Biomedical Informatics 260

Coordinate Systems and Image Registration

Lecture 7

David Paik, PhD

Spring 2019

Today: Coordinate Systems and Image Registration

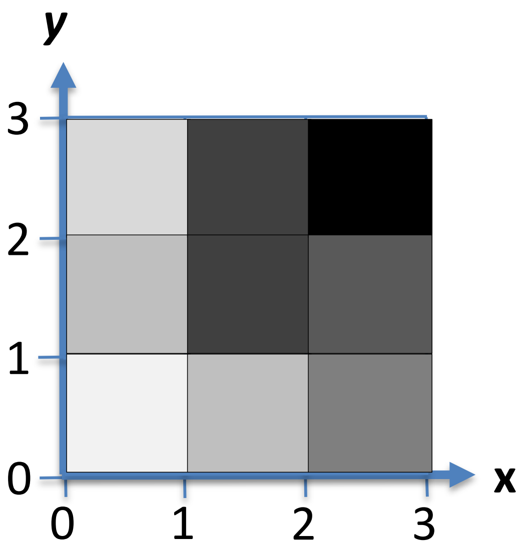
- You have two images of the same patient (or even different patients) that you want to align with each other
  - Multimodality imaging
  - Aligning cohorts of research subjects
- Topics:
  - Coordinate Systems
  - 3D Rotations
  - Image Registration Algorithm

Coordinate Systems and Transformations

Coordinate Systems

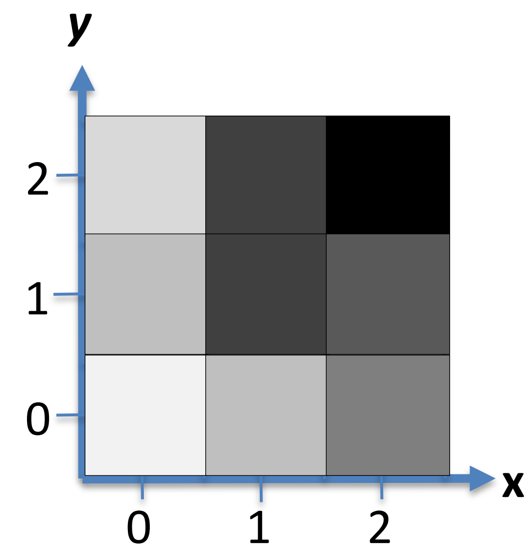
- What coordinate systems are most useful in imaging?
- What coordinate systems enable efficient transformations of large numbers of vertices?
- How do we perform rotations in 3D?

2D Cartesian Coordinates



Geometry

x,y∈[0,*N*]

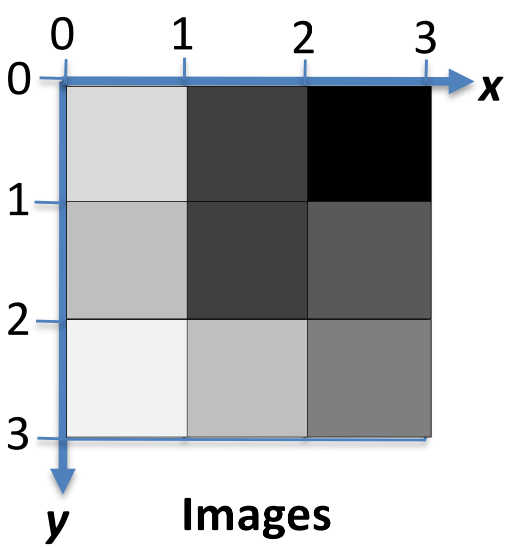


Pixel Centered

x,y∈[-0.5,*N*-0.5]

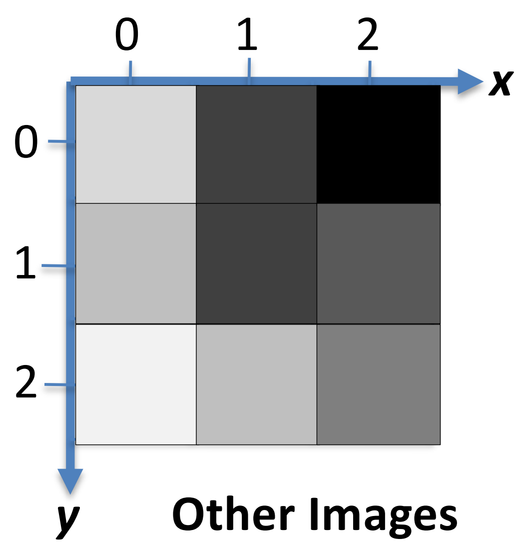
Some pragmatic considerations:

- ½ pixel offset
- upper vs lower left origin
- row,col vs *x,y*
- starting at 0 vs. 1
- pixel dimensions



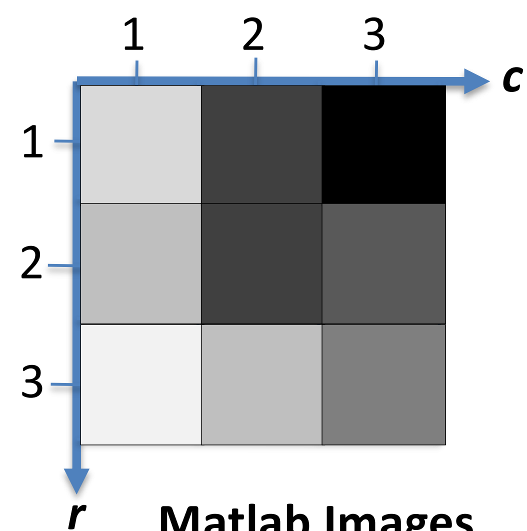
Images

x,y∈[0,*N*]



Other Images

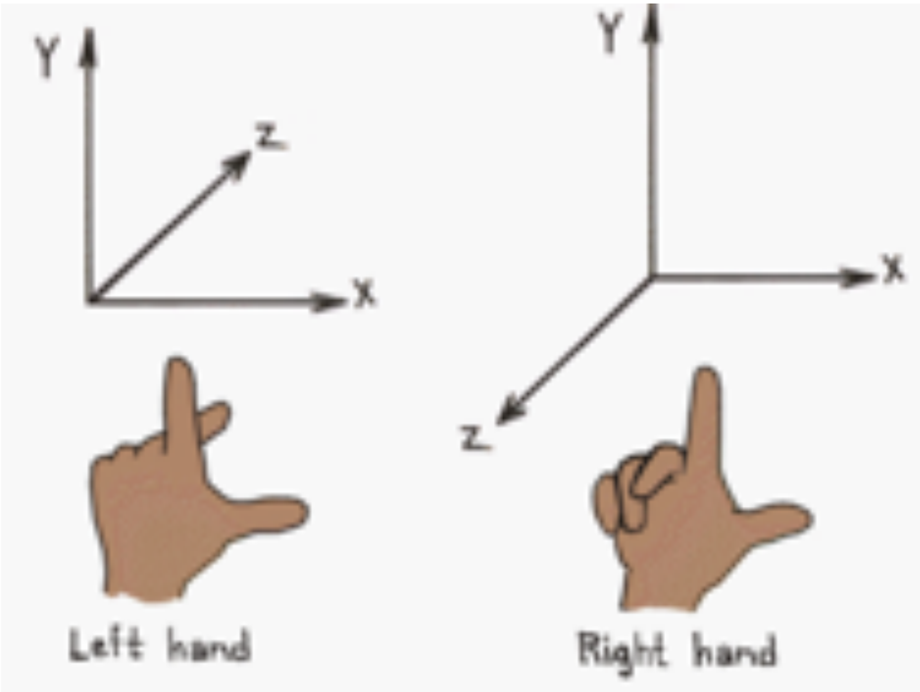
x,y∈[-0.5,*N*-0.5]



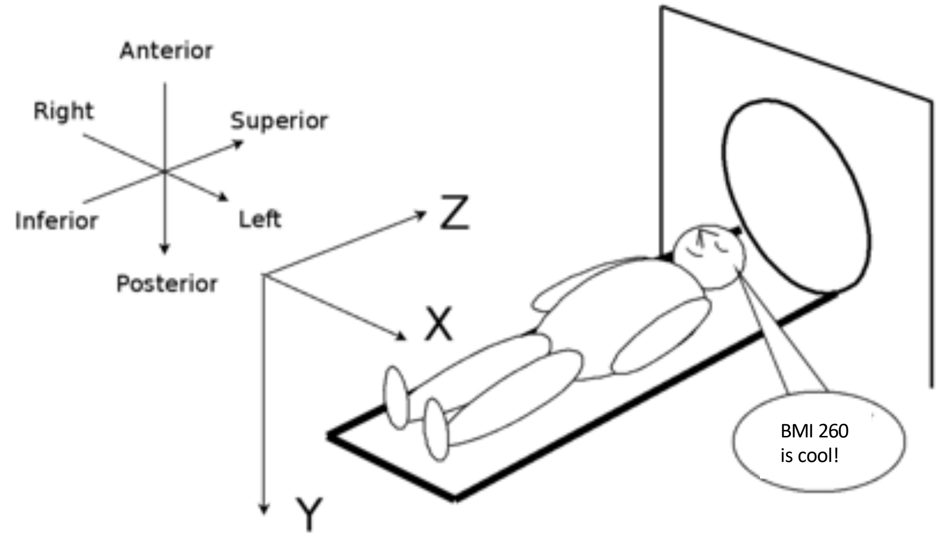
Matlab Images

r,c∈[0.5,*N*+0.5]

3D Cartesian Coordinates



G Otto, Penn State



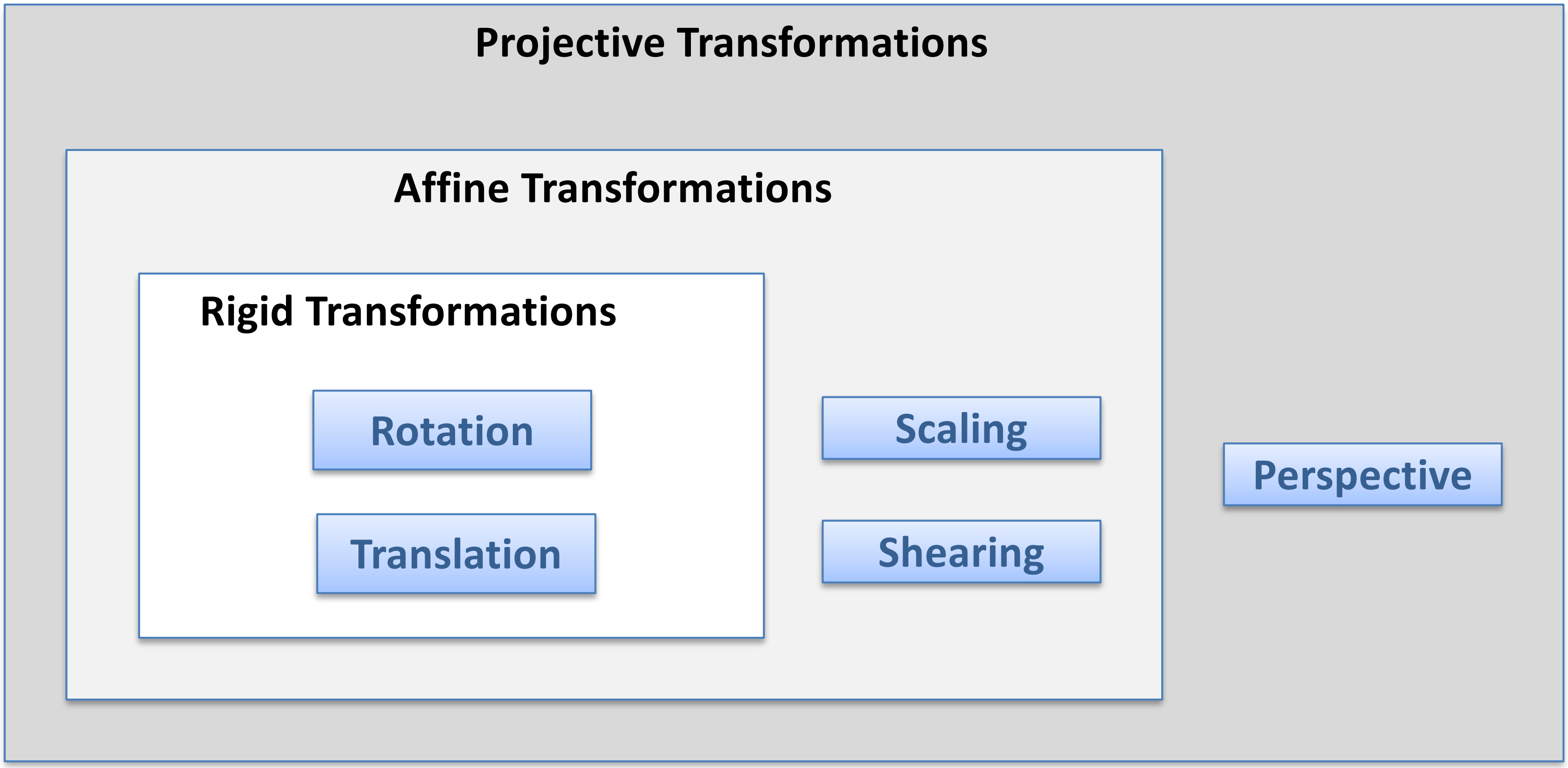
vtk.org

Generally, right handed coordinate systems are preferred

DICOM is *L-P-S* (positive *X-Y-Z* = Left-Posterior-Superior)
Some formats are *R-A-S* (positive *X-Y-Z* = Right-Anterior-Superior)

What is the danger of not being careful about coordinate systems?

