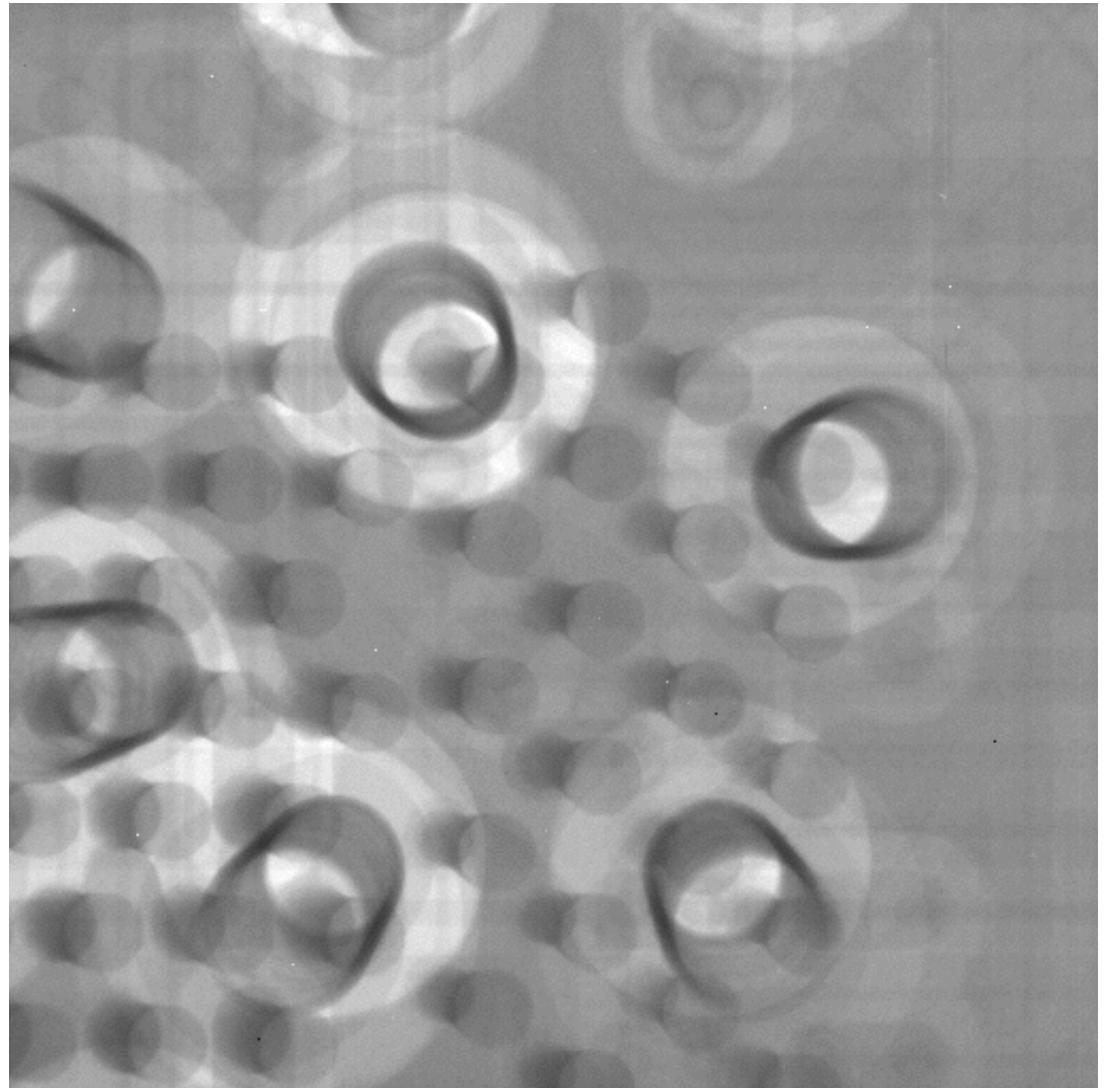
# CCD Cameras



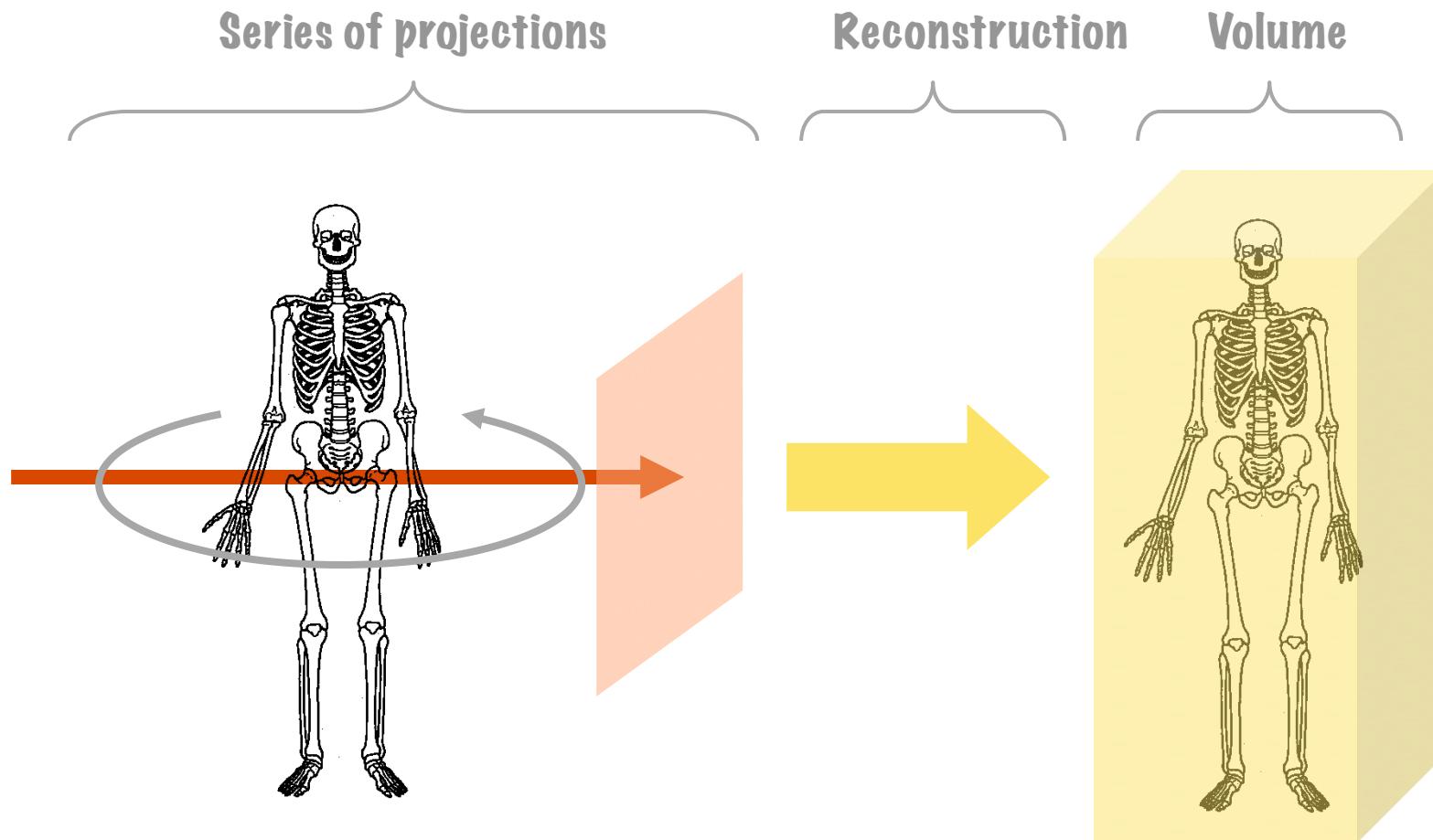
# X-Rays



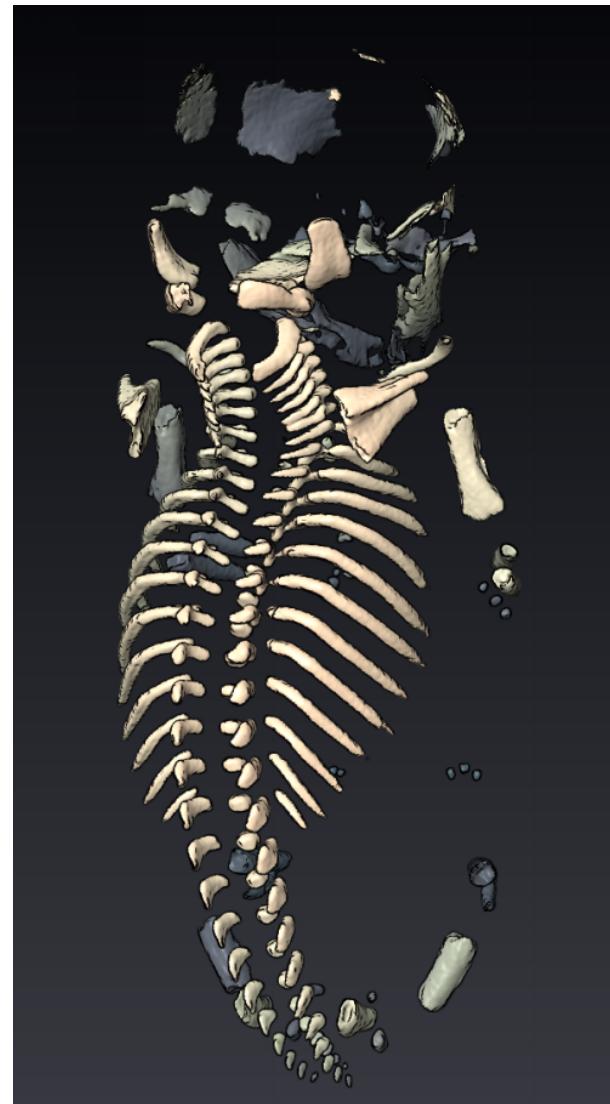
