# Lecture 13
# Theory of Registration

ch. 10 of *Insight into Images* edited by Terry Yoo, et al.

Spring 2024

16-725 (CMU RI) : BioE 2630 (Pitt)

Dr. John Galeotti

## Registration?

- The process of aligning a target image to a source image
- More generally, determining the spatial transform that maps points in one image to corresponding points in the other image

# Registration Criteria

- What do we compare to determine alignment?
- Three general philosophies:
  - Intensity-based
    - This is what we've mostly seen so far
    - Compare actual pixel values from one image to another
    - Comparison can be complex, such as with mutual information
  - Segmentation-based
    1. Segment the images
    2. Register the binary segmentations
  - Landmark-based
    - Mark key points in both images (often by hand)
    - Derive a transform that makes every pair of landmarks match.

## Types of Spatial Transforms

- Rigid (rotate, translate)
- Affine (rigid + scale & shear/skew)
- Deformable (free-form = affine + vector field)
- Many others

# ITK (Legacy) Registration Flowchart, with Theory Notation

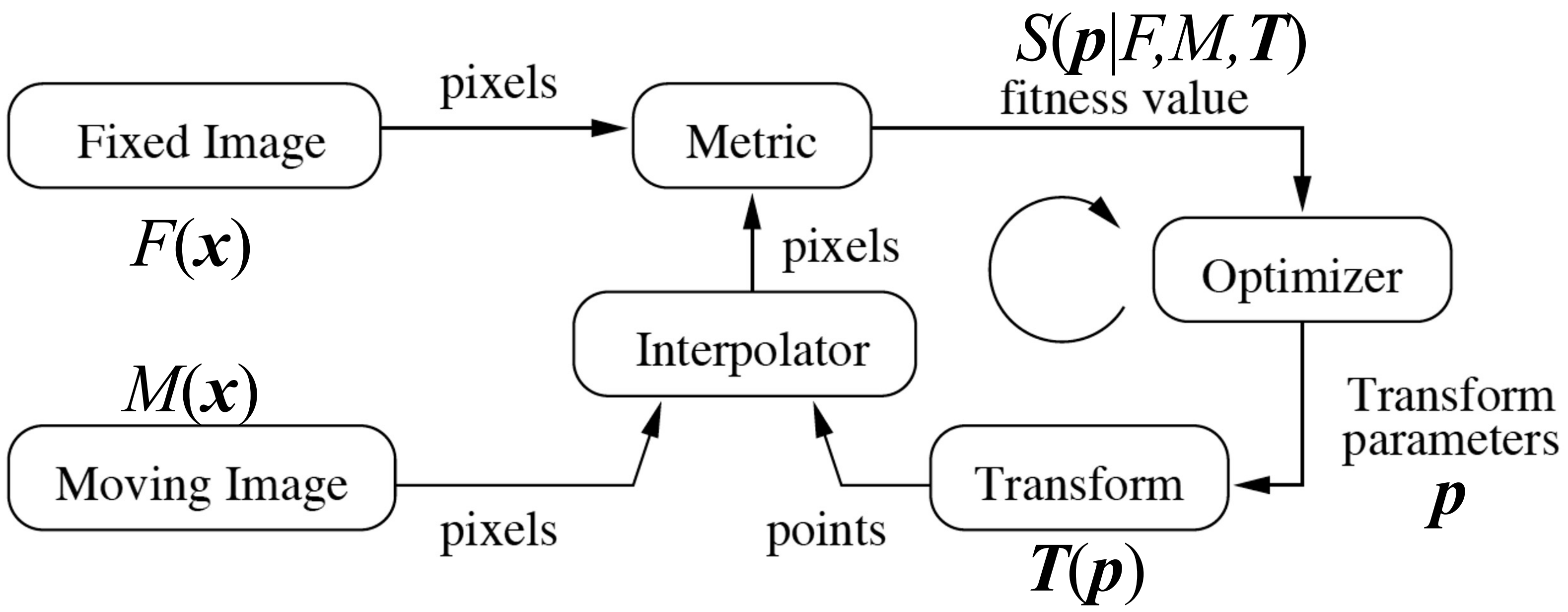


Figure 8.2 from the *ITK Software Guide* v 2.4, by Luis Ibáñez, et al., also showing the notation used by ch. 10 of *Insight into Images*, by Terry Yoo, et al.