Types of Coordinate System Transformations



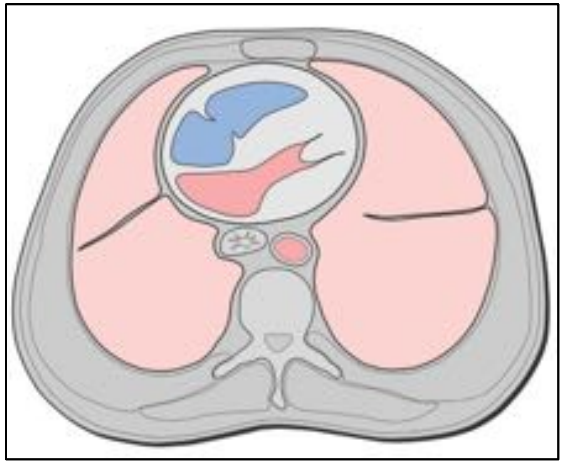
Rigid: Preserves angles, distances and handedness

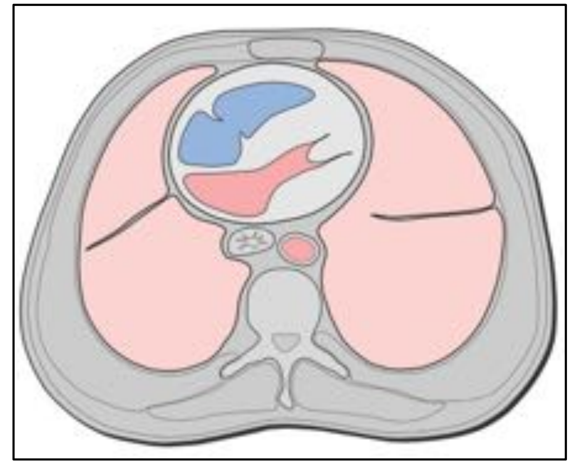
Affine: Preserves co-linearity, parallel lines