# X-Ray Images



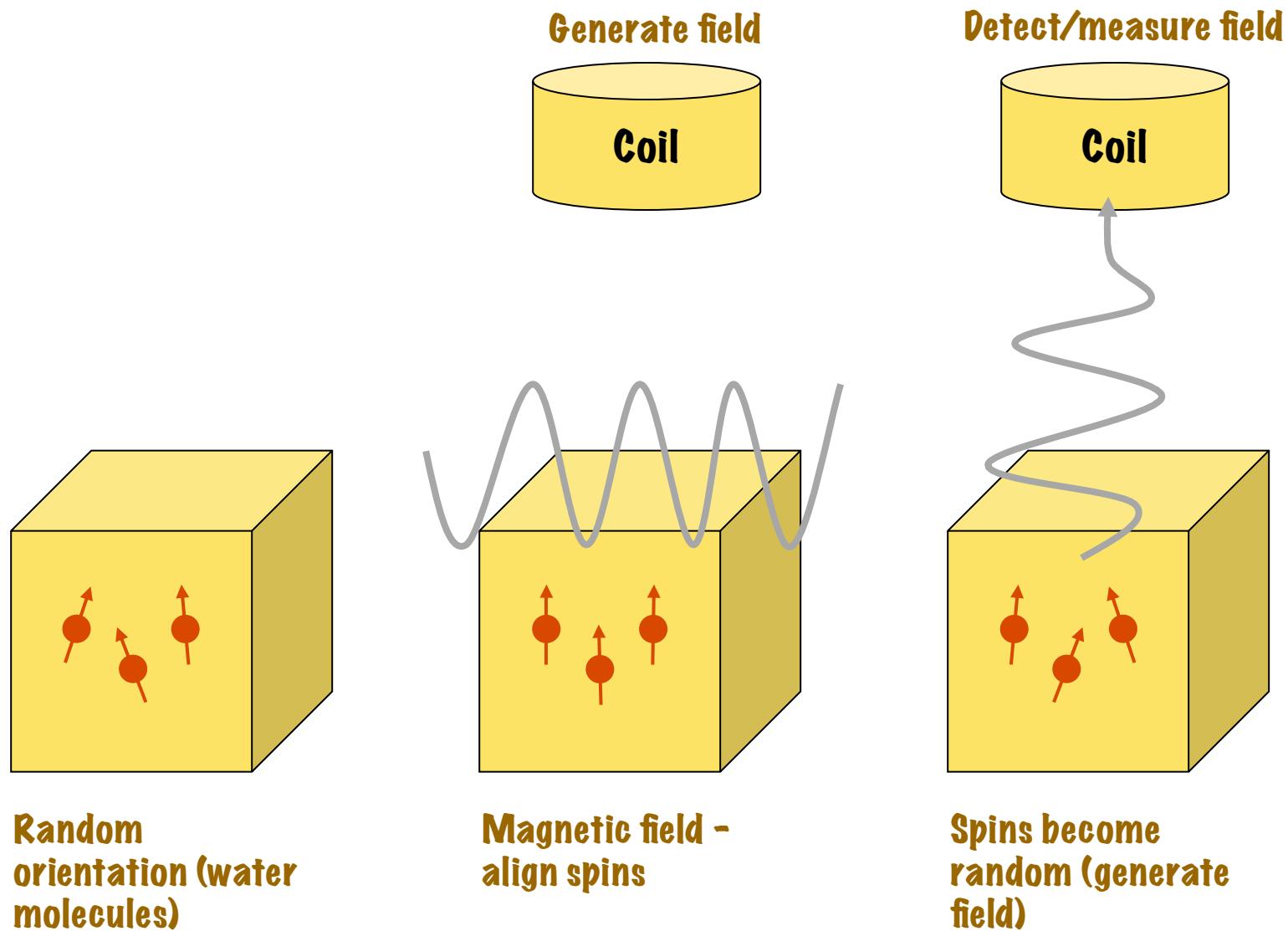
# Computed Tomography



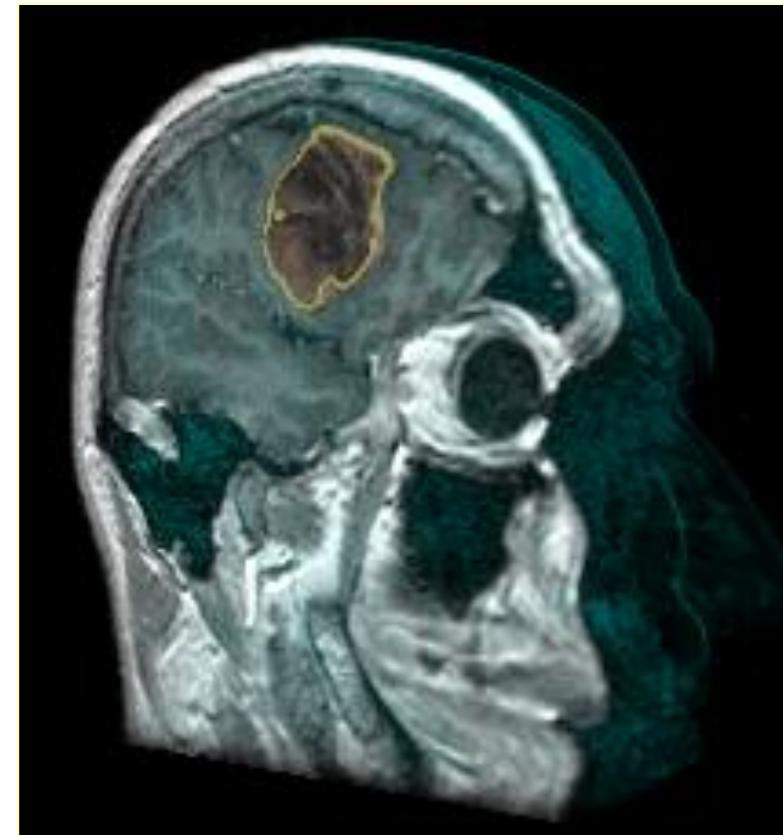
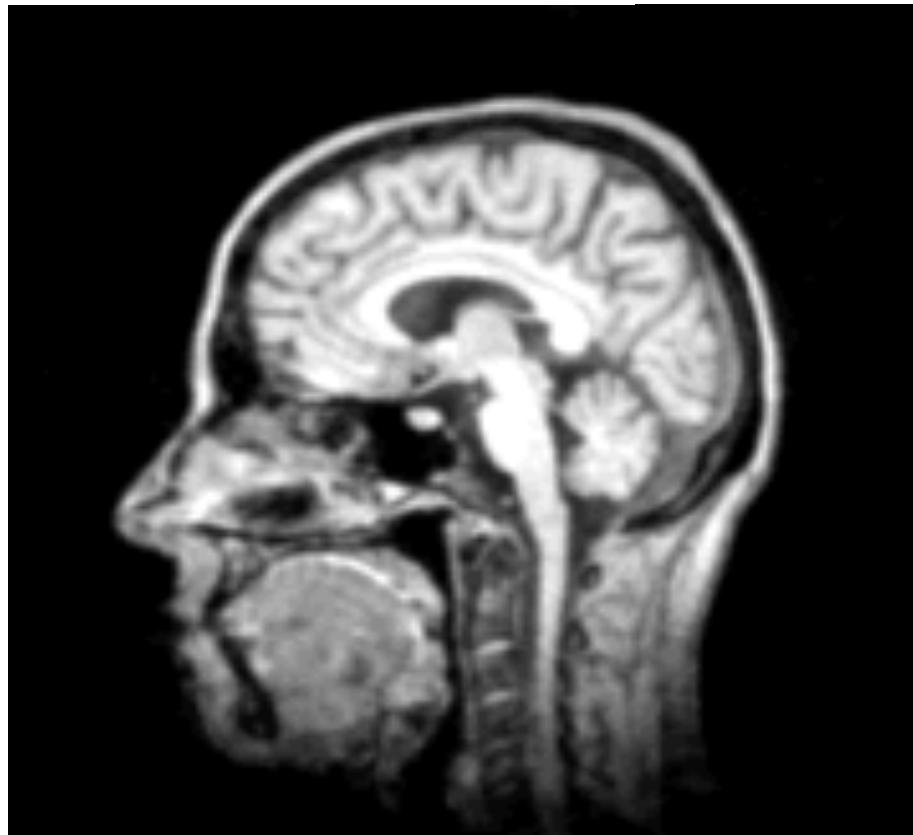
# CT (CAT)