# Example Transform Notation

- Example notation for a rigid 2D transform:

$$\mathbf{x}' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$\mathbf{x}' = \mathbf{T}\left(\mathbf{x} \middle| \mathbf{p}\right) = \mathbf{T}\left(x, y \middle| t_x, t_y, \theta\right)$$

- Goal: find parameter values (i.e., $t_x$, $t_y$, θ) that optimize some image similarity metric.

# Optimizer

- Optimizer adjusts the transform in an attempt to improve the metric
- Often requires the derivative of the image similarity metric, $S$

Constant during registration!

$$\frac{\partial S(\mathbf{p}|F,M,\mathbf{T})}{\partial p_i} = \sum_{j \in \text{dimensions}} \frac{\partial S(\mathbf{p}|F,M,\mathbf{T})}{\partial x'_j} \frac{\partial x'_j}{\partial p_i}$$

$$\begin{array}{c}\text{Transform}\\ \text{Jacobian}\\ \text{(parameter}\\ \text{version)}\end{array} = J = \begin{bmatrix} \frac{\partial x'_1}{\partial p_1} & \cdots & \frac{\partial x'_1}{\partial p_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial x'_N}{\partial p_1} & \cdots & \frac{\partial x'_N}{\partial p_m} \end{bmatrix}$$

Spatial coordinates (output of transform)

## Understanding the Transform Jacobian

- $J$ shows how changing $p$ shifts a transformed point in the moving image space.
- This allows efficient use of a pre-computed moving-image gradient to infer changes in corresponding-pixel intensities for changes in $p$
- Now we can update $dS/d\mathbf{p}$ by just updating $J$

## Transforms

- Before we discuss specific transforms, let’s discuss the…
- *Fixed Set* = the set of points (i.e. physical coordinates) that are unchanged by the transform
- The fixed set is a very important property of a transform

## Identity Transform

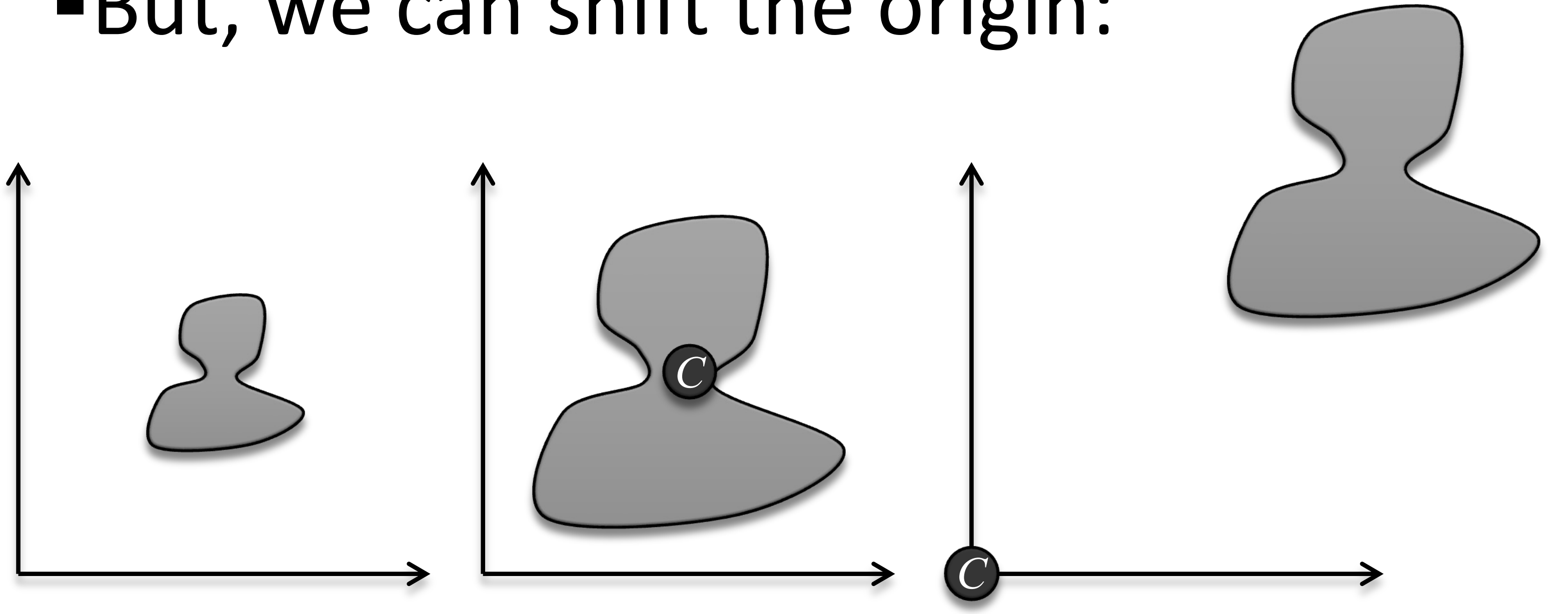
- Does “nothing”
- Every point is mapped to itself
- Fixed set = everything (i.e., the entire space)