Projective: Preserves incidence, cross ratios

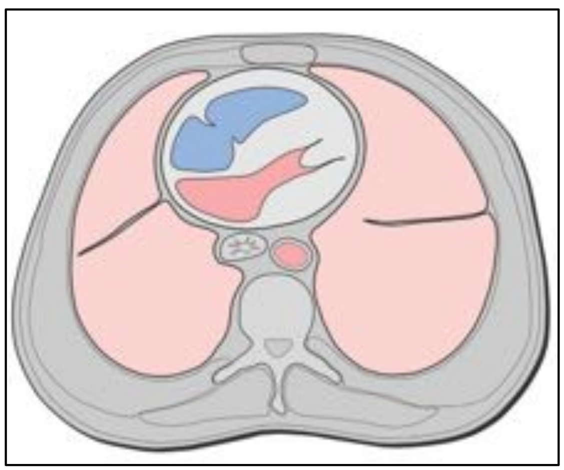
Coordinate Transformations

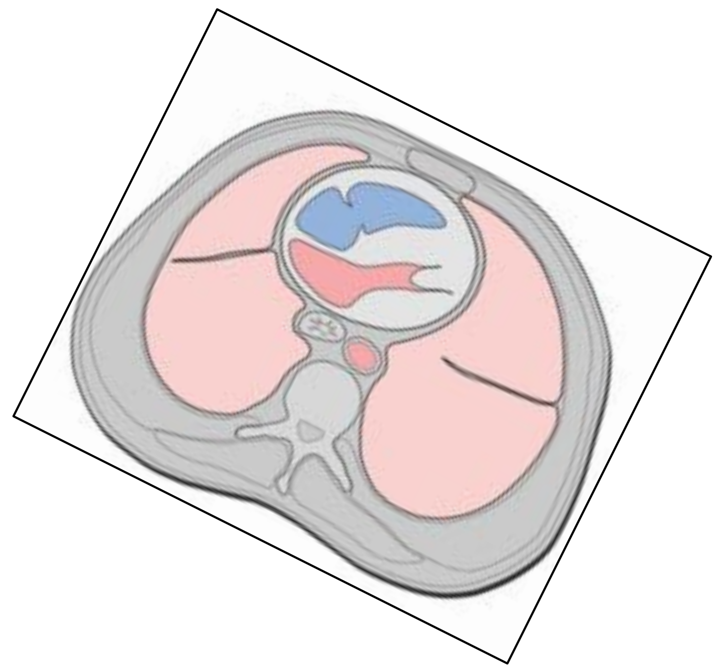
Translation





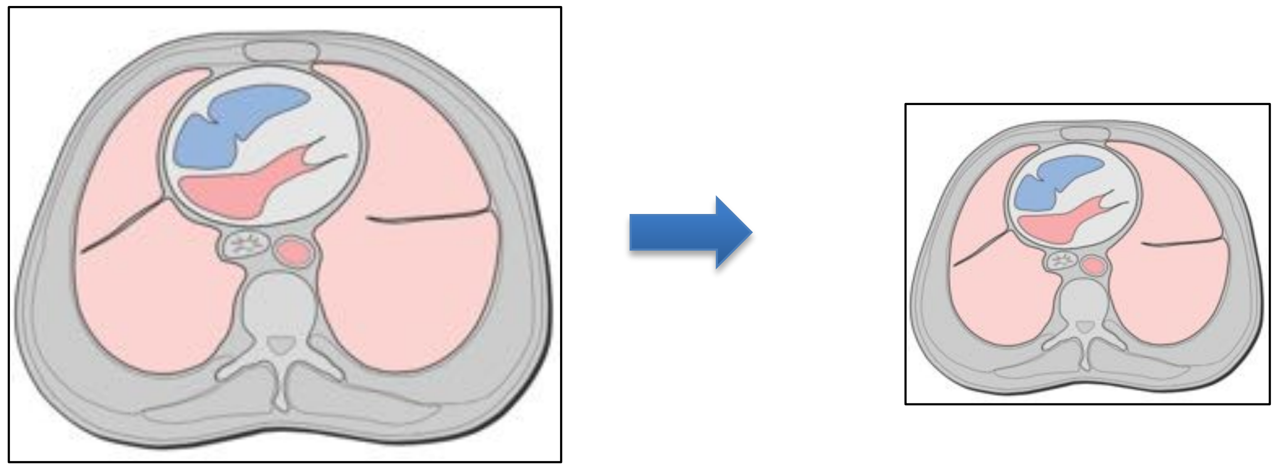
Rotation



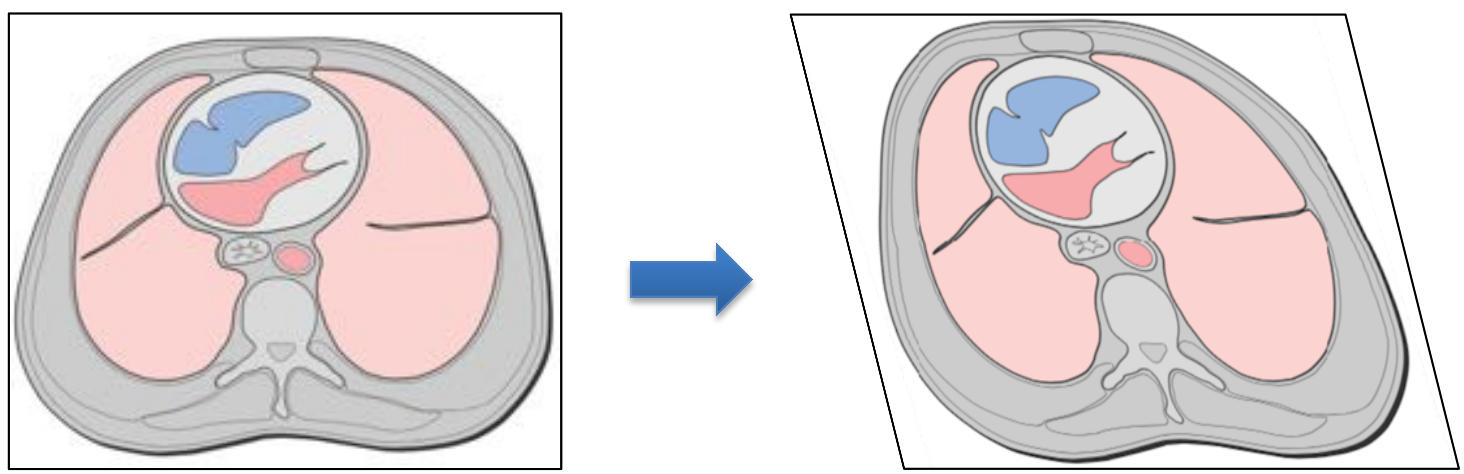


Coordinate Transformations

Scaling



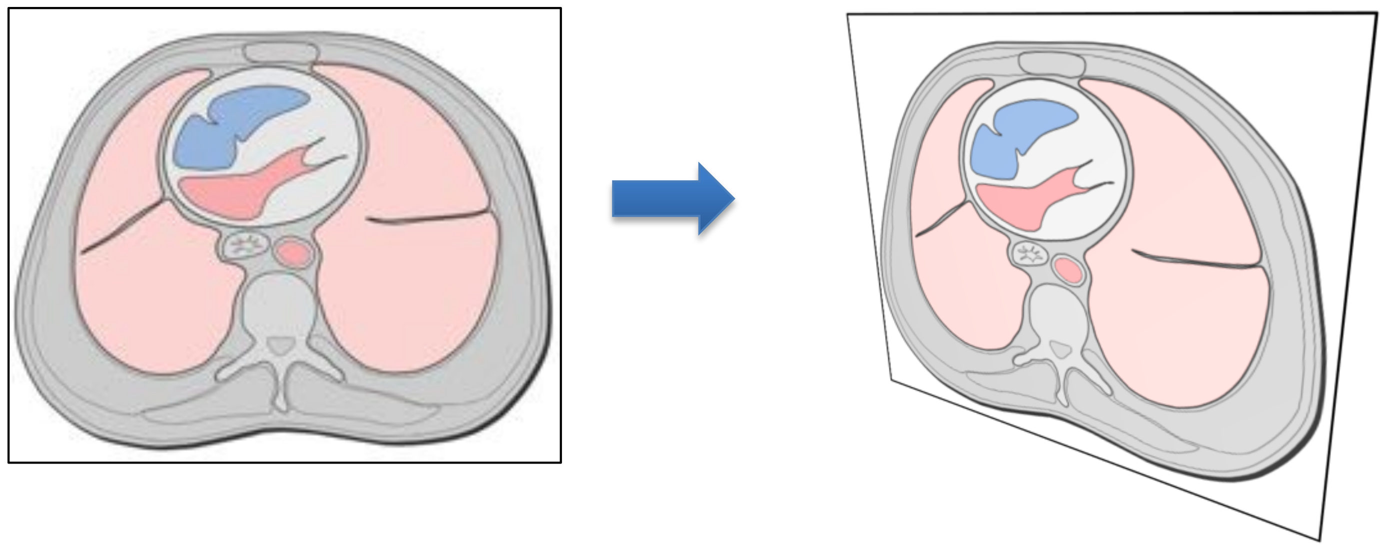
Shear



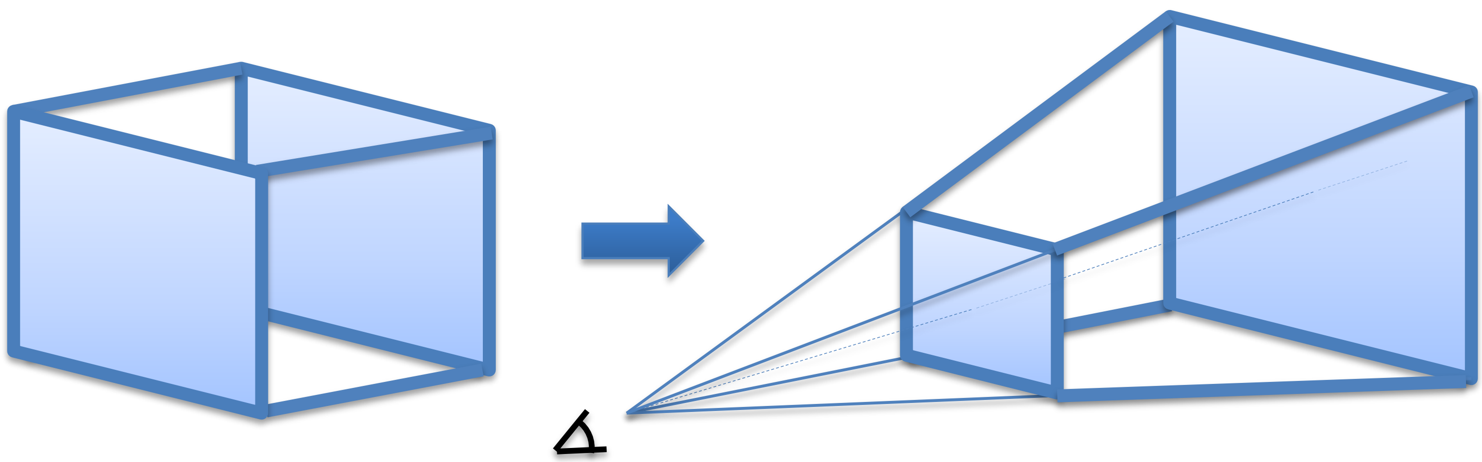
Coordinate Transformations

Perspective (2D)

(*more common in document processing*)



Perspective (3D)



3D Cartesian Coordinates Transforms with Matrices

Affine Transformations

Rigid Transformations

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Rotation about X

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Rotation about Y

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Rotation about Z

addition, not multiplication

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

Translation

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

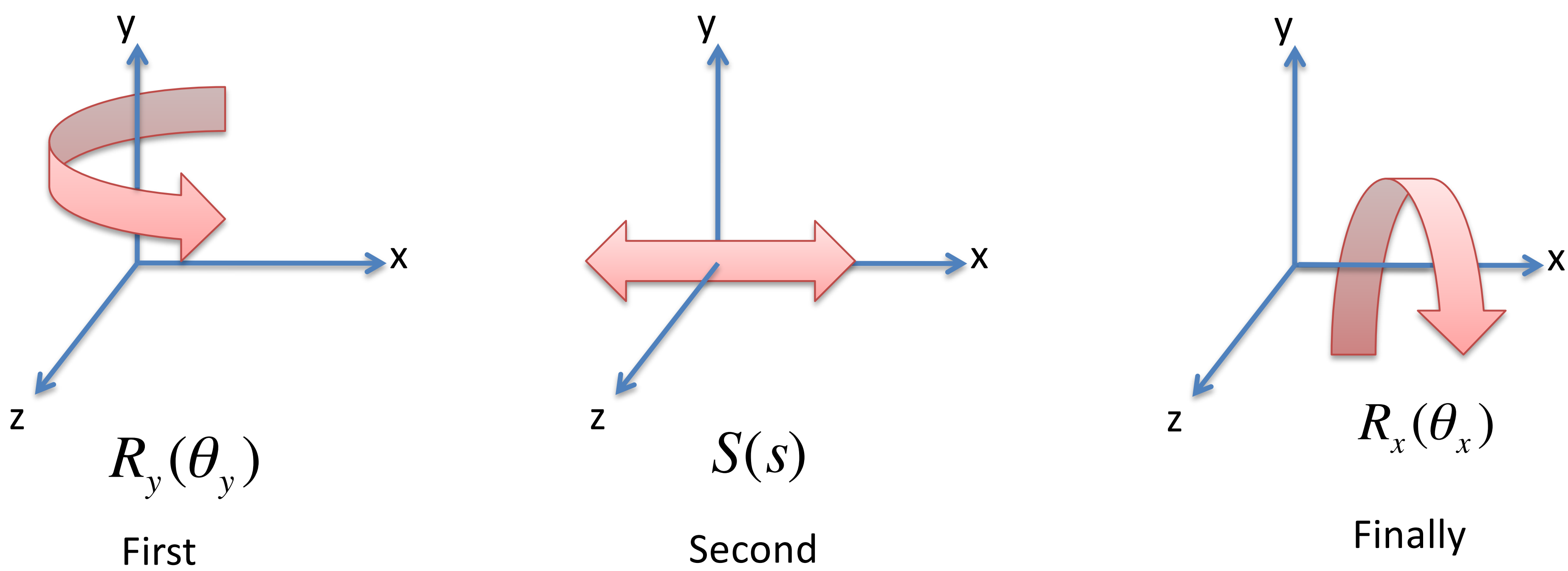
Scaling

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 1 & Sh_{xy} & Sh_{xz} \\ Sh_{yx} & 1 & Sh_{yz} \\ Sh_{zx} & Sh_{zy} & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Shearing

Composing Multiple Transforms

For example,



$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \left(R_x(\theta_x)\left(S(s)\left(R_y(\theta_y)\right) \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}\right)\right)\right)$$

For a given transformation, pre-compute the matrix to apply to thousands or millions of vertices
Take advantage of matrix multiplication associativity

The fly in the ointment… *__translation is not a matrix multiply__*

Homogenous Coordinates

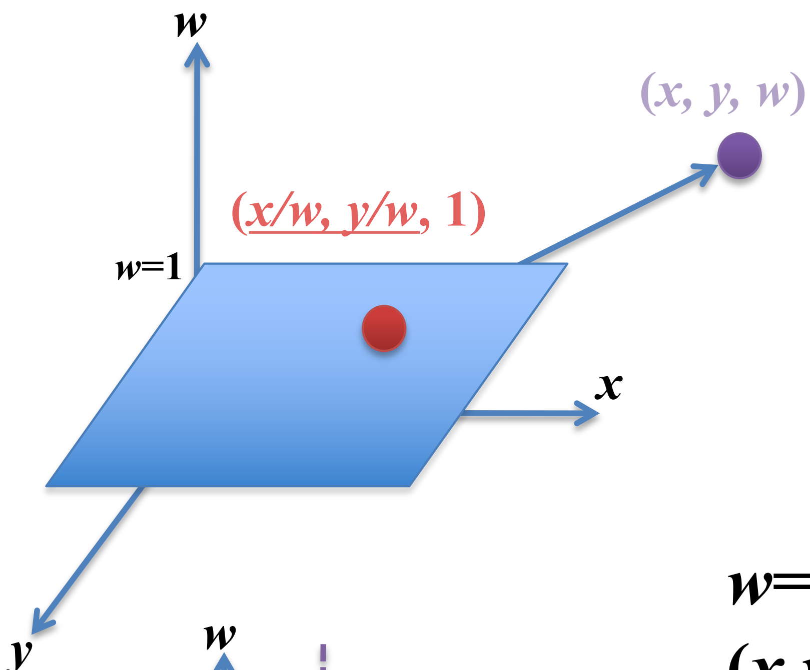
Homogeneous Coordinates to the Rescue

We've seen this trick again and again (think 2D image as a surface)

- Go one dimension higher to convert *all* transformations into multiplication
  - (x,y,z) is now represented as $(x,y,z;w)$
  - **Implements translation as a matrix *multiply***
- To convert to homogeneous:
  - $(x,y,z) \rightarrow (x,y,z;1)$
- To go back to regular coordinates:
  - $(x,y,z;w) \rightarrow (x/w,\ y/w,\ z/w)$
- The extra degree of freedom adds redundancy
  - Non-zero multiples represent the same point
  - $(1,2,3;1)$ is the same spatial point as $(2,4,6;2)$

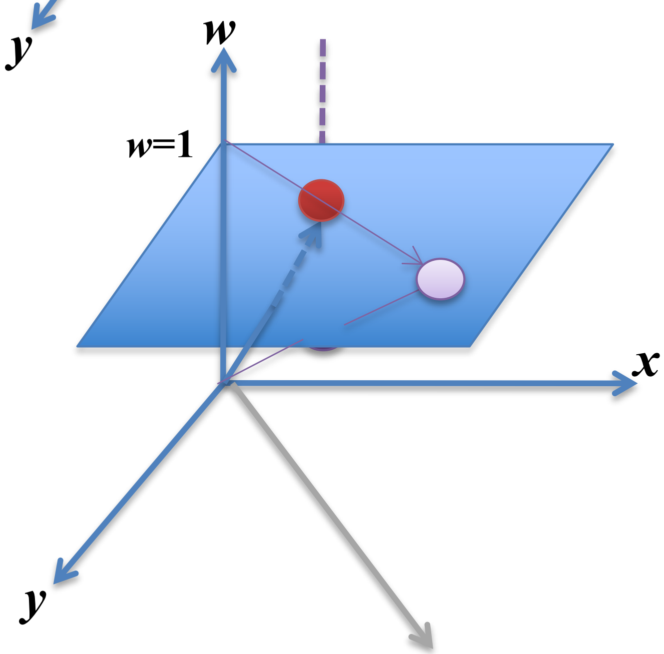
Geometric Interpretation

2D Homogeneous Coordinates in 3D Space



Project all homogeneous points to w=1 plane through the origin

All represent the same 2D point:
(1,2;1) (2,4;2) (3,6;3)



w=0 is special:
(x,y,z;0) is a point at infinity in the direction (x,y,z)

In the limit as w goes to zero, these 2D homogeneous points lie on the same x-y line pointing away from the origin:

(1,2;1) (1,2;0.1) (1,2;0.01) etc.
(1,2) (10,20) (100,200) etc.