# Magnetic Resonance Imaging

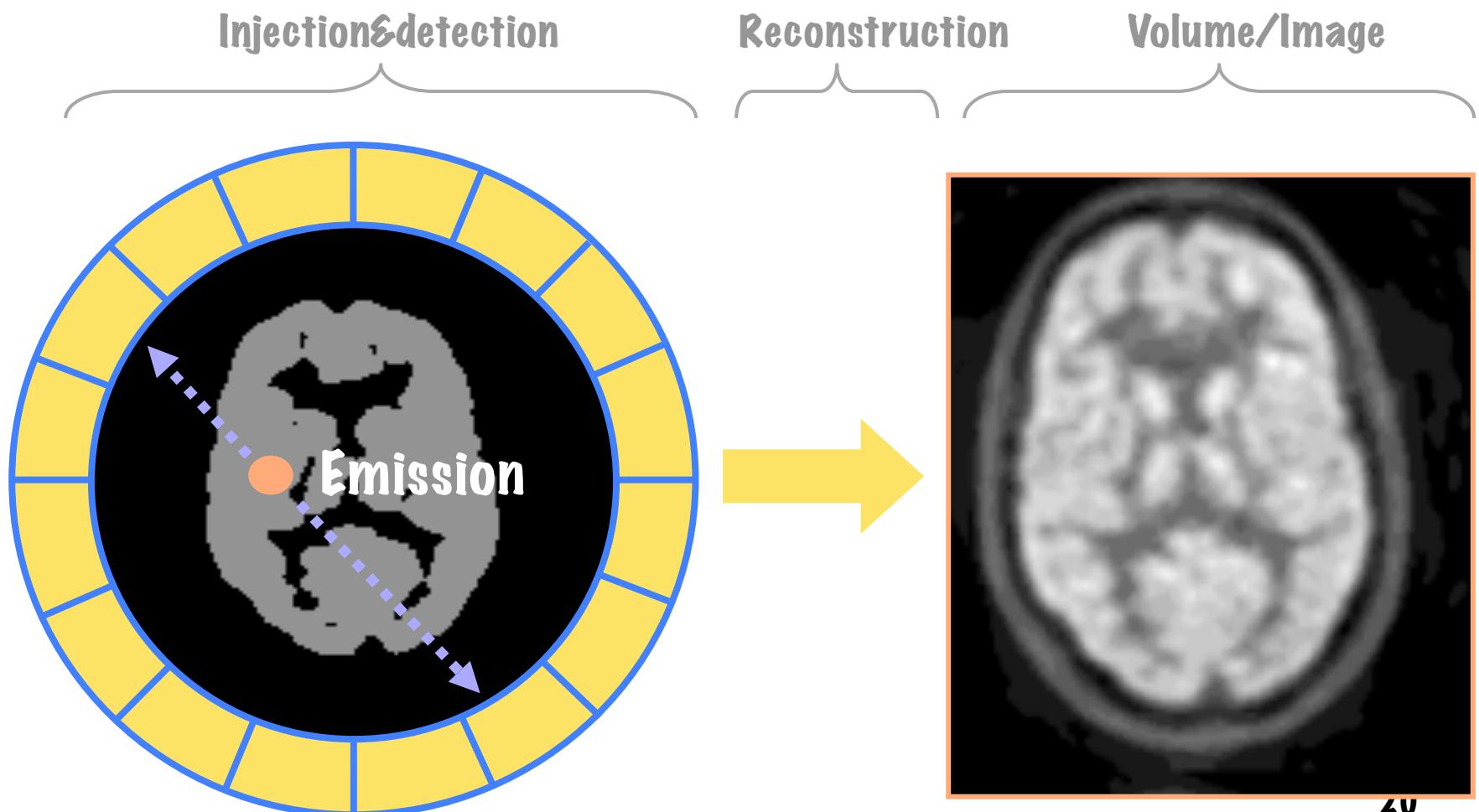


# MRI

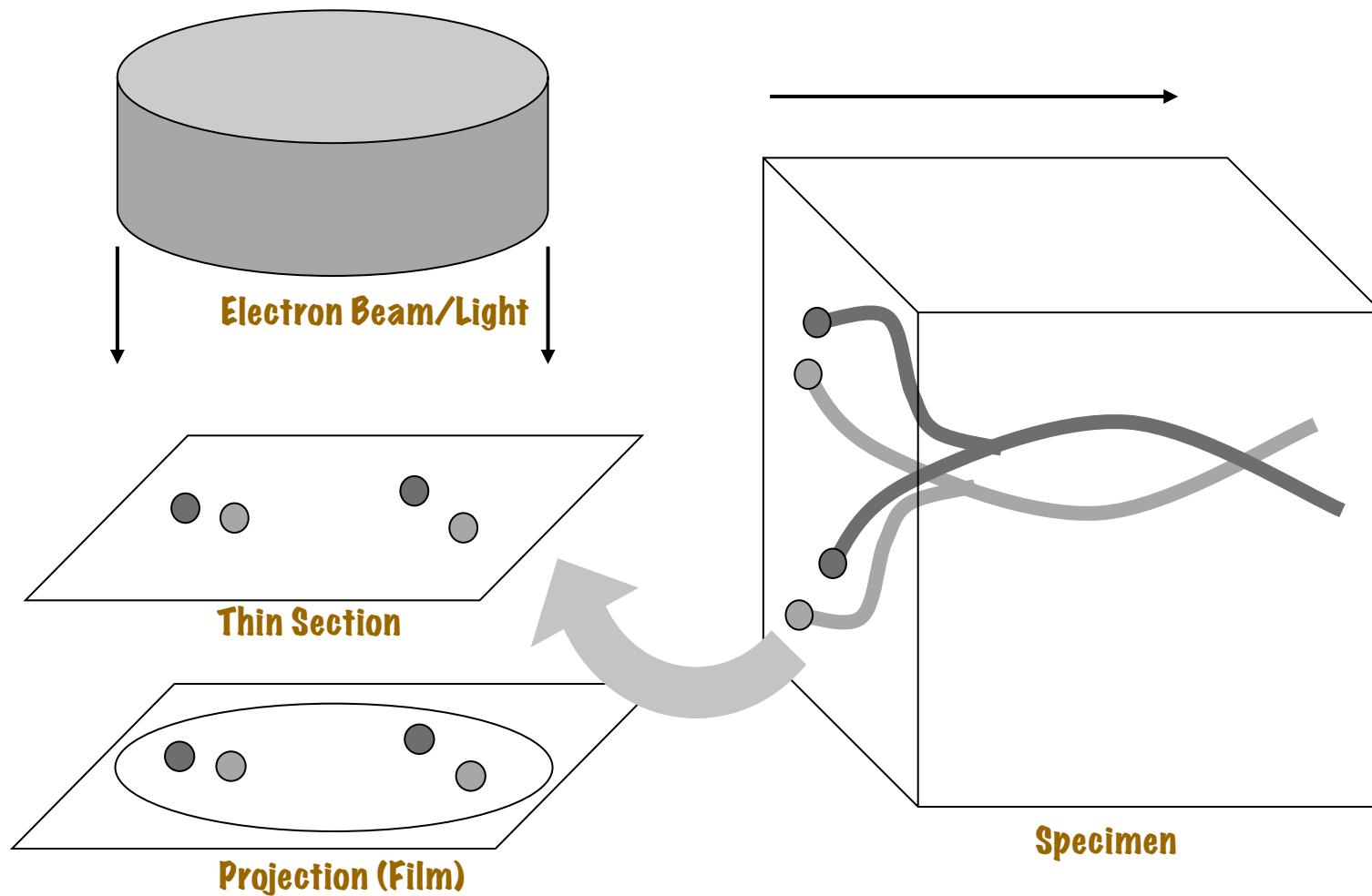


# Nuclear Medicine

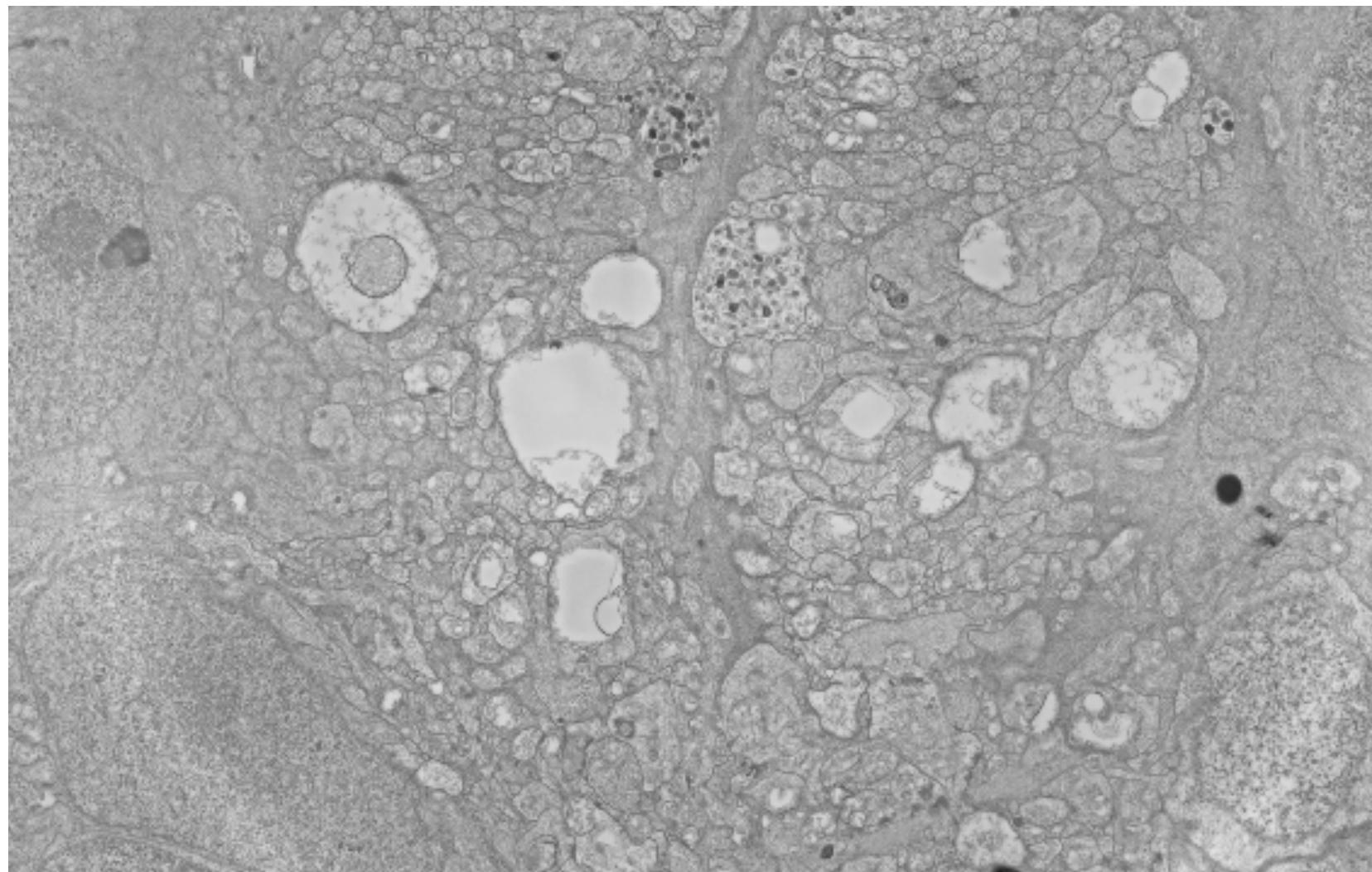
## PET, SPECT, ...



# Serial Sectioning

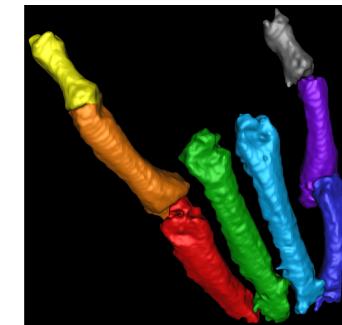
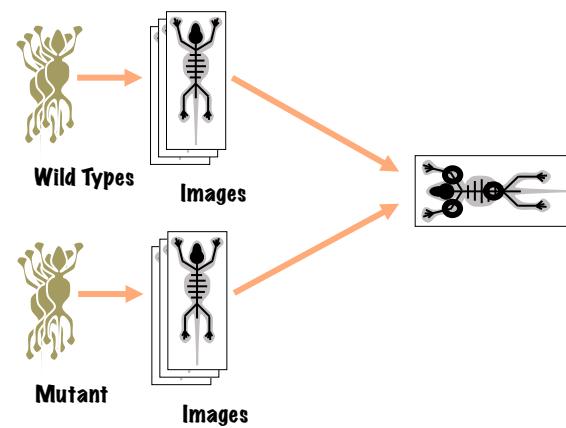
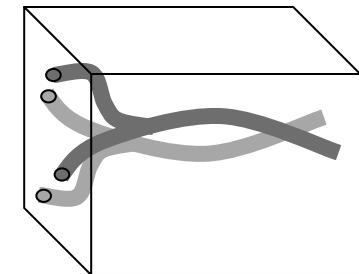
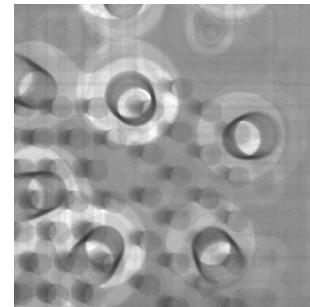


# Serial Section Transmission Electron Microscopy



# Examples

- Quality control of surface-mount packaging
- Retinal architecture from serial section TEM
- Image-based phenotyping