## Translation Transform

- Fixed set = empty set
- Translation can be closely approximated by:
  - Small rotation about distant origin, and/or…
  - Small scale about distant origin
  - Both of these *do* have a fixed point
- Optimizers will frequently (accidently) do translation by using either rotation or scale
  - This makes the optimization space harder to use
  - The final transform may be harder to understand

# Scaling Transform

- Isotropic scaling (same in all directions)
- Anisotropic scaling
- Fixed set = origin = "center" = $C$
- But, we can shift the origin:



## Translation from Scaling

$$\mathbf{x}' = \begin{bmatrix} x_1' \\ \vdots \\ x_N' \end{bmatrix} = D\begin{bmatrix} x_1' - C_1 \\ \vdots \\ x_N' - C_N \end{bmatrix} + \begin{bmatrix} C_1 \\ \vdots \\ C_N \end{bmatrix}$$

$$\mathbf{x}' = D\begin{bmatrix} x_1' \\ \vdots \\ x_N' \end{bmatrix} + (1-D)\begin{bmatrix} C_1 \\ \vdots \\ C_N \end{bmatrix}$$

$$\mathbf{x}' = D\begin{bmatrix} x_1' \\ \vdots \\ x_N' \end{bmatrix} + \begin{bmatrix} T_1 \\ \vdots \\ T_N \end{bmatrix}$$