3D Homogeneous Coordinates Transforms with Matrices

Affine Transformations

Rigid Transformations

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Rotation about X

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Rotation about Y

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Rotation about Z

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Translation

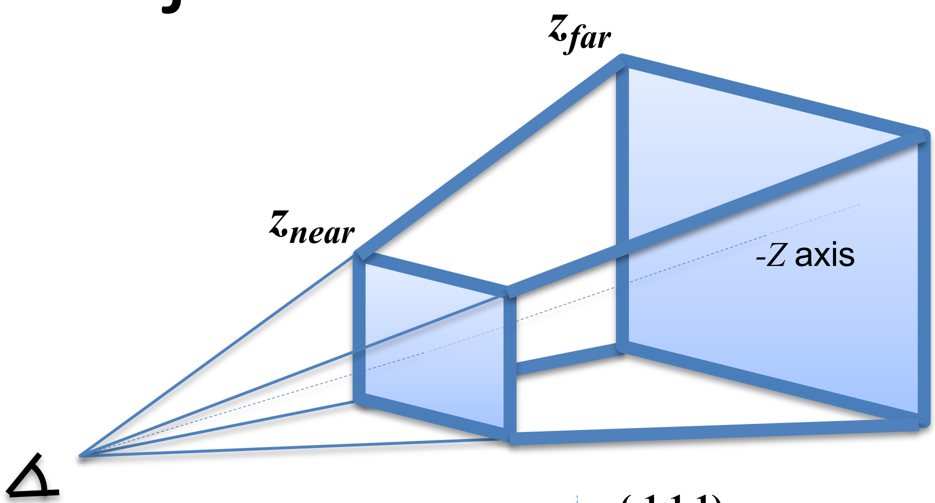
$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

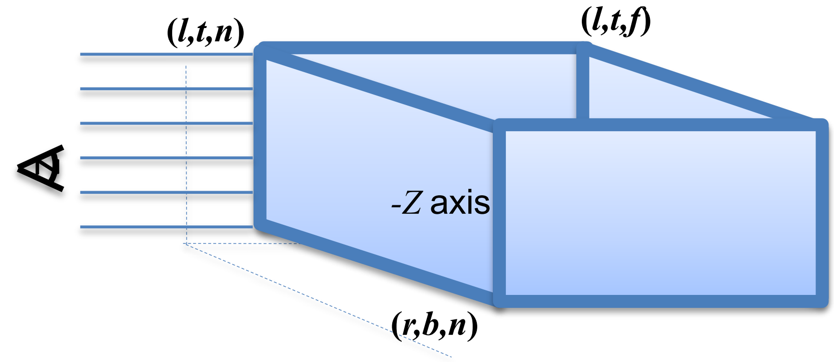
Scaling

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & Sh_{xy} & Sh_{xz} & 0 \\ Sh_{yx} & 1 & Sh_{yz} & 0 \\ Sh_{zx} & Sh_{zy} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

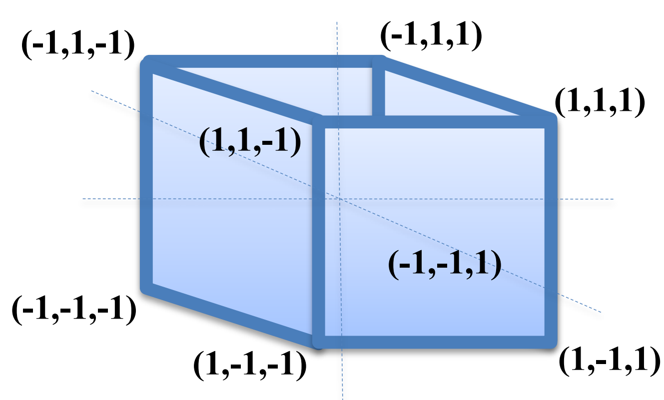
Shearing

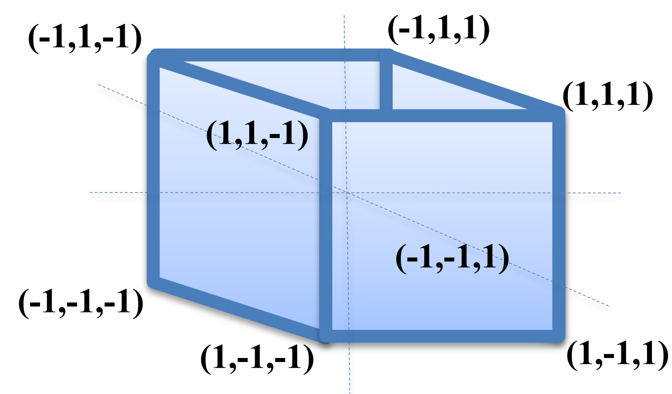
Projection Matrix in Homogeneous Coordinates





Normalized Device Coordinates





$$\begin{bmatrix} \cot(fov_x/2) & 0 & 0 & 0 \\ 0 & \cot(fov_y/2) & 0 & 0 \\ 0 & 0 & \dfrac{z_{far}+z_{near}}{z_{near}-z_{far}} & \dfrac{2\cdot z_{far}\cdot z_{near}}{z_{near}-z_{far}} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

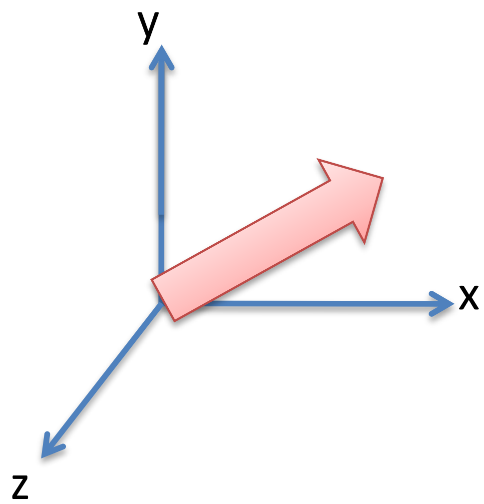
Perspective Projection

$$\begin{bmatrix} \dfrac{2}{r-l} & 0 & 0 & -\dfrac{r+l}{r-l} \\ 0 & \dfrac{2}{t-b} & 0 & -\dfrac{t+b}{t-b} \\ 0 & 0 & -\dfrac{2}{f-n} & -\dfrac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Orthographic Projection

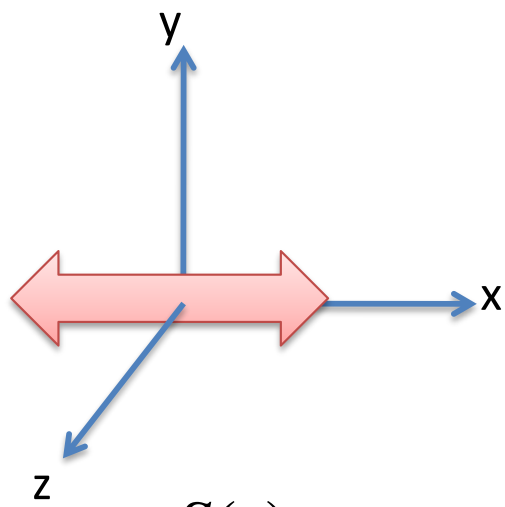
Composing Transforms

For example,



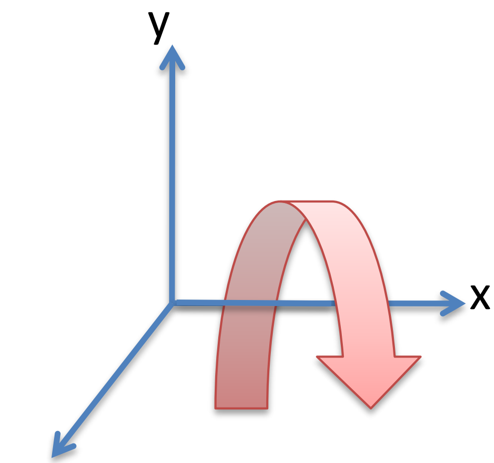
$T(t)$

First



$S(s)$

Second



$R_x(\theta_x)$

Finally

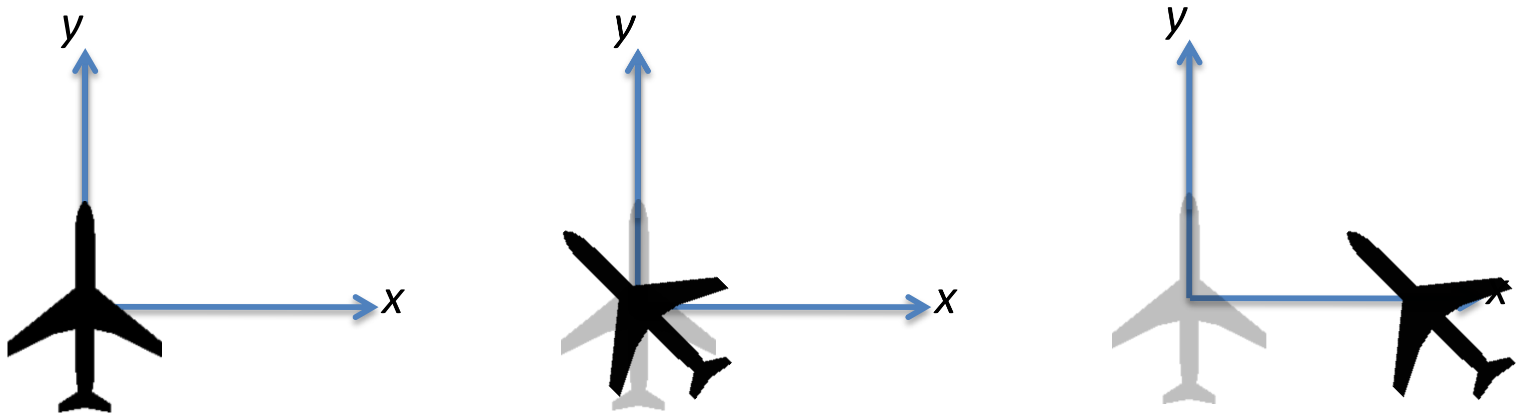
$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = R_x(\theta_x) \cdot S(s) \cdot T(t) \quad \cdot \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Now all transformations are matrix multiplies *<u>including translation</u>*

Again, note the ordering of the matrix multiplications

Order of Composing Transforms

Example: First, 45° rotation. Then +X translation.



$\vec{x}$ $\quad\quad$ $\vec{x}' = R\vec{x}$ $\quad\quad$ $\vec{x}'' = T\left(R\vec{x}\right)$

Order of transforms matters: $TR\vec{x} \neq RT\vec{x}$ *matrix multiplication not commutative!*

Transformations are done in a fixed global coordinate system and read right-to-left (i.e., inside-out)

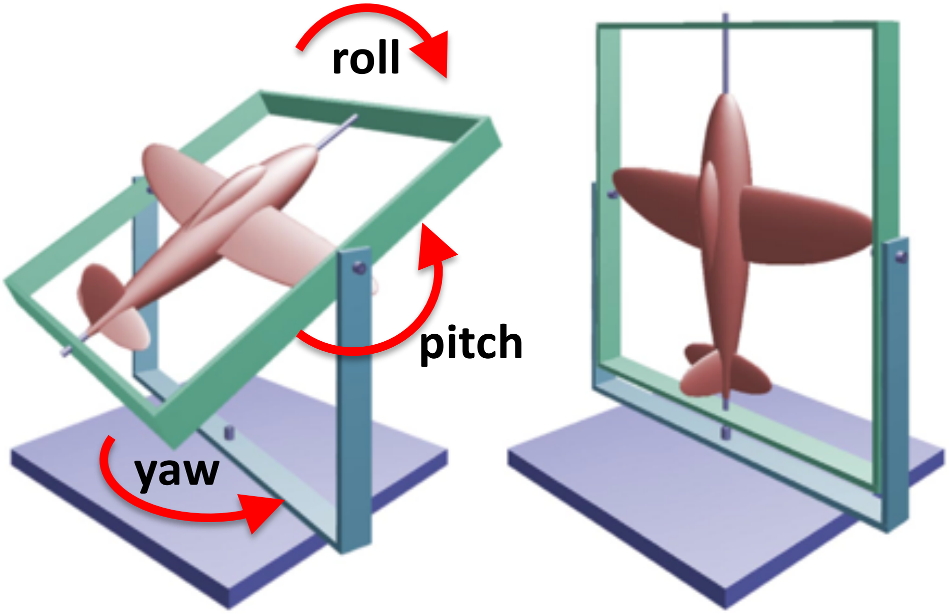
Alternatively, transformations can be thought of in the object's (or camera's) coordinate system if read from left-to-right (i.e., outside-in)

But beware, non-rigid transformations also transform the object's coordinate axes

When might you want to use the object's or camera's coordinate system?

3D Rotations with Quaternions

A Closer Look at 3D Rotations



At 90° pitch, yaw and roll are now aligned!
(i.e., gimbal lock)

3D rotations have a degeneracy when two rotational axes approach alignment

Examples of gimbal lock:

Apollo 11

Robotic Arm

Animation

http://www.fho-emden.de/~hoffmann/gimbal09082002.pdf

3D Rotation with Quaternions:

An Extension of Complex Numbers from 2D to 4D

$\pm I, \pm J, \pm K$ are all different imaginary numbers

$$I^2 = J^2 = K^2 = IJK = -1$$

$$IJ = K \quad JK = I \quad KI = J$$

$$JI = -K \quad KJ = -I \quad IK = -J$$

Quaternions form a 4D vector: (a,b,c,d) is $a+bI+cJ+dK$

Unit length quaternions represent rotations in $\mathbf{R}^3$

$$|a + bI + cJ + dK| = \sqrt{a^2 + b^2 + c^2 + d^2}$$

Angle of rotation θ given by: $a = \cos(\theta/2)$

Axis of rotation given by: (b,c,d)

This arbitrary axis of rotation is what makes quaternions so useful!