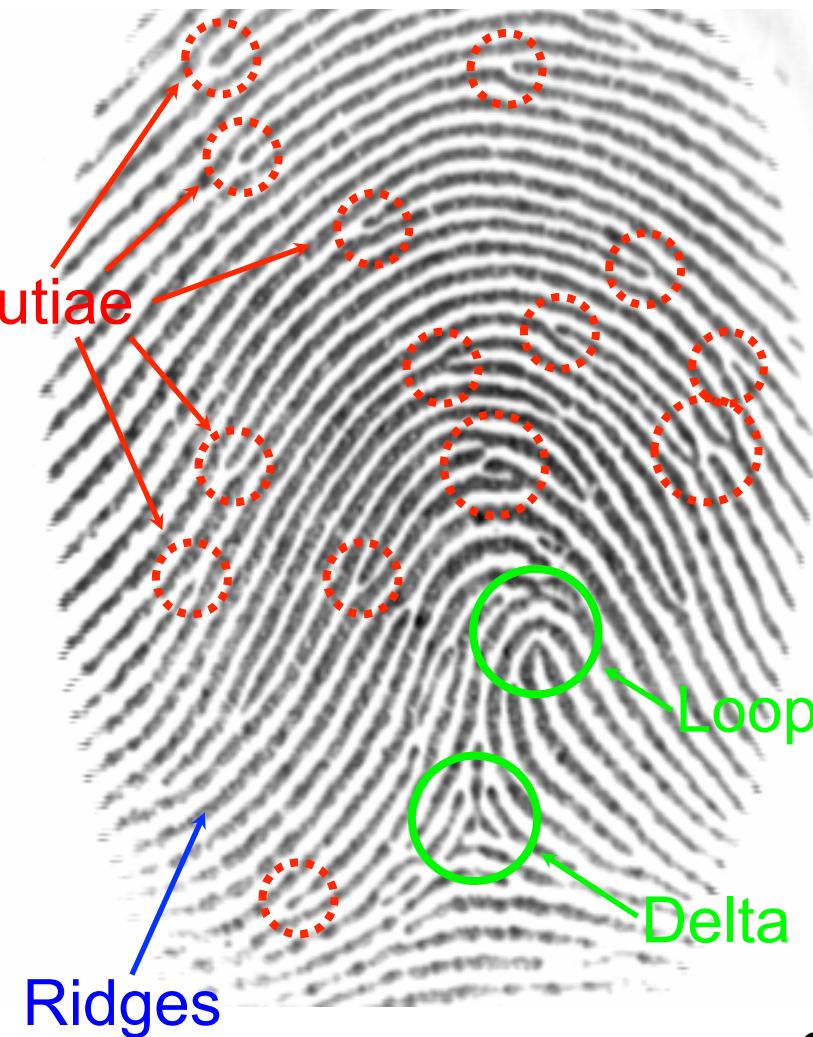
# Fingerprint images

- Ink technique
  - spread ink
  - press on paper
  - capture with CCD camera or scanner
- Latent fingerprints
- Live-scan
  - Optical sensors
  - Capacitive sensor
  - Thermal sensor
  - Piezoelectric (pressure)



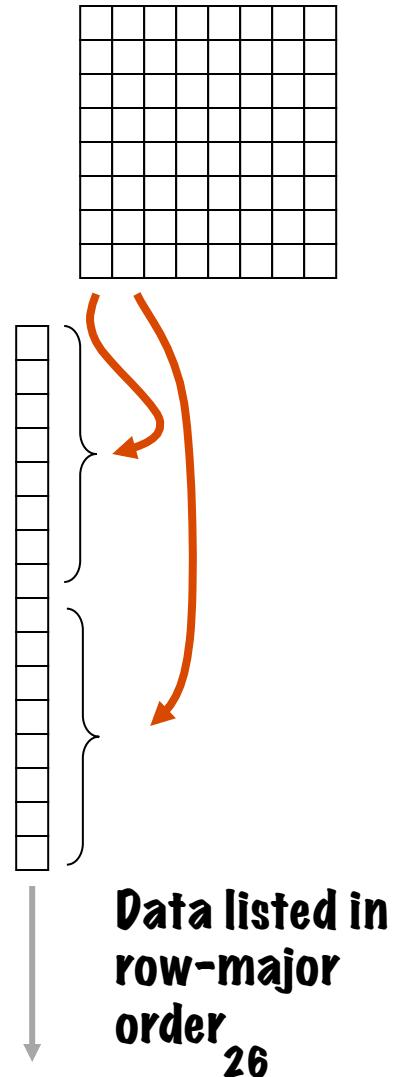
# Fingerprint matching

- Fingerprint patterns are unique to the individual
- Matching
  - using the ridges directly is hard
  - often singularity points are used
    - Local: Minutiae
    - Global: Loop, delta, ...



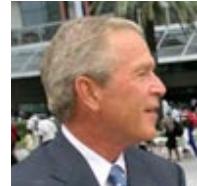
# Data Structures for Images

- Typically “objects”
  - Matlab - Arrays
  - Toolkits define image objects
- Header information
  - Dimension, size, datatype, grid spacing (resolution)
  - “Metadata” (e.g. annotation)
- Data options
  - Nested arrays
  - Unroll structured grid into 1D array
    - Coherent chunk of memory
    - Pointer to memory
  - Often controlled access to memory (e.g. Matlab and speed)



# What Appears on Screen/Monitor?

- A mapping from *abstract* image to computer/graphics hardware
  - Processing for this mapping is one goal of IP
- Typically
  - Values (greyscale or color) map to brightness of screen pixels (proportionally)
  - Domain of image (e.g. grid) maps to screen relesate (proportionally)



How do colors and pixel positions get mapped?

# Image Viewers Displays

- Conventions to make things easy
  - E.g. 24 bit RGB mapped to RGB of screen buffer
    - “Color” images
  - 8 or 16 bit unsigned data is displayed as “greyscale”
    - Generally R=G=B (approximately)
- Other data
  - Negative numbers
  - Floats
  - Larger integers
  - Special numbers such as NaN, etc.

Depends on the viewer. Typically min and max of the data set are set to “black” and “white”, respectively, and the rest are mapped proportionally. The viewer doesn’t always make clear what it is doing. Beware when looking at nonstandard data.

# Image Viewers

- Matlab - built in display
  - “imview(arg1)”
- Photoshop
- Native on OS
  - Preview (Mac OS)
  - “Windows Picture and Fax Viewer”
- Web browsers
- Open source
  - Gimp
  - XV

# Arithmetc Operations on Images

- Arithmetic operations on pixel values
  - Multiple images with the same domain
  - Image become arguments
    - Implied that the operation is applied pointwise across the domain
  - Addition, subtraction, multiply, divide, boolean

$$h = f + g \Rightarrow h(i, j) = f(i, j) + g(i, j)$$
$$\forall (i, j) \in \mathcal{D}$$

# Array vs. Matrix Operations

$$\underbrace{\begin{pmatrix} a & b \\ c & d \end{pmatrix}}_A \times \underbrace{\begin{pmatrix} x & y \\ w & z \end{pmatrix}}_X = \begin{pmatrix} ax + bw & ay + bz \\ cx + dw & cy + dz \end{pmatrix}$$

Matrix multiply  
( MATLAB `A*X` )

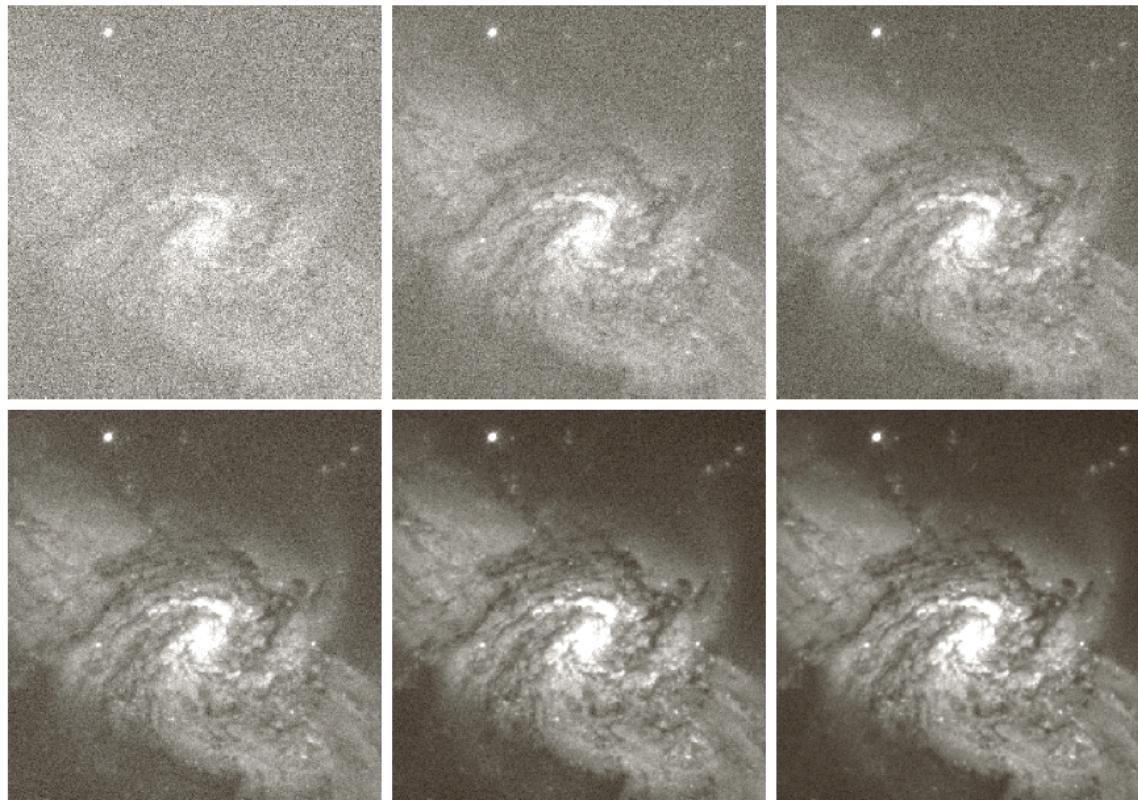
$$\underbrace{\begin{pmatrix} a & b \\ c & d \end{pmatrix}}_A \times \underbrace{\begin{pmatrix} x & y \\ w & z \end{pmatrix}}_X = \begin{pmatrix} ax & by \\ cw & dz \end{pmatrix}$$