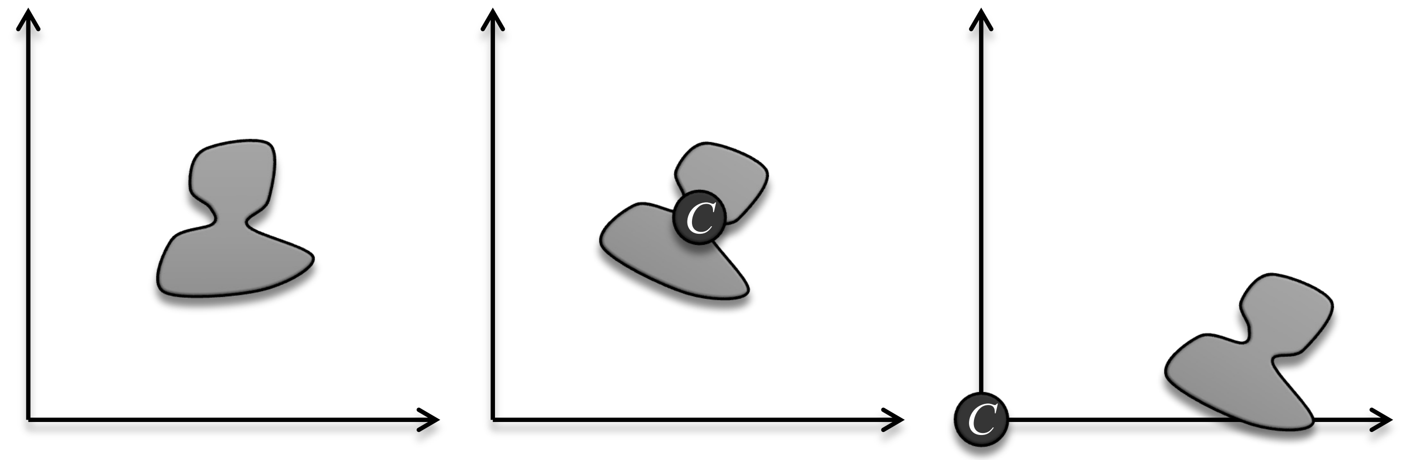
$$\therefore T_i = (1-D)C_i$$

- $D$ = Scaling Factor
- $C$ = Fixed Set

  i.e., shifted origin
- $T_i$ = Translation derived from scaling along dimension $i$ if using center $C$

# 2D Rotation Transform

- Rotation transforms are typically specific to either 2D or 3D
- Fixed set = origin = “center” = $C$



# Translation from 2D Rotation

$$\mathbf{x}' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x - C_x \\ y - C_y \end{bmatrix} + \begin{bmatrix} C_x \\ C_y \end{bmatrix}$$

$$\mathbf{x}' = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 1-\cos\theta & \sin\theta \\ -\sin\theta & 1-\cos\theta \end{bmatrix} \begin{bmatrix} C_x \\ C_y \end{bmatrix}$$

$$\mathbf{x}' = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \end{bmatrix}$$

$$\therefore \begin{aligned} T_x &= C_x\left(1-\cos\theta\right) + C_y \sin\theta \\ T_y &= -C_x \sin\theta + C_y\left(1-\cos\theta\right) \end{aligned}$$

- θ = Rotation angle
- $C$ = Fixed Set

  (Just one point)
- $T_i$ = Translation along dimension $i$ derived from rotation about center $C$

# Polar Coordinates: 2D Rotation = Multiplication

$$\mathbf{x}' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$(x, y) = re^{\mathbf{i}\phi} = (r\cos\phi, \mathbf{i}r\sin\phi)$$

$$(x', y') = re^{\mathbf{i}(\phi+\theta)} = re^{\mathbf{i}\phi}e^{\mathbf{i}\theta}$$

## Optimizing 2D Rotations

- Remember, optimization searches for the parameter values (i.e., θ) that give the best similarity score, $S$
- Ex: Gradient descent update step:

$$\theta' = \theta + \tfrac{\partial S}{\partial \theta} \lambda$$

$$e^{\mathbf{i}\theta'} = e^{\mathbf{i}\theta} e^{\mathbf{i}\frac{\partial S}{\partial \theta}\lambda} = e^{\mathbf{i}\theta} e^{\mathbf{G}\lambda}, \text{ where } \mathbf{G} = \mathbf{i}\,\tfrac{\partial S}{\partial \theta}$$

- The variation, $\mathbf{G}$, is the gradient of $S$
- Step length is λ

## Optimizing 2D Rotations with Scaling

- Transform is now multiplication by $De^{i\theta}$:
- Ex: Gradient descent update step:

$$\mathbf{G} = D\frac{\partial S}{\partial D} + \mathbf{i}\frac{\partial S}{\partial \theta}$$

$$D'e^{\mathbf{i}\theta'} = De^{\mathbf{i}\theta}e^{\mathbf{G}\lambda}$$

- Apply transform to point as:

$$\left(x', y'\right) = Dre^{\mathbf{i}(\phi+\theta)} = De^{i\theta} \cdot re^{\mathbf{i}\phi}$$