Quaternions:

Composing Multiple Rotations

$$R = \begin{bmatrix} a^2+b^2-c^2-d^2 & 2bc-2ad & 2bd+2ac & 0 \\ 2bc+2ad & a^2-b^2+c^2-d^2 & 2cd-2ab & 0 \\ 2bd-2ac & 2cd+2ab & a^2-b^2-c^2+d^2 & 0 \\ 0 & 0 & 0 & a^2+b^2+c^2+d^2 \end{bmatrix}$$

A quaternion represented as a 4x4 homogeneous transformation matrix

$$pq = (p_a q_a - p_b q_b - p_c q_c - p_d q_d,\ \ p_a q_b + p_b q_a + p_c q_d - p_d q_c,$$
$$p_a q_c - p_b q_b + p_c q_a + p_d q_b,\ p_a q_d + p_b q_c - p_c q_b + p_d q_a)$$

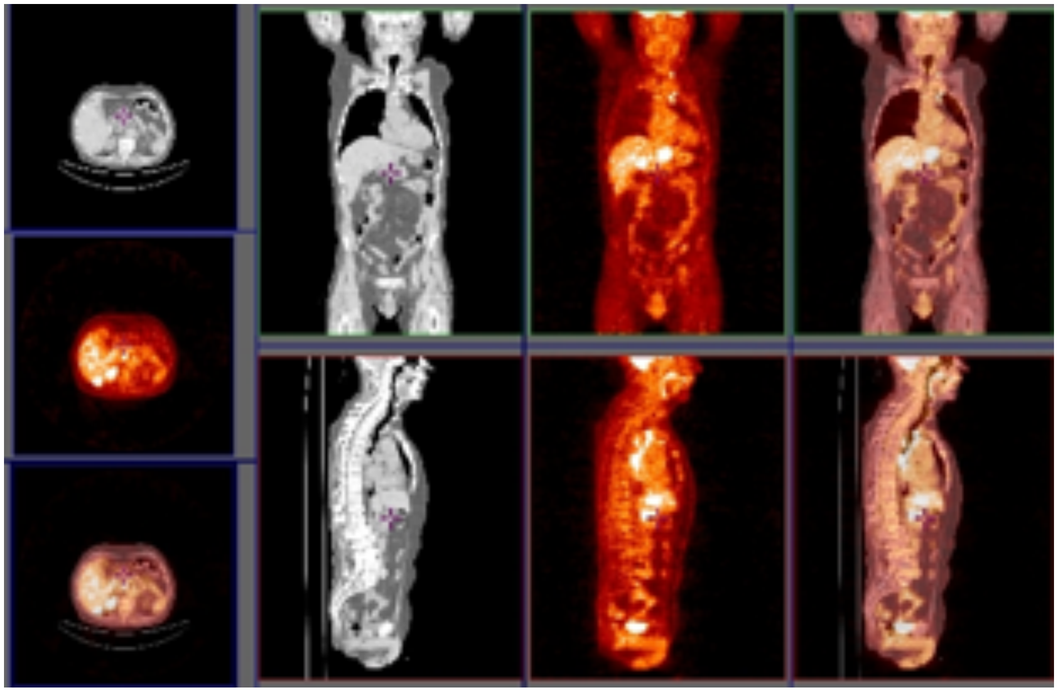
- Just like matrices, quaternions are not commutative: $pq \neq qp$

Successive rotations can be composed as quaternion multiplications: $pqrs$

- **Right-to-Left can be viewed in global coordinate system**
- **Left-to-Right can be viewed in object's coordinate system**

Image Registration

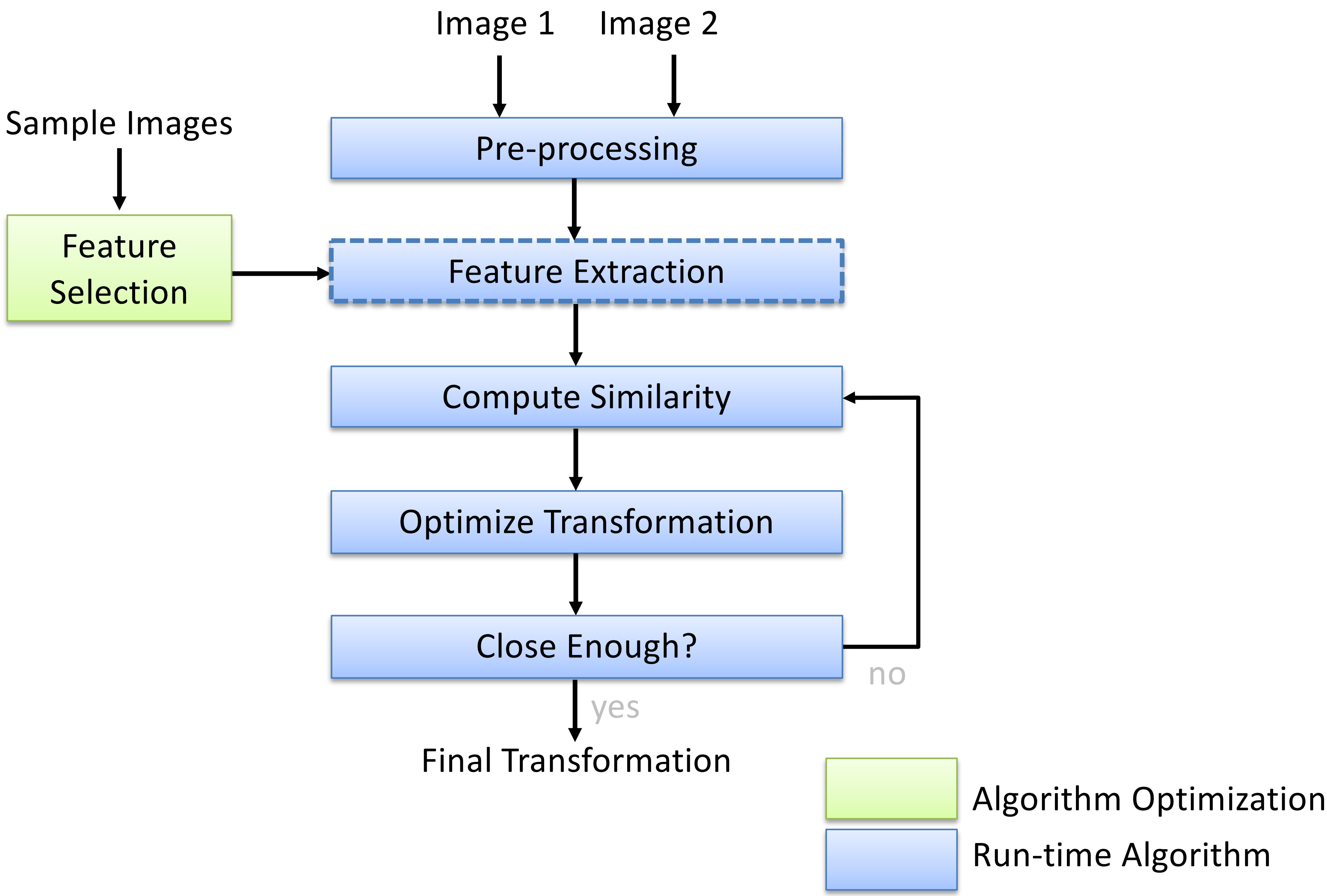
Image Registration



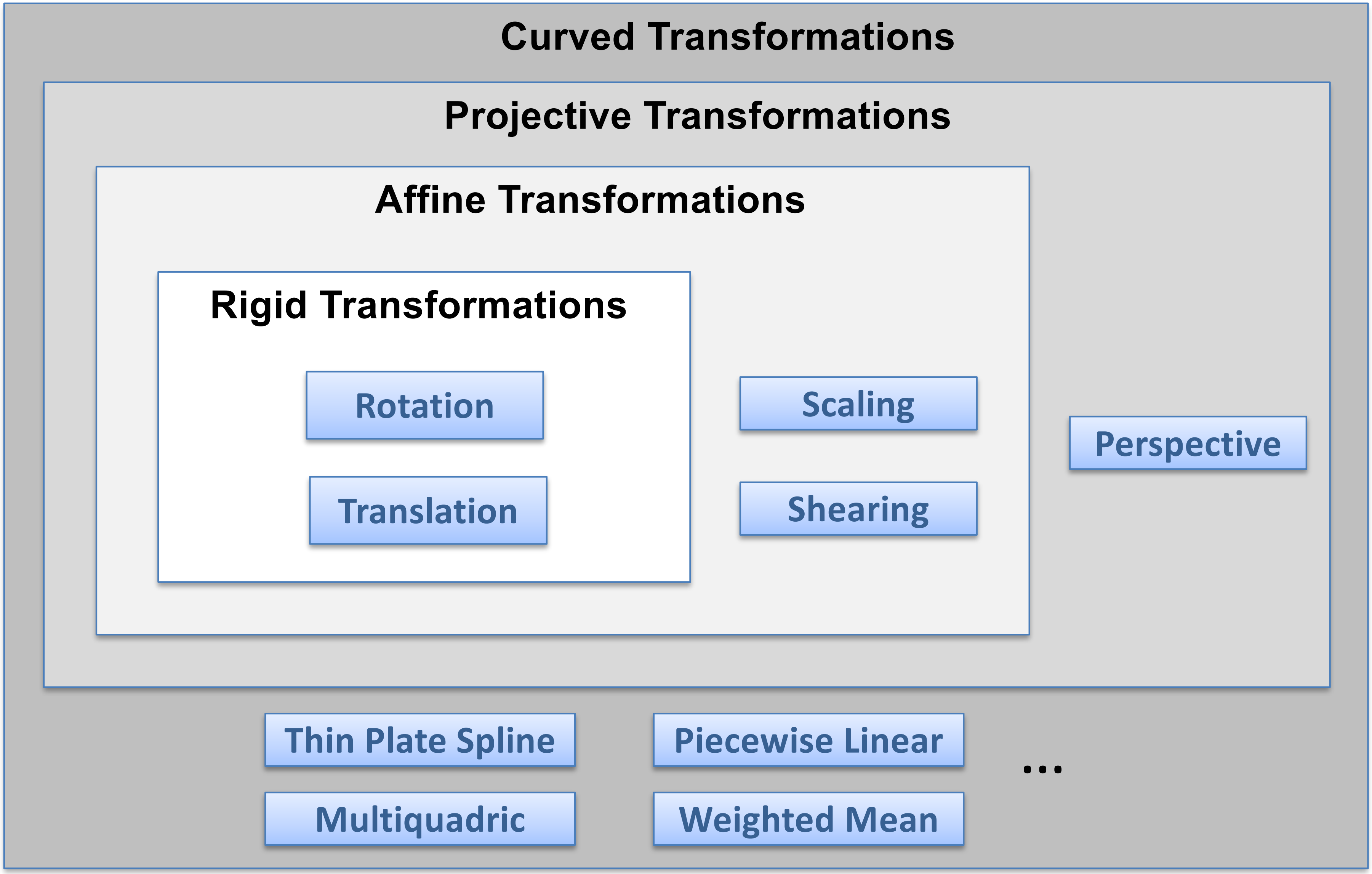
"Image registration is the process of aligning images so that the correspondences between them can be seen more easily."

$$\arg\max_{T}\left\{similarity(Image_1, T\{Image_2\})\right\}$$

Typical Registration Algorithm Architecture



Types of Image Transformations

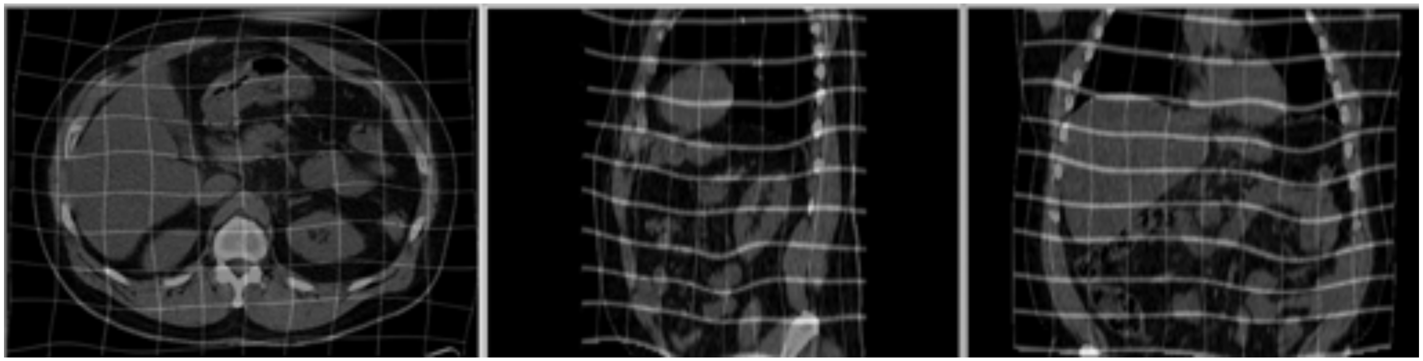


Need for Different Transformations

- Rigid Transformation
  - Hard anatomy (head)
  - Bones
- Affine Transformation
  - Scaling due to size change
  - Shear due to body leaning
- Perspective Transformation
  - Pictures from two different perspectives
  - 2D/3D Registration