Array multiply  
( MATLAB `A.*X` )

Images can be represented as matrices, but the operations refer to array operations unless otherwise specified

# Arithmetic operations: $f + g$

Averaging (adding) multiple images can reduce noise

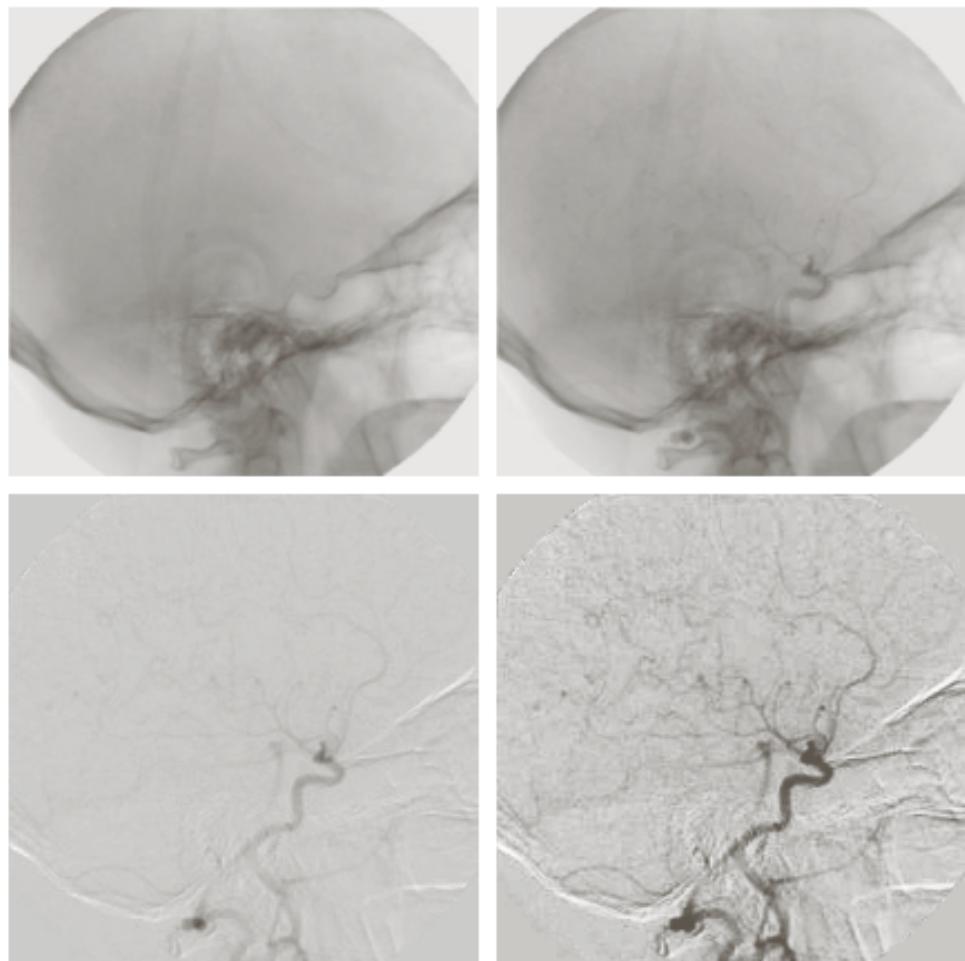


a b c  
d e f

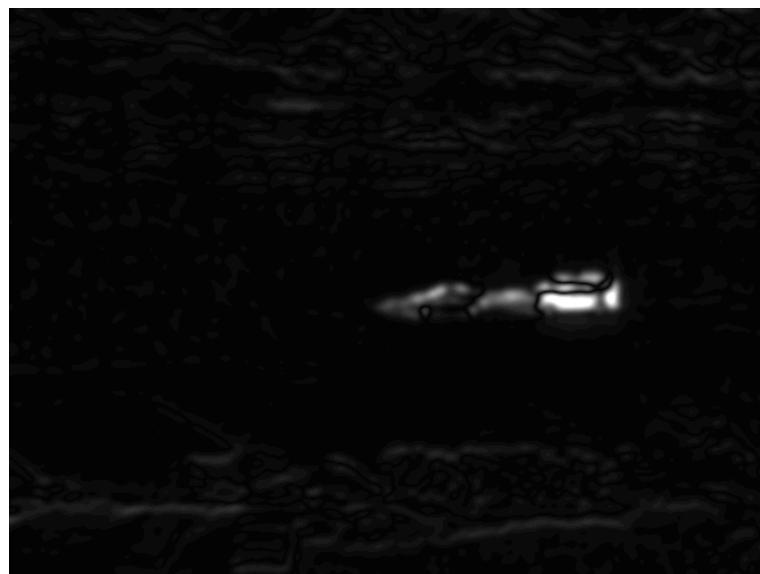
**FIGURE 2.26** (a) Image of Galaxy Pair NGC 3314 corrupted by additive Gaussian noise. (b)–(f) Results of averaging 5, 10, 20, 50, and 100 noisy images, respectively. (Original image courtesy of NASA.)

# Arithmetic operations: $f - g$

Digital Subtractive Angiography  
(DSA)



# Image Subtraction: Motion Detection



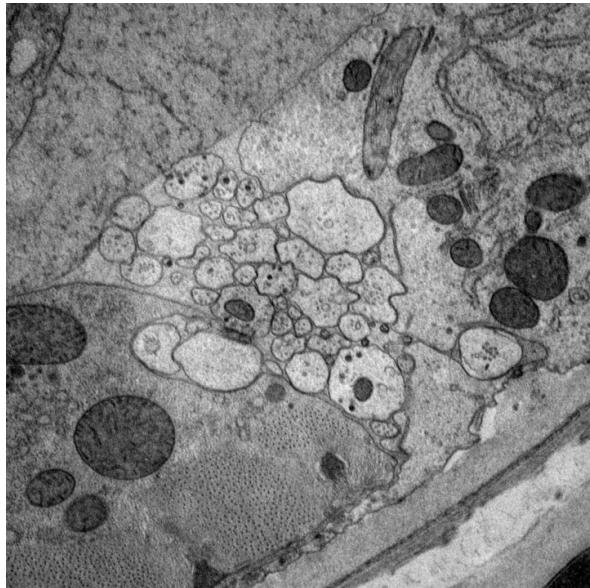
37

# Arithmetic operations: $f \times g$

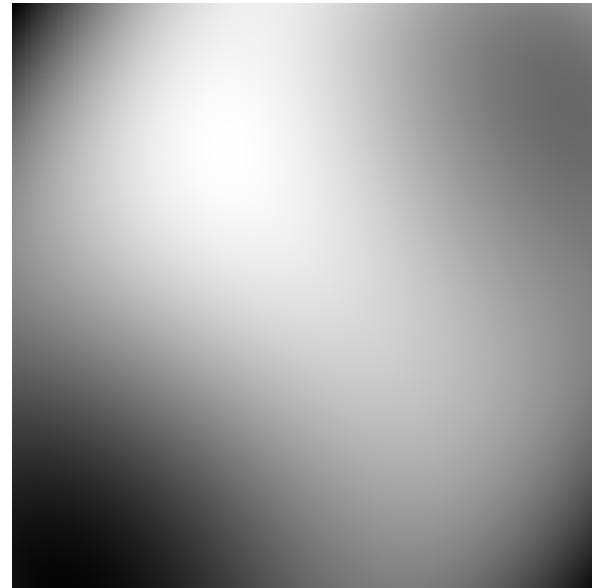


**FIGURE 2.30** (a) Digital dental X-ray image. (b) ROI mask for isolating teeth with fillings (white corresponds to 1 and black corresponds to 0). (c) Product of (a) and (b).