## Similarity Transform

- $P' = T(P) = (P\text{-}C)De^{\boldsymbol{i}\theta}+C$
- $P$ = arbitrary point
- $C$ = fixed point
- $D$ = scaling factor
  - Rigid transform if $D = 1$
- $\theta$ = rotation angle
- $P$ & $C$ are complex numbers: $(x+\boldsymbol{i}y)$ or $re^{\boldsymbol{i}\theta}$
- Store derivatives of $P$ in Jacobian matrix

## Affine Transform

- Only thing guaranteed preserved is collinearity
- $\mathbf{x}' = \mathbf{A}\,\mathbf{x} + \mathbf{T}$
- $\mathbf{A}$ is a complex matrix of coefficients
- Translation expressed as shifted fixed point:
  - $\mathbf{x}' = \mathbf{A}\,(\mathbf{x}-\mathbf{C}) + \mathbf{C}$
- $\mathbf{A}$ is optimized similar to scaling factor

# Quaternions: 3D Scaling & Rotation

- Quotient of two vectors:
  - $\mathcal{Q}$ = **A / B**
- Operator that produces second vector:
  - **A =** $\mathcal{Q}$ ★ **B**
- Composed of a *versor* (for rotation) and a *tensor* (for scaling)
  - $\mathcal{Q} = \mathbf{T} \diamond \mathbf{V}$
  - Requires a total of 4 numbers

## Tensors: Representing 3D Scaling

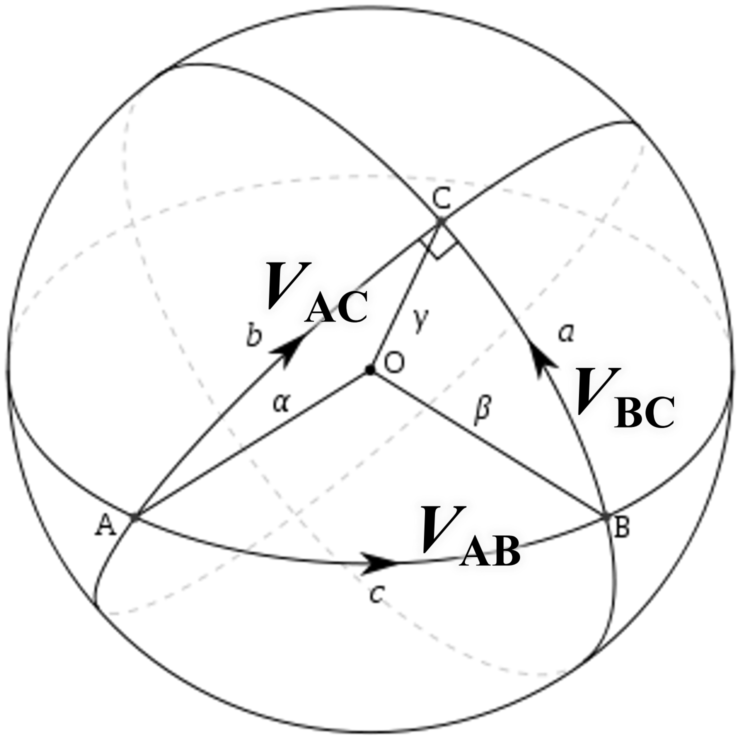
- Often denoted $\boldsymbol{T}$
- Tensors change the length of a vector
- For parallel vectors, tensors are scalars

## Versors: Representing 3D Rotations

- Often denoted $V$
- Problem: 3D Polar coordinates have a singularity at the poles
  - As do all 2-parameter 3D rotation representations
  - Longitude lines converge at the north and south pole
- Solution: Use 3 parameters!
- A versor is a vector pointing along the axis of rotation.
- The length of a versor gives the amount of rotation.