Need for Different Transformations

- Curved Transformation
  - Soft Tissue Deformation
  - Changes in Body Position
  - Surgical Changes
  - Growth
    - Developmental Growth
    - Tumor Growth
    - Weight Changes
  - Comparison between different subjects
    - e.g., to a reference atlas
  - **Adds a *significant* computational burden**



slicer.org

Optimization Methods

- Objective function (AKA cost function) is our similarity metric
- Parameter space defined by transformation parameters
- Examples
  - Gradient Descent
  - Powell's method
  - Nelder-Meade downhill simplex (amoeba)
  - Levenberg Marquardt
  - Particle Swarm Optimizer
  - Limited memory Broyden Fletcher Goldfarb Shannon
- Multi-resolution approaches are very helpful
  - Progressively less blurring of input images
  - Adapting optimizer step sizes at each scale

Example Similarity Metric: Mean Squared Difference

$$MSD = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\left(I_1(i,j) - T\{I_2\}(i,j)\right)^2$$

- Amongst the simplest similarity metrics
- I_1 = fixed image
- I_2 = moving image
- Usually $M*N$ is very large so first compute joint histogram, then estimate *MSD* from joint histogram

What assumptions does this metric make on the intensity scale?
In what types of registration problems are these assumptions violated?

Example Similarity Metric: Correlation Coefficient

$$\rho = \frac{E\left[\left(I_1 - \mu_1\right)\left(T\{I_2\} - \mu_2\right)\right]}{\sigma_1 \sigma_2}$$

- More generalized than mean squared difference
- Also generally computed from joint histogram

How is this more general purpose than mean squared difference?

Example Similarity Metric: Kappa Statistic

		I_1	
		1	0
$\mathbf{T}\{I_2\}$	1	a	b
	0	c	d

$$N = a + b + c + d$$

$$p_o = \frac{a+d}{N}$$

$$p_e = \frac{a+c}{N}\frac{a+b}{N} + \frac{b+d}{N}\frac{c+d}{N}$$

$$\boxed{\kappa = \frac{p_o - p_e}{1 - p_e}}$$

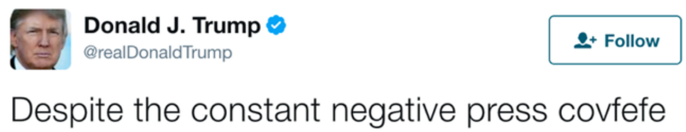
- Useful for comparing segmented images
  - e.g., pixels are categorical variables
- p_o is observed rate of agreement
- p_e is expected rate of agreement by chance

Example Similarity Metric: Mutual Information

- Extremely useful for comparisons between image intensity scales that may be very different (not even monotonic transforms of each other)
- Has become a very popular similarity metric for image registration

A Brief Diversion into Information Theory

Message Entropy



A set of messages (e.g., tweets) consisting of a string of n symbols (e.g., 280 chars) where each symbol has s different values has s^n possible variations (e.g., 26^{280} possible tweets)

A measure of complexity (or uncertainty) for *all possible messages* (e.g., tweets) that increases linearly with n would be:

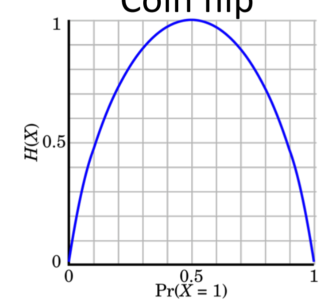
$$H = \log s^n = n \log s$$

Complexity of *a specific message* (e.g., tweet) can be determined by examining the histogram of symbol (e.g., letter) occurrences:

$$H = \sum_i p_i \log \frac{1}{p_i} = -\sum_i p_i \log p_i$$

(Shannon Entropy)

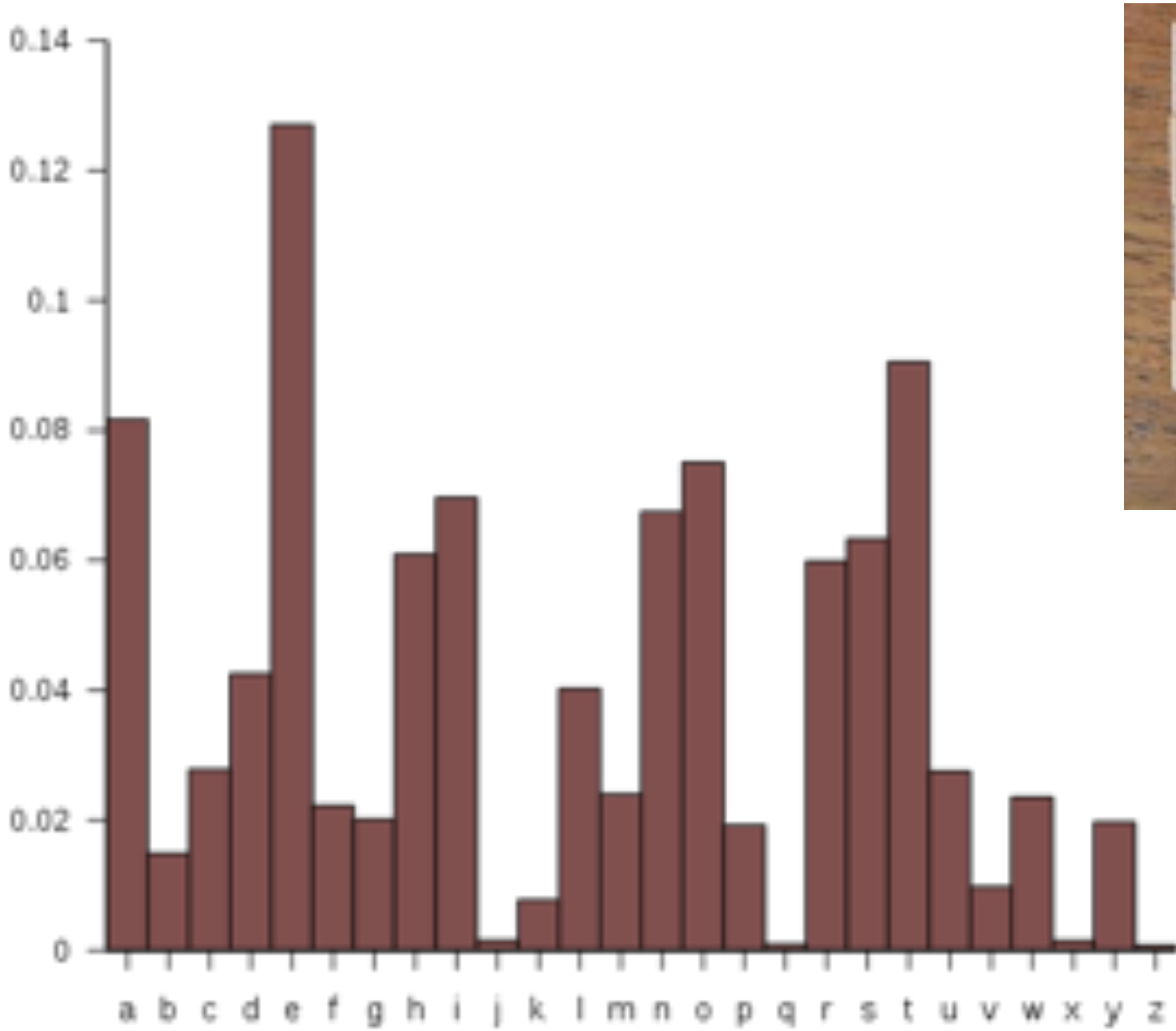
Coin flip



Maximized when all probabilities are equal (flat histogram)

Information Content of a Tweet

But of course real language doesn't use letters in equal frequencies



Letter Frequency in the Oxford Dictionary

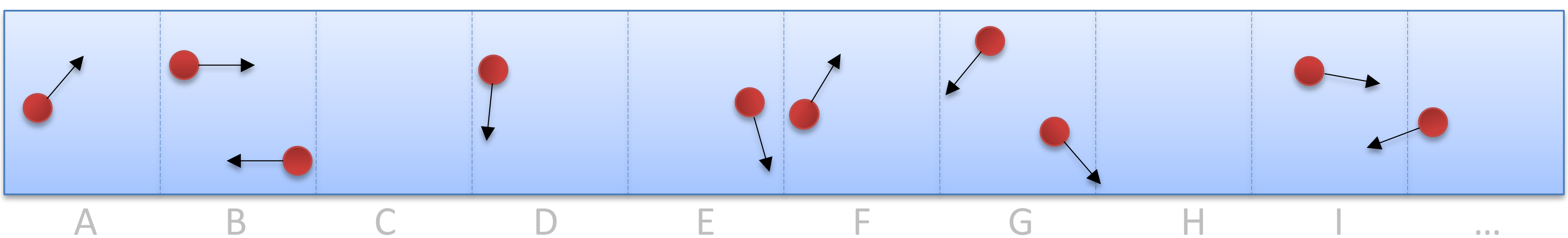
Real language has low entropy (not flat)

How does this relate to thermodynamic entropy?

Imagine a long glass tube full of gas molecules that is considered in segments

Partitioned tube is like a histogram; molecules are like counts in the histogram

Low Entropy (1 possible arrangement)

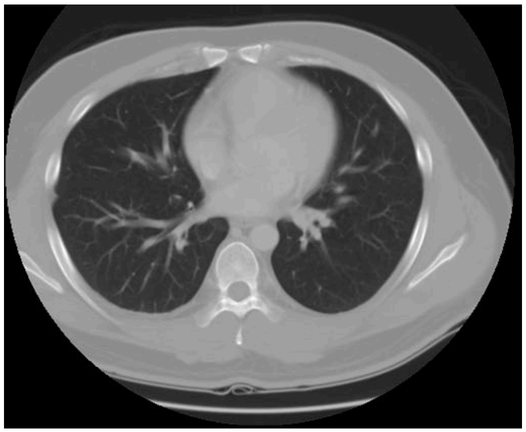


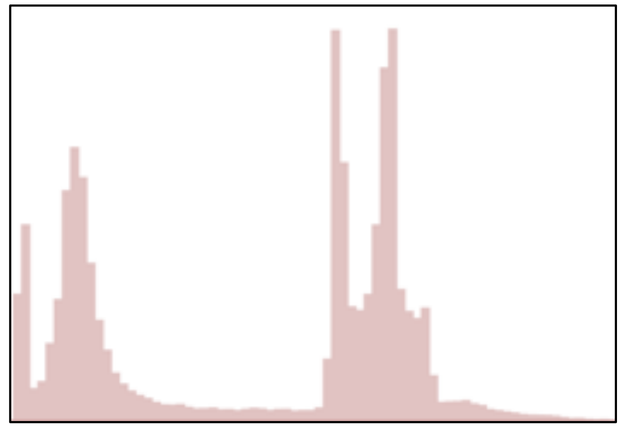
High Entropy (many possible arrangements)

Image Entropy

Message = Image (instead of tweet) Symbol = Pixel Value (instead of letter)

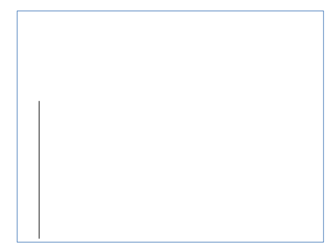
Probability values come from histogram of pixel values