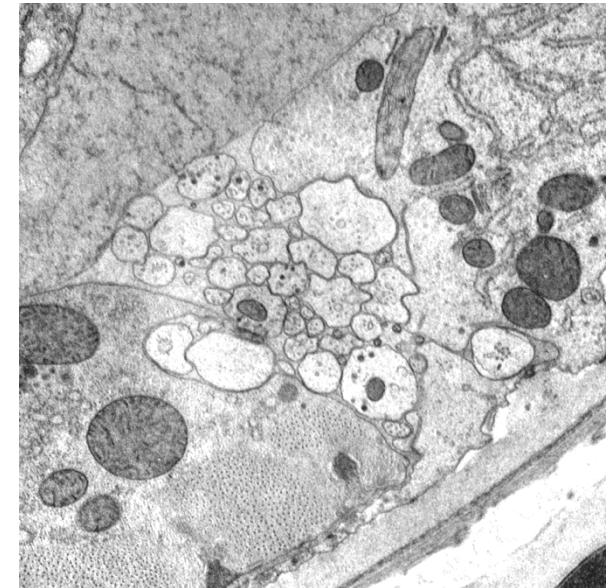
# Arithmetic operations: $f / g$



Captured image



Illumination



Corrected image

# Operations on Cartesian Image Grids

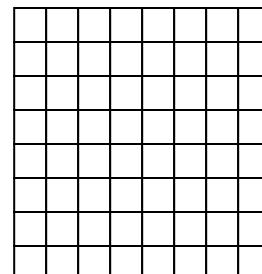
- Grid resolution
- Neighborhoods
- Adjacency and connectivity
- Paths
- Connected components
- Flood fill

# Image Coordinates and Resolution

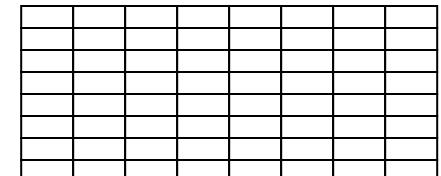
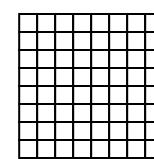
- A single point on an image grid is a “pixel”
  - Sometimes this is just the location, sometimes also the value
- References to pixels
  - Single index (implied ordering) “i” or “f(i)”
  - Multiple index (gives position on logical grid) “i,j” or “f(i,j)”
- Physical coordinates
  - Logical coordinates place the pixel in physical space
  - r - resolution (e.g. mm's)
  - o - origin

$$(x_{ij}, y_{ij}) = (r_x i + o_x, r_y j + o_y)$$

***Resolution vs size vs dimension***

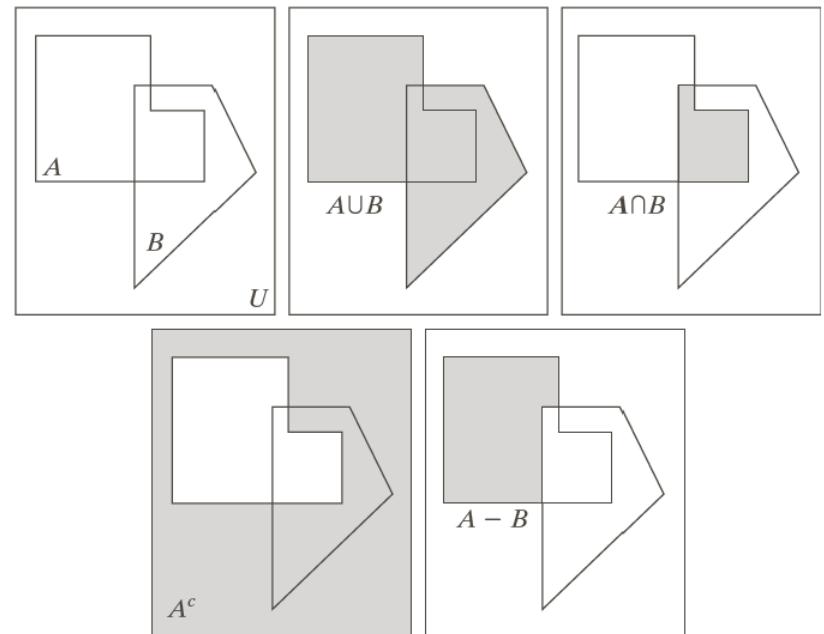


**Different physical realizations of the same logical grid**



# Index Sets

- An index set is a collection of pixel locations
  - Used to specify subsets of an image
  - All boolean set operations apply
- Convention
  - Represent the set as an image with 0 indicating non membership and >0 indicating membership
  - Logical operations become arithmetic operations



# Neighborhood

- *Neighborhood ( $N$ ):* a set of *relative* indices that satisfy the symmetry condition

- Symmetry:

$$(i, j) \in \mathcal{N} \Leftrightarrow (-i, -j) \in \mathcal{N}$$

- Applying neighborhoods:

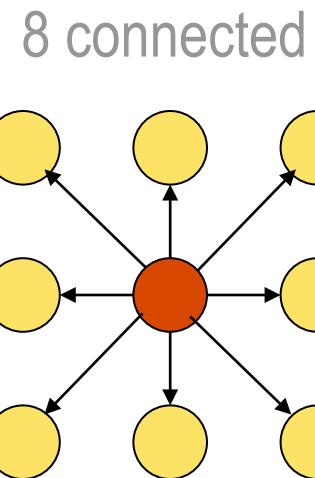
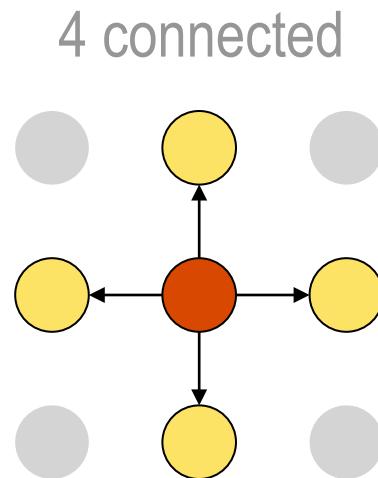
- $\mathcal{N}(i, j) = \{(k, l) | (k - i, l - j) \in \mathcal{N}\}$
  - I.e. you translate neighborhoods to different locations

- Notice:

$$(p, q) \in \mathcal{N}(i, j) \Leftrightarrow (i, j) \in \mathcal{N}(p, q)$$

# Adjacency

- Impose topological structure on the grid
- Local relationships between pixels
- Help to establish distances, paths, connectedness, etc.
- Typically adjacency is local and symmetric
- For 2D images we consider:



Denote  
 $I \sim J$

# Paths

- *Path*: Ordered set of indices such that consecutive indices are adjacent

$$\mathcal{P} = (I_1, I_2, \dots, I_n) \text{ such that } I_i \sim I_{i+1} \\ \forall i = 1, \dots, n - 1$$

- *Noncyclic path* – unique indices
- *Closed path* – noncyclic and first and last adjacent

# Distances in Images

- Grid distance vs physical distance

- Physical distance between pixels I and J

$$D(I, J) = \sqrt{(x_I - x_J)^2 + (y_I - y_J)^2}$$

- Grid distance: options

- Grid Euclidean

- Manhattan (city block)

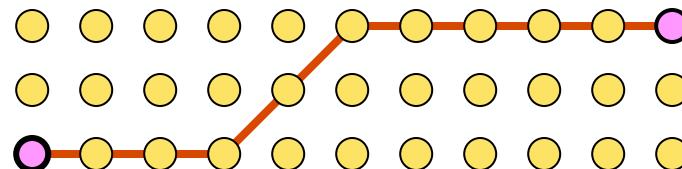
$$D((i, k), (j, l)) = \sqrt{(i - j)^2 + (k - l)^2}$$