# Versors on Unit Spheres

- Arc $c$ is the versor $V_{\mathbf{AB}}$ that rotates the unit vector $\mathbf{A}$ to the unit vector $\mathbf{B}$
- $V_{\mathbf{AB}} = \mathbf{B} / \mathbf{A}$
- The versor can be repositioned anywhere on the sphere without changing it
- $V_{\mathbf{AC}} = V_{\mathbf{BC}} \diamond V_{\mathbf{AB}}$
  - NOT commutative



## Versor Addition

- Adding two versors is analogous to averaging them.
- **<u>Do NOT use versor addition with gradient descent</u>**
  - Use composition instead:
  - $\boldsymbol{V}_{t+1} = \mathrm{d}\boldsymbol{V}_{\mathrm{t}} \diamond \boldsymbol{V}_{\mathrm{t}}$

# Optimization of Versors

- Versor angle should be scaled using an exponent
- $\boldsymbol{V}^{w}$ will rotate by $w$ times as much as $\boldsymbol{V}$
  - $\Theta(\boldsymbol{V}^{w}) = w\theta$, where $\Theta(\boldsymbol{V})=\theta$
- Versor increment rule:

$$d\boldsymbol{V} = \left[\frac{\partial S(\boldsymbol{V})}{\partial \boldsymbol{V}}\right]^{\lambda}$$

# Rigid 3D Transform

- Use versor instead of phasors/polar coordinates
- $P' = \boldsymbol{V} \bigstar (P - C) + C$
- $P' = \boldsymbol{V} \bigstar P + T$, where $T = C - \boldsymbol{V} \bigstar C$
- $P$ = point, $T$ = translation, $C$ = fixed point, $\boldsymbol{V}$ = versor
- Represented by 6 parameters:
  - 3 for versor
  - 3 for shifted center

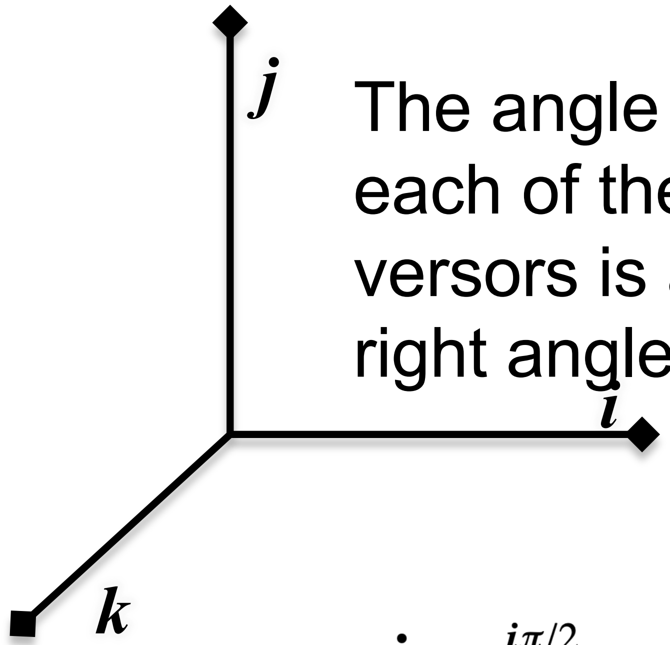
# Elementary Quaternions

- The 3 elementary quaternions are the 3 orthogonally-oriented right versors ($\boldsymbol{i},\boldsymbol{j},\boldsymbol{k}$):

  $-\boldsymbol{i} = \boldsymbol{k} \diamond \boldsymbol{j}$

  $-\boldsymbol{j} = \boldsymbol{i} \diamond \boldsymbol{k}$

  $-\boldsymbol{k} = \boldsymbol{j} \diamond \boldsymbol{i}$



The angle of each of these versors is a right angle.

$$\boldsymbol{i} = e^{\boldsymbol{i}\pi/2}$$

$$\boldsymbol{j} = e^{\boldsymbol{j}\pi/2}$$

$$\boldsymbol{k} = e^{\boldsymbol{k}\pi/2}$$

# Versors: Numerical Representation

- Any right versor $\boldsymbol{v}$ can be represented as:
  - $\boldsymbol{v} = x\boldsymbol{i} + y\boldsymbol{j} + z\boldsymbol{k}$, with $x^2 + y^2 + z^2 = 1$