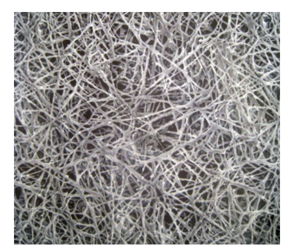


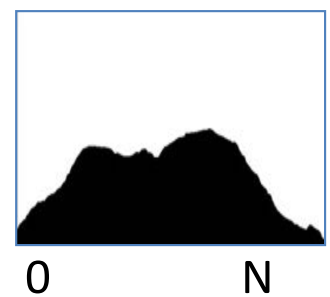
Entropy is a measure of pixel histogram dispersion

A uniform image of 1 pixel value (peaky histogram) contains little information; **low entropy**



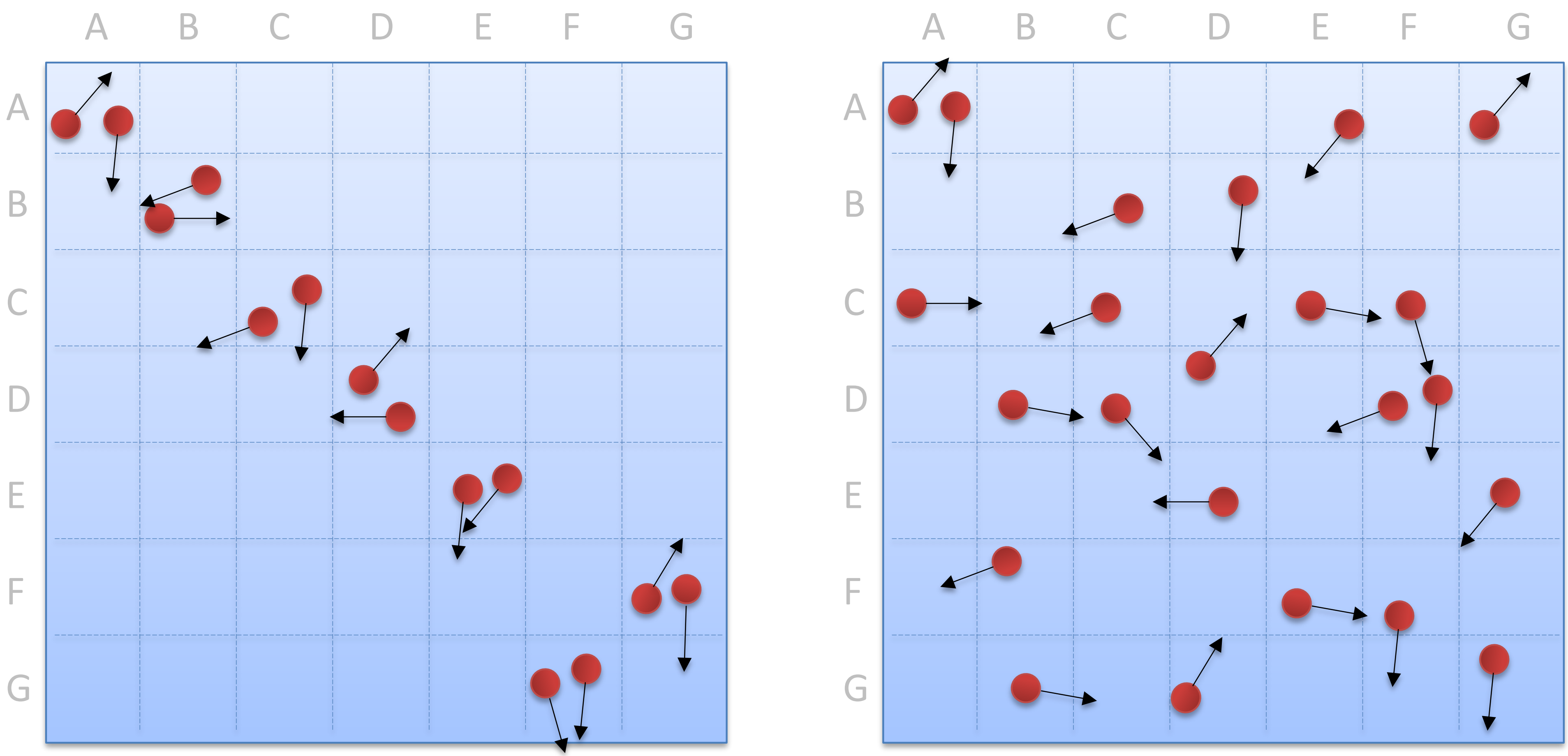
An image of highly distributed pixel values (broad histogram) has high information; **high entropy**





Now we allow molecules to vary along two axes

Now imagine gas molecules moving around in plane



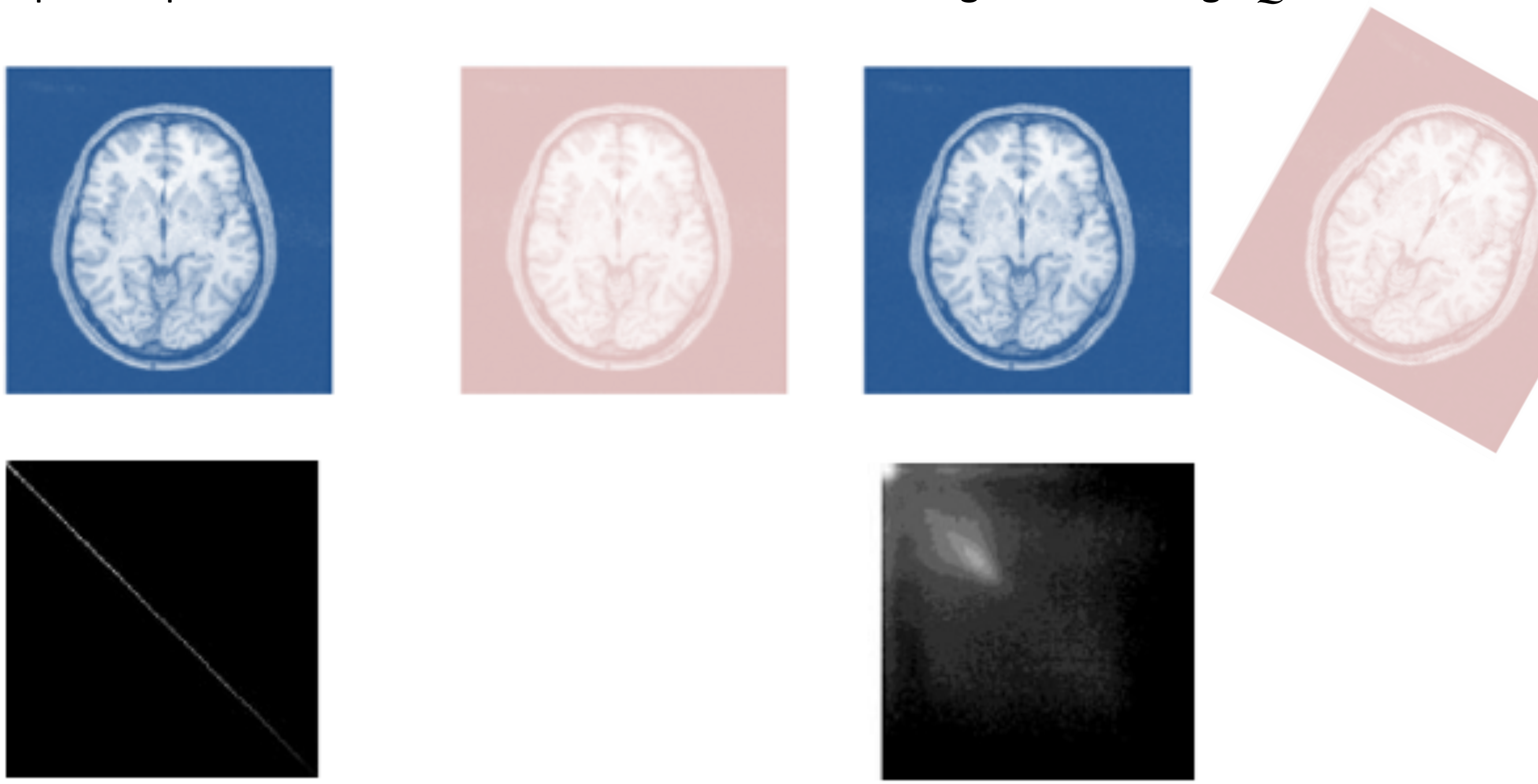
Low Joint Entropy **High Joint Entropy**

If we observed tightly clustered molecules,
information about one axis would tell us quite a bit of information about the other axis.
(note that they don't have to cluster on the diagonal to have low joint entropy)

Joint Entropy of Aligned Images

Histogram pixel values come from image pixel ***joint*** histogram of aligned images

(p_i, q_i) is the pair of pixel values at the same coordinates in image P and image Q



Low Joint Entropy

High Joint Entropy

Viola and Wells, MIT, 1997
Pluim et al, IEEE TMI 2003

Entropy and Mutual Information

Entropy:

$$H(A) = -\sum_i p(a_i)\log p(a_i)$$ Marginal Entropy

$$H(A,B) = -\sum_i \sum_j p(a_i,b_j)\log p(a_i,b_j)$$ Joint Entropy

$$H(A \mid B) = -\sum_i \sum_j p(a_i \mid b_j)\log p(a_i \mid b_j)$$ Conditional Entropy

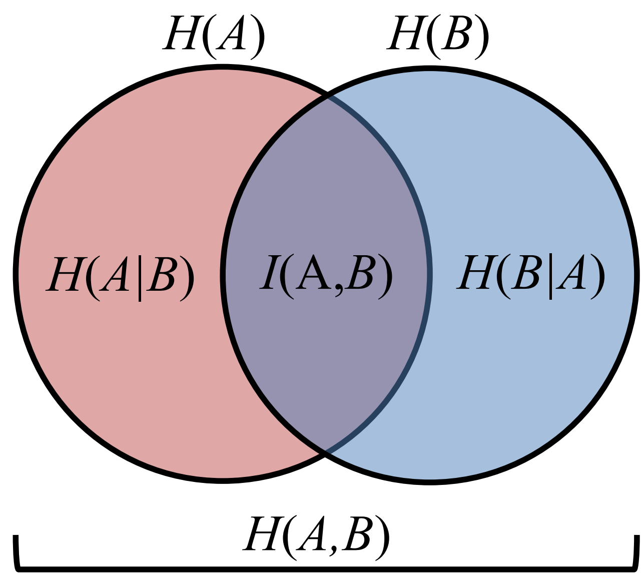
Mutual Information:

$$I(A,B) = \sum_i \sum_j p(a_i,b_j)\log \frac{p(a_i,b_j)}{p(a_i)p(b_j)}$$

$$= H(A) + H(B) - H(A,B)$$

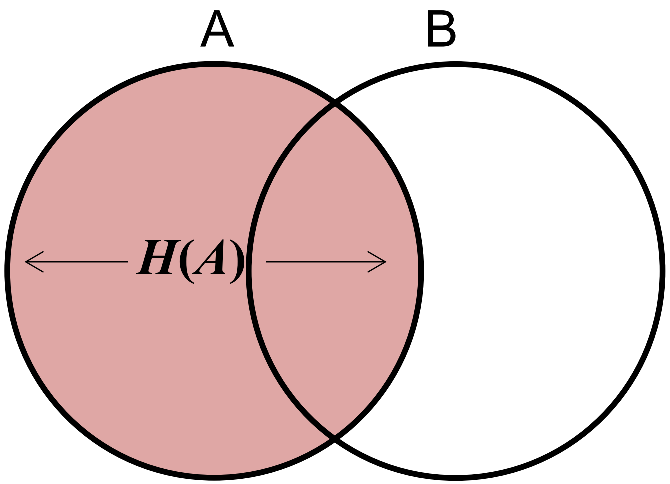
$$= H(A) - H(A \mid B)$$

$$= H(B) - H(B \mid A)$$

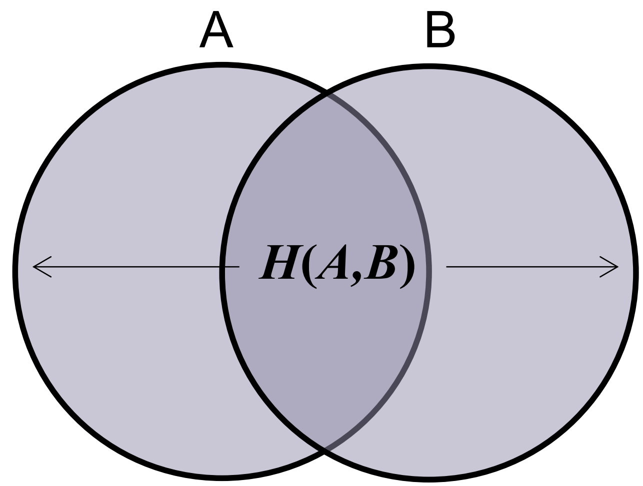


- Difference between marginal entropies (assumes independence) and joint entropy
- How much does knowing one variable reduces the uncertainty about the other

