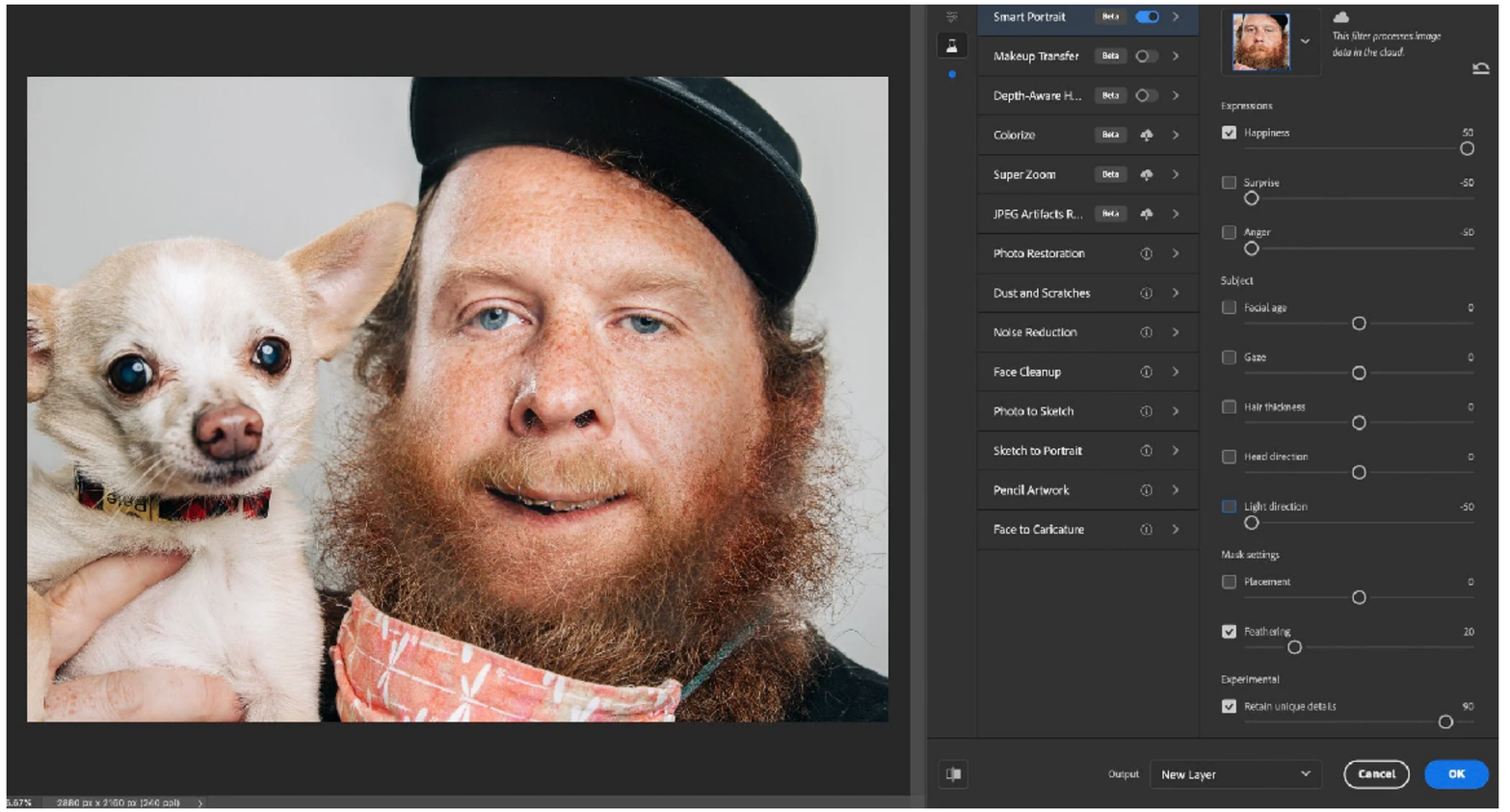
COMP0026: Image Processing

Image Transformations



[Popular Science]

Lectures will be Recorded

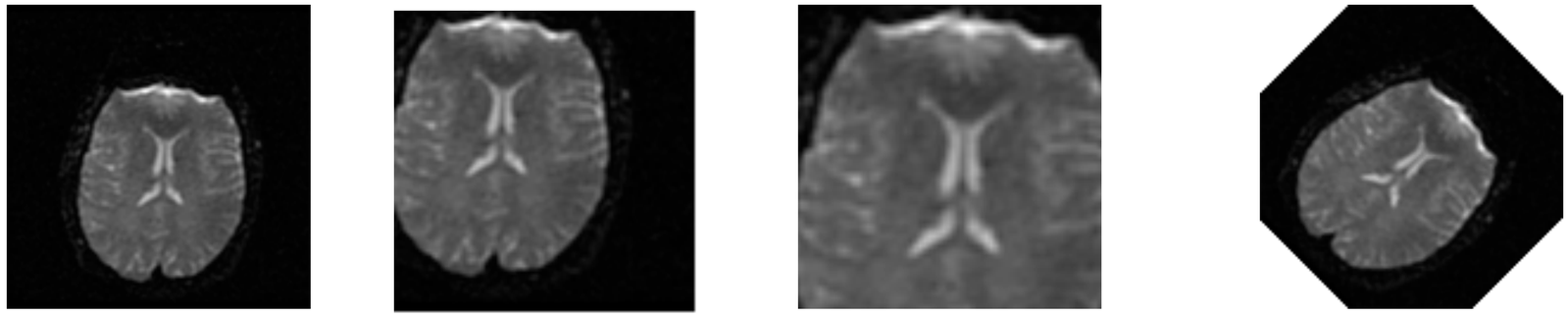
Outline

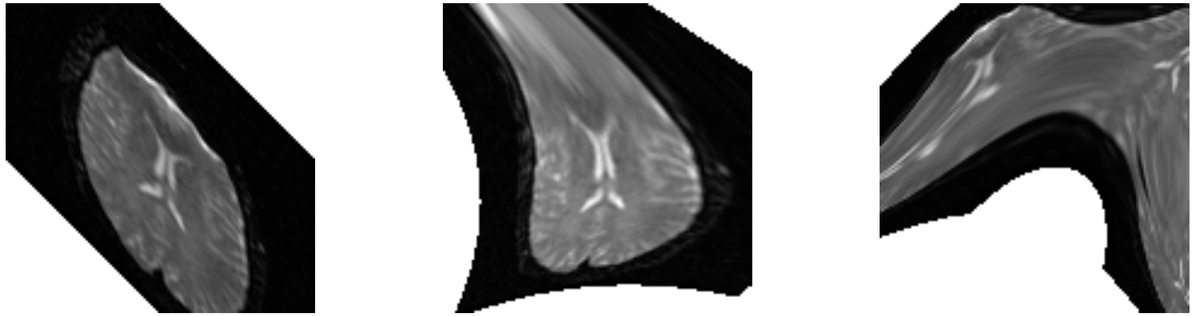
- Grey-level transformations
  - Histogram equalization

- Geometric transformations
  - Affine transformations
  - Interpolation
  - Warping and morphing

Geometric Transformations

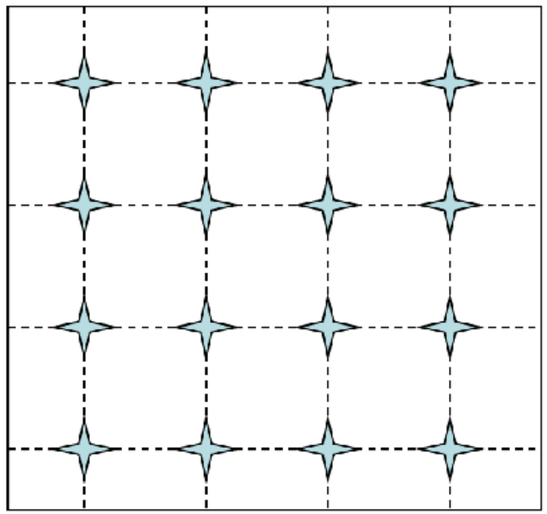
Change the location of image features

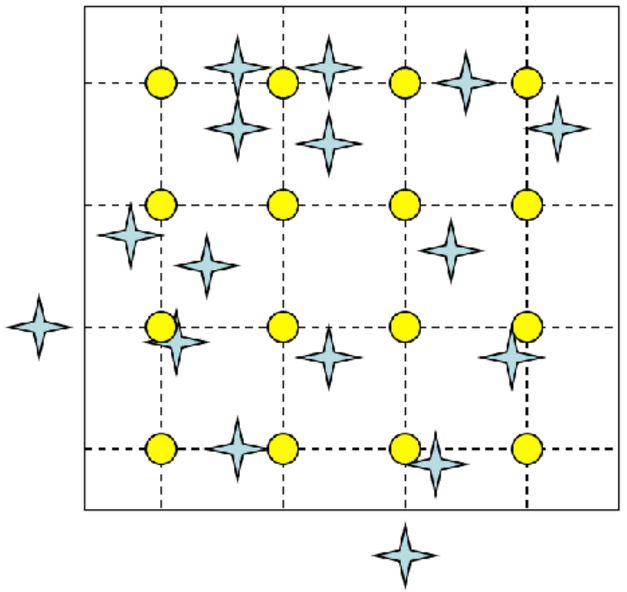




Geometric Transformations

- Mapping image f(u, v) to g(x, y) based on a given mapping function: x(u, v), y(u, v).
- Forward Mapping
  - For each point (u, v) in the original image, find the corresponding position (x, y) in the deformed image by the forward mapping function, and let g(x,y)=f(u,v).
  - What if the mapped position (x,y) is not an integer sample in the desired image?





Warping points are often non-integer samples

Many integer samples “o” are not assigned Values

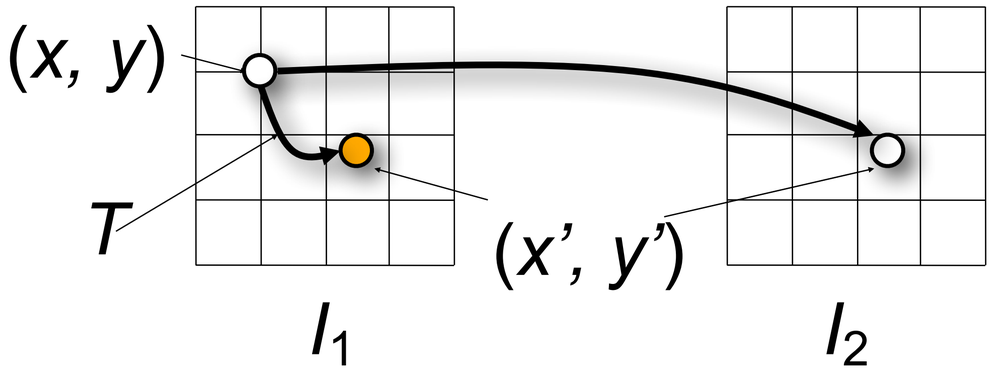
Geometric Transformations

- I_1 is the original image, I_2 is the warped image.

$$I_2(x', y') = I_1(x, y)$$

$$(x', y') \leftarrow T(x, y)$$

$$T : \mathbb{R}^2 \rightarrow \mathbb{R}^2$$



Affine Transformations

translation, scaling, rotation, shear

$$x' = ax + by + t_x$$

$$y' = cx + dy + t_y$$

Affine Transformations

translation, scaling, rotation, shear

$$x' = ax + by + t_x$$

$$y' = cx + dy + t_y$$

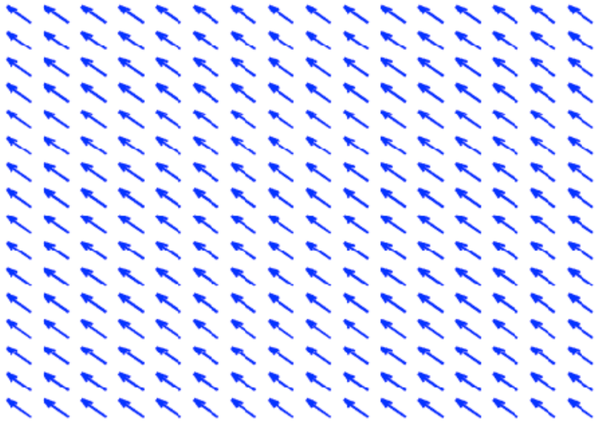
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

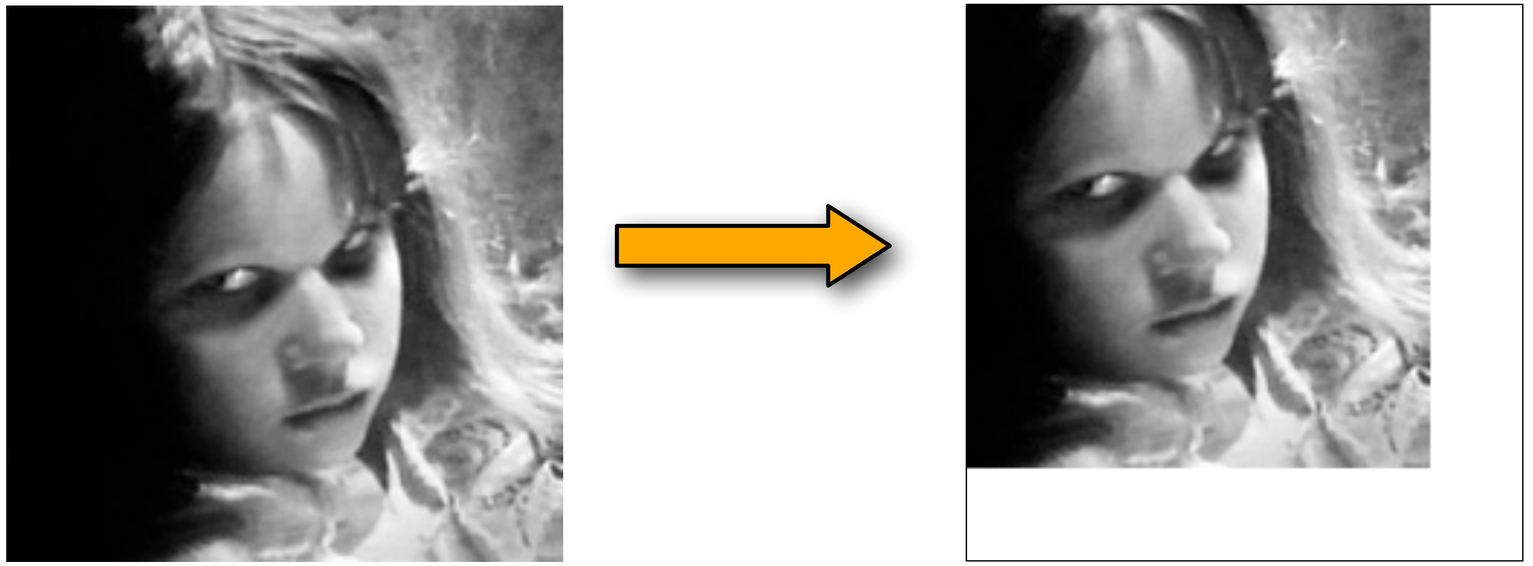
identity matrix (rotation + scaling + shear?)

Translation

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

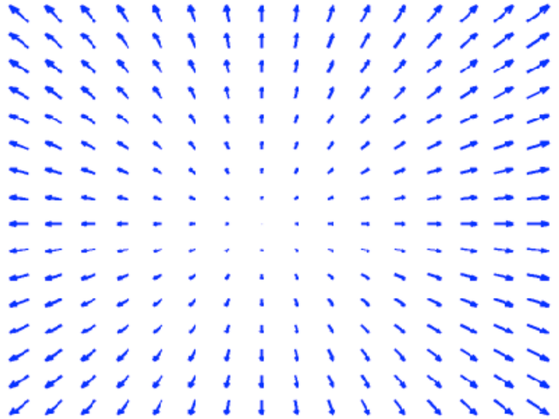




Scaling

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

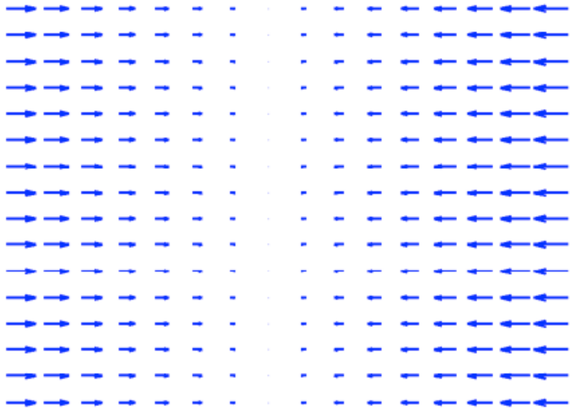
$$\begin{pmatrix} t_x \\ t_y \end{pmatrix} = \begin{pmatrix} -w/2 \\ -h/2 \end{pmatrix}$$

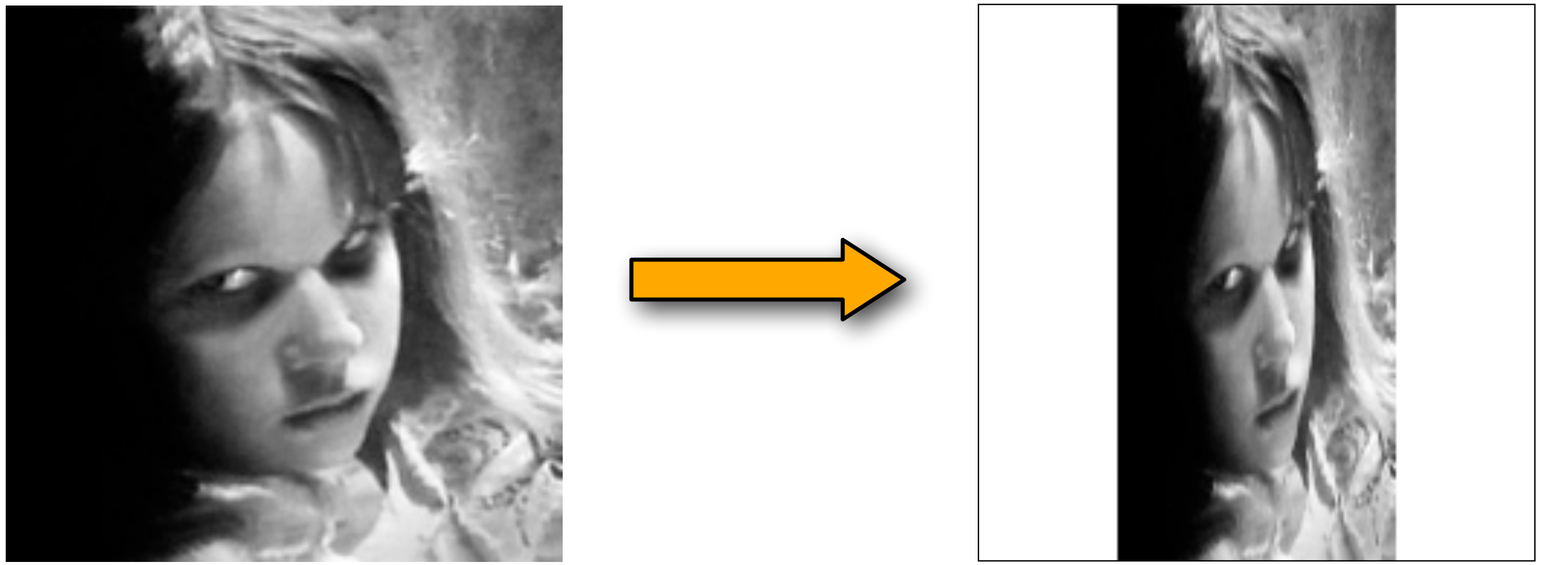


Stretch

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 0.5 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} t_x \\ t_y \end{pmatrix} = \begin{pmatrix} -w/4 \\ 0 \end{pmatrix}$$

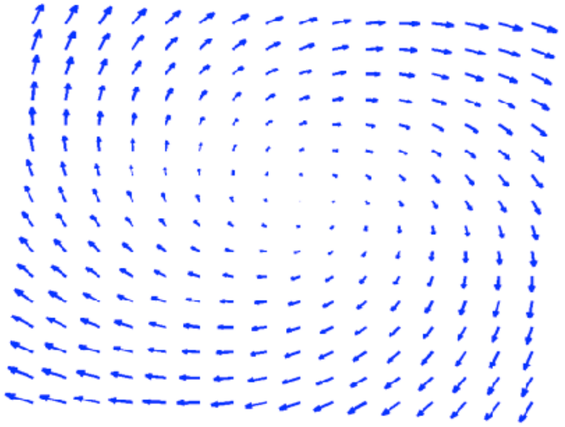




Rotation

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \quad \theta = \pi/4$$

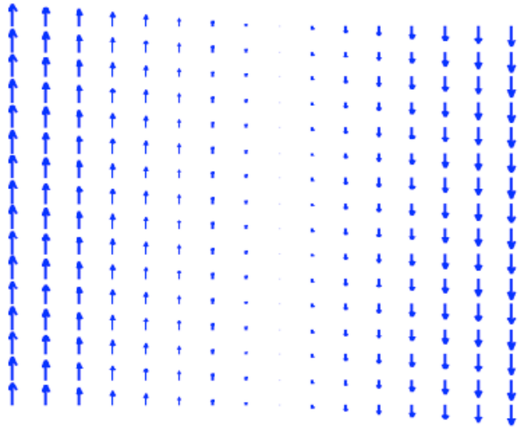
$$\begin{pmatrix} t_x \\ t_y \end{pmatrix} = \begin{pmatrix} -(W(\cos\theta - 1) + H\sin\theta)/2 \\ (W\sin\theta - H(\cos\theta - 1)/2) \end{pmatrix}$$



Shear (along y axis)

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ s & 1 \end{pmatrix}$$

$$\begin{pmatrix} t_x \\ t_y \end{pmatrix} = \begin{pmatrix} 0 \\ -sH/2 \end{pmatrix}$$



Homogeneous Coordinates

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Remember:

$$x' = ax + by + t_x$$
$$y' = cx + dy + t_y$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

Homogeneous Coordinates

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

“Subsequent” operations are inserted here,
by pre-multiplying

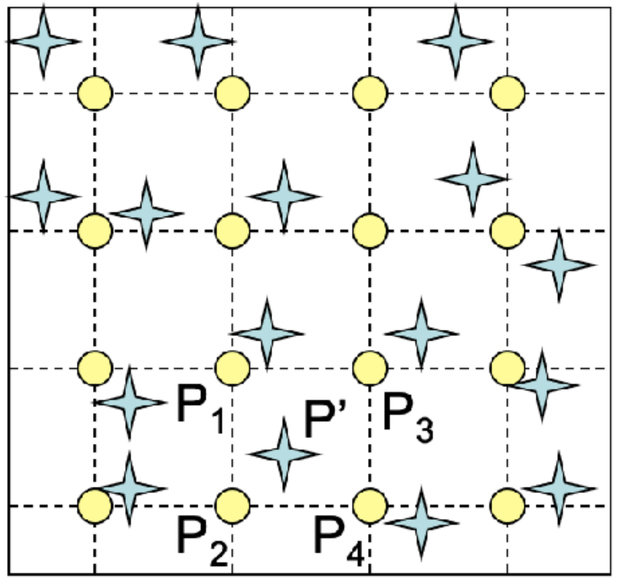
Implementation

Always use the inverse transformation:

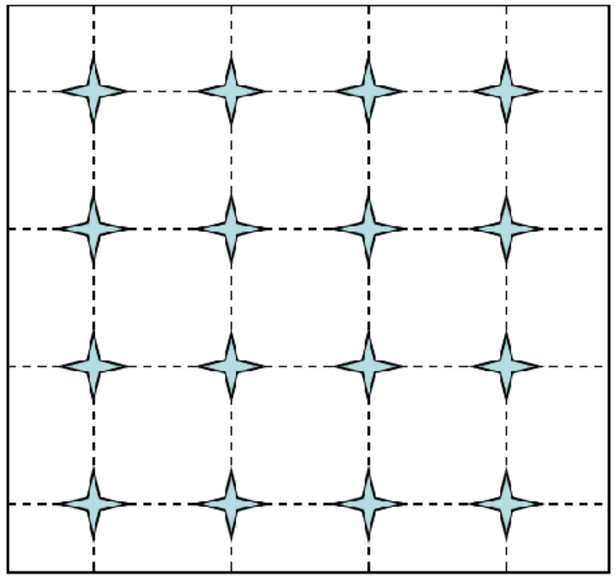
```
function warpedImage = transform(image, T)
    [W,H] = size(image);
    warpedImage = zeros(W,H);
    invT = invert(T);
    for x=1:W
        for y=1:H
            warpedImage(x,y)=image(invT(x,y));
        end
    end
```

Interpolation