- Any generic versor $V$ can be represented using the right versor $\boldsymbol{v}$ parallel to its axis of rotation, plus the rotation angle θ:
  - $V = e^{\boldsymbol{v}\theta}$
  - $V = cos\theta + \boldsymbol{v}\, sin\theta$
  - $V = cos\theta + (x\boldsymbol{i} + y\boldsymbol{j} + z\boldsymbol{k})\, sin\theta$, with $x^2 + y^2 + z^2 = 1$
  - $V = (cos\theta, x\, sin\theta, y\, sin\theta, z\, sin\theta)$ , with $x^2 + y^2 + z^2 = 1$

# Similarity 3D Transform

- Replace versor with quaternion to represent both rotation and scale
- $P' = \boldsymbol{Q} \star (P - C) + C$
- $P' = \boldsymbol{Q} \star P + T$, where $T = C - \boldsymbol{Q} \star C$
- $P$ = arbitrary point
- $C$ = fixed point
- $\boldsymbol{Q}$ = quaternion

An N-Dimensional Multi-Modal Registration Metric:

## Mutual Information

## Different Modalities

- Problem: In CT, a tumor may be darker than the surrounding liver tissue, but in MRI, the tumor may be brighter, while both modalities may have liver darker than other organs, but vasculature may be visible in CT but not in MRI, etc.
- Directly comparing pixel values is hard
  - Sometimes bright maps to bright
  - Sometimes bright maps to dark
  - Sometimes both bright & dark map to just dark, etc.
- Old, “bad” solutions:
  - Try to simulate CT pixel values from MRI data, etc.
    - But if we could do this, then we wouldn’t need both modalities!
  - Try to segment first, register second

## Solution

- For each registration attempt, optimize the “niceness” of the resulting joint probability distributions for mapping pixel values from one modality to the other
- How?
- Maximize the mutual information between the two images

## Mutual Information

- Based on information theory
- Idea:  If both modalities scan the same underlying anatomy, then there should be redundant (i.e., *mutual*) information between them.
  - If bones are visible, then they should overlap
  - Image edges should mostly overlap
  - In general, each image intensity value in one image should map to only a few image intensities in the other image.

## Mutual Information

- Our similarity metric should now maximize:
  - The mutual information between the images, =
  - The information that each image provides about the other
- Assumption: Mutual information will be at a maximum when images are correctly aligned
- Note: We do NOT need a model of this mapping before registration—the mapping is learned as it is optimized

## Mutual Information, Conceptually

- Look at the joint histogram of pixel intensities
  - For every pair of pixels, one mapped onto the other, use their pixel intensities to look up the appropriate bin of the 2D histogram, and then increment that bin.
  - We want this joint histogram to be tightly clustered, i.e. “peaky”
  - Bad registrations will make the joint histogram look like a diffuse cloud

# Joint Histogram Examples

**Good Registration →**

**Tightly Clustered Joint Histogram**

| CT Values / MR Values | Vessels( darkest) | Tumor( dark) | Liver | Other(b right) |
|---|---|---|---|---|
| Liver & Vessels (dark) | ✓ | | ✓ | |
| Other(b right) | | | | ✓ |
| Tumor (Brightest) | | ✓ | | |

**Bad Registration →**

**Diffuse/Blurred Joint Histogram**

| CT Values / MR Values | Vessels( darkest) | Tumor( dark) | Liver | Other(b right) |
|---|---|---|---|---|
| Liver & Vessels (dark) | ✓ | ✓ | ✓ | ✓ |
| Other(b right) | ✓ | ✓ | ✓ | ✓ |
| Tumor (Brightest) | ✓ | ✓ | ✓ | ✓ |

## Mutual Information: Details

- Calculated by measuring entropies
  - M.I. = difference between joint entropy and the sum of individual entropies
- The math encourages transformations that make the images overlap on complex parts, something most similarity measures don't do
- In effect, M.I seeks a transform that finds complexity and explains it well.

## Mutual Information:

- Is robust with respect to occlusion—degrades gracefully
- Is less sensitive to noise and outliers
- Has an improved version, Mattes, which uses math that results in a smoother optimization space.
- Is now the de-facto similarity metric for multi-modal registration.