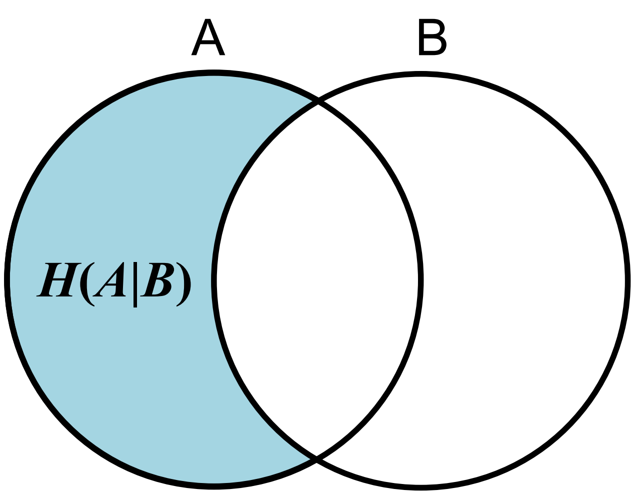
Entropy and Mutual Information



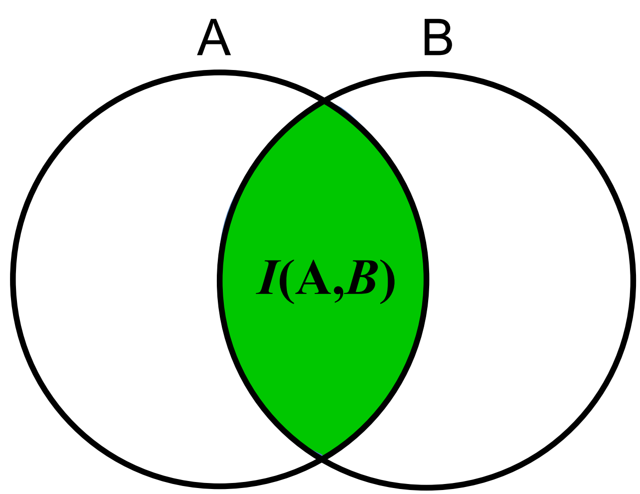
Marginal Entropy



Joint Entropy

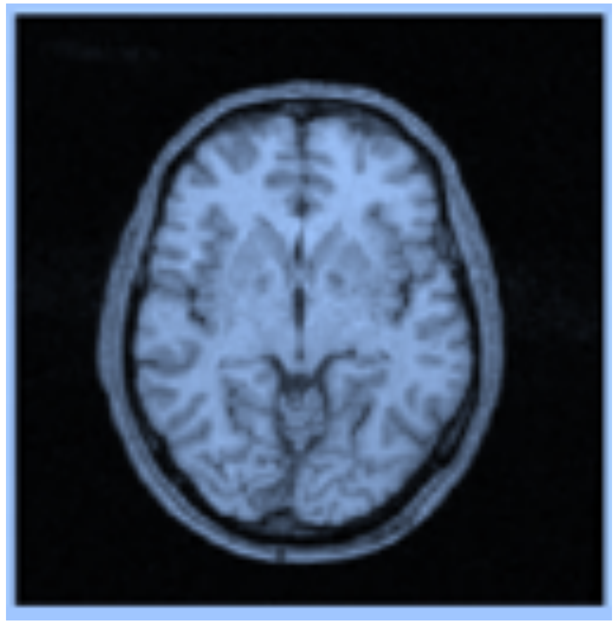


Conditional Entropy



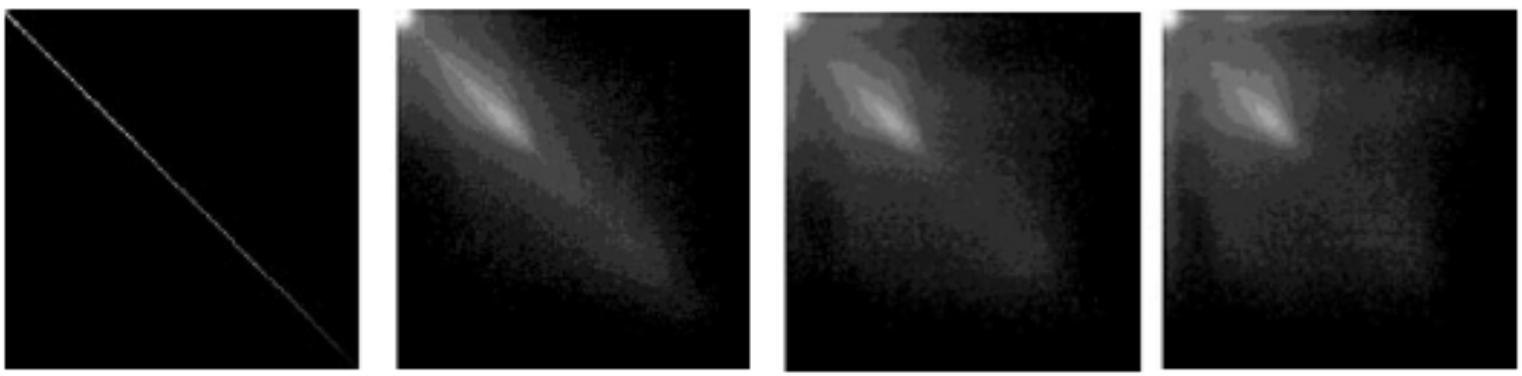
Mutual Information

Maximization of Mutual Information



Joint histogram of image with itself rotated at various angles

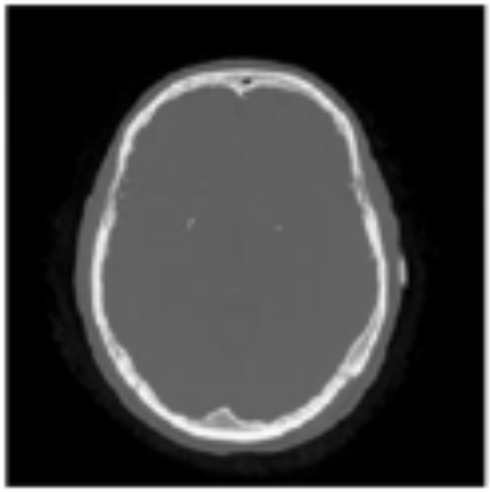
Joint Histograms:



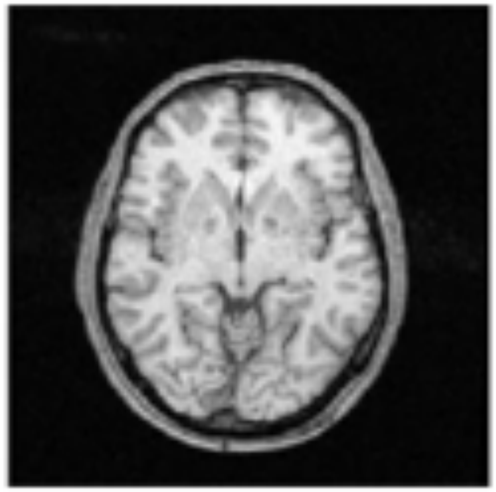
Rotation: 0° 2° 5° 10°

Pluim et al, IEEE TMI 2003

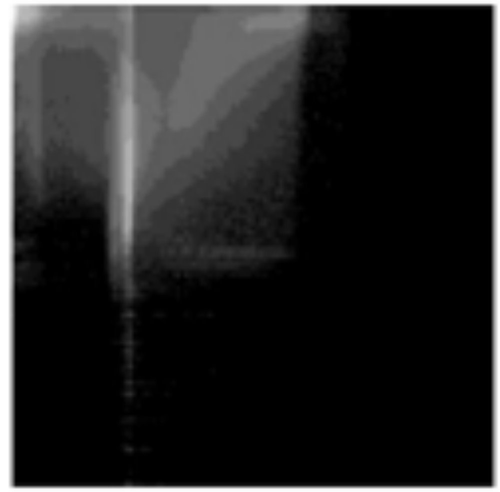
Mixed Modality Registration



CT



MR



Joint Histogram

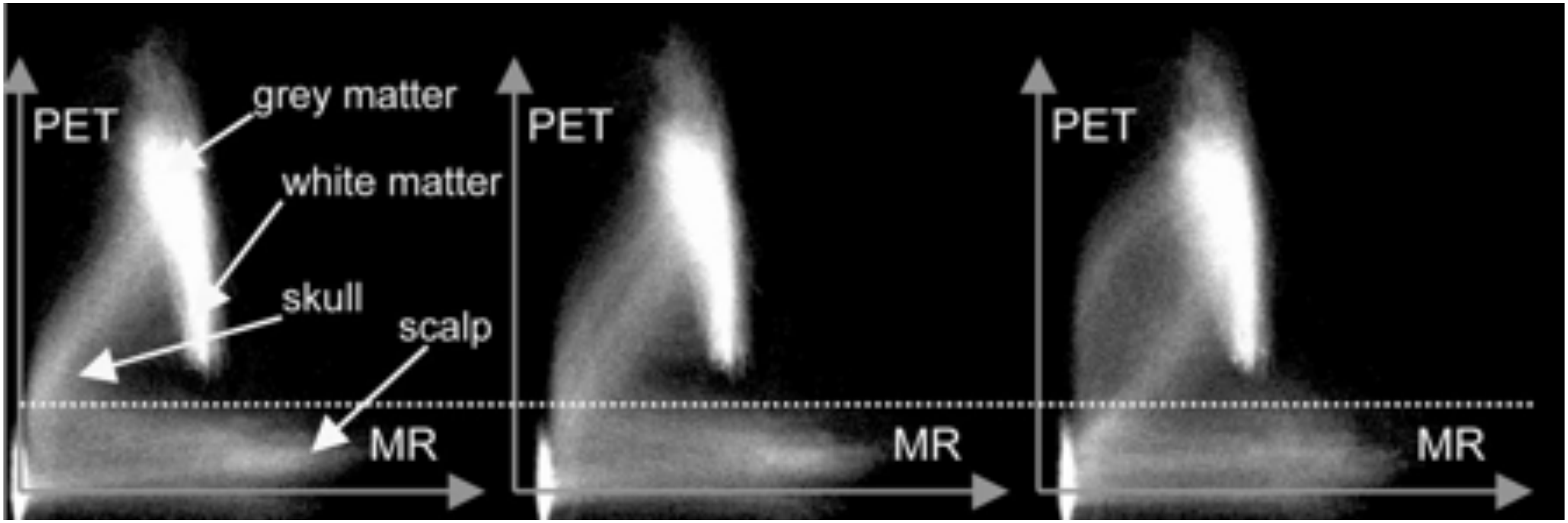
Non-monotonic transform between intensity scales is OK!

(low joint entropy doesn't have to be on the diagonals)

Unlike cross-correlation, sum of squared differences, ratio image uniformity that depend on numerically similar pixel values between images

In the joint histogram, which modality is on the horizontal axis and which modality is on the vertical axis?

PET/MR Joint Distribution Example



Translational
Misalignment: 0 mm 2 mm 5 mm

It can even do well with low resolution functional imaging with fewer spatial details

Where do you see the most notable difference in mutual information?

Hallpike and Hawkes, Imaging 2002

What does it mean for me?

- Today’s Topics
  - Coordinate Systems
    - 2D
    - 3D including homogeneous coordinates
  - 3D Rotations
    - Quaternions
  - Image Registration Algorithm
    - Transforms
    - Optimizers
    - Similarity Metrics
      - Mutual Information
- Dealing with translation and rotation may be more subtle than they first seam

Next Lecture:

Radiomics