- Shortest path

$$D((i, k), (j, l)) = |i - j| + |k - l|$$

- Assign cost to each transition between adjacent pixels

- Find path with shortest cost

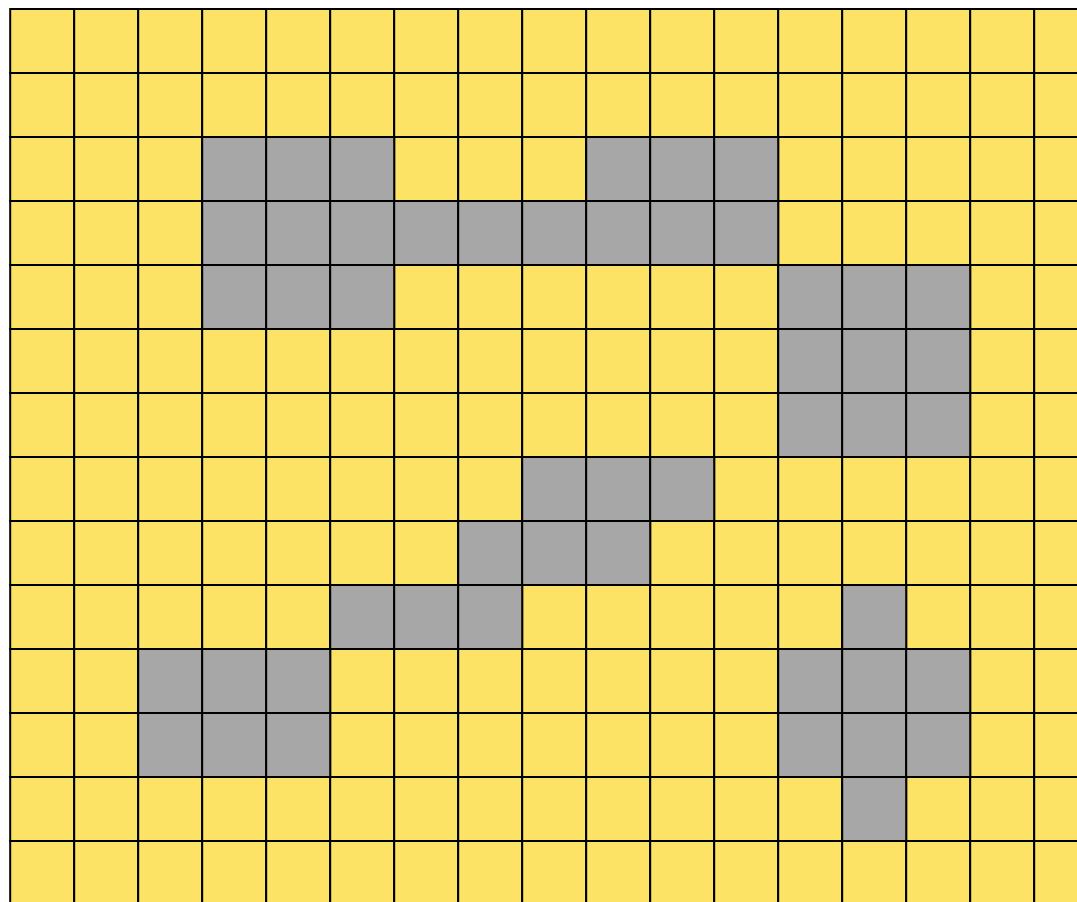


# Connected Component

- Consider image with a binary property
  - I.e. test on each index  $B(I)$  returns either true or false
- *Correct path* : path for which every pixel satisfies  $B(I)$
- *Connected component (C)* : set of pixels such that for every pair of pixels in  $C$  there exists a correct path between them

# Connected Component

- How many distinct CC's are there?



# A Simple Algorithm: Flood Fill

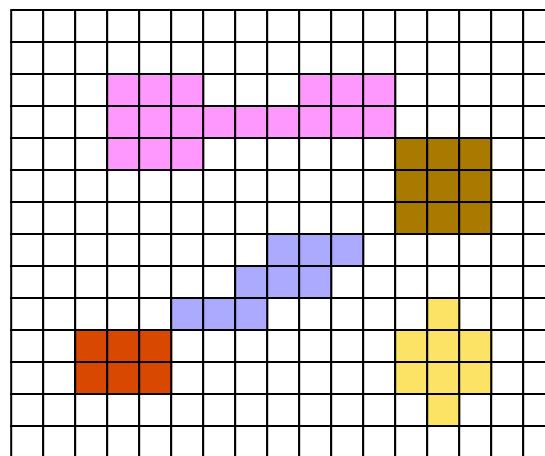
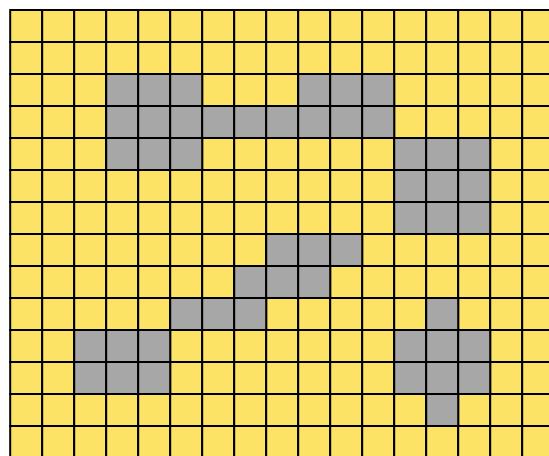
- Highlight regions in an image
- “ $\text{Test}(i, j)$ ” - is value at pixel  $(i, j)$  between  $a$  and  $b$
- Inputs: seed, image, test function
- Data structures: input array, output array, list of grid points to be processed

# A Simple Algorithm: Flood Fill

- Empty list, clear output buffer (=0)
- Start at seed  $(i,j)$  and if  $\text{Test}(i,j)$ , put  $(i,j)$  on list and mark  $\text{out}[i,j]=1$
- Repeat until list of points is empty:
  - Remove point  $(i,j)$  from list
  - (Loop) for all 4 neighbors  $(i',j')$  of  $(i,j)$ 
    - If  $(\text{Test}(i',j') \text{ and } \text{out}[i',j']=0)$  put  $(i',j')$  on list and mark  $\text{out}[i',j']=1$
- Properties
  - Guaranteed to stop
  - Worst case run time

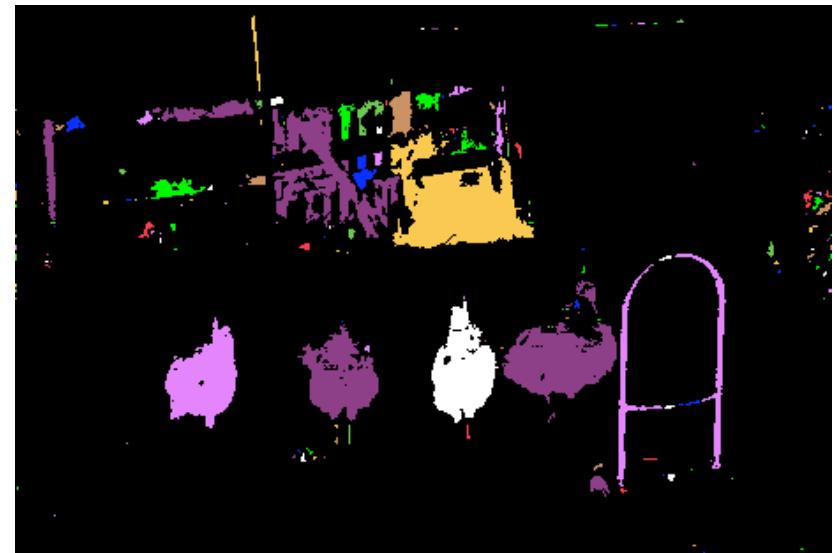
# Connected Component Analysis

- Input: image and test
- Output: an integer image (label map) that has either “0” (failed test) or a positive integer associated with each distinct connected component



# CC – Purpose

- When objects are distinguishable by a simple test (e.g. intensity threshold)
- Delineate distinct objects for subsequent processing
  - E.g Count the number, sizes, etc.
  - Statistics, find outliers irregular objects



# CC Output/Extensions

- Output is typically a “label map”
- How to handle foreground/background, etc
- How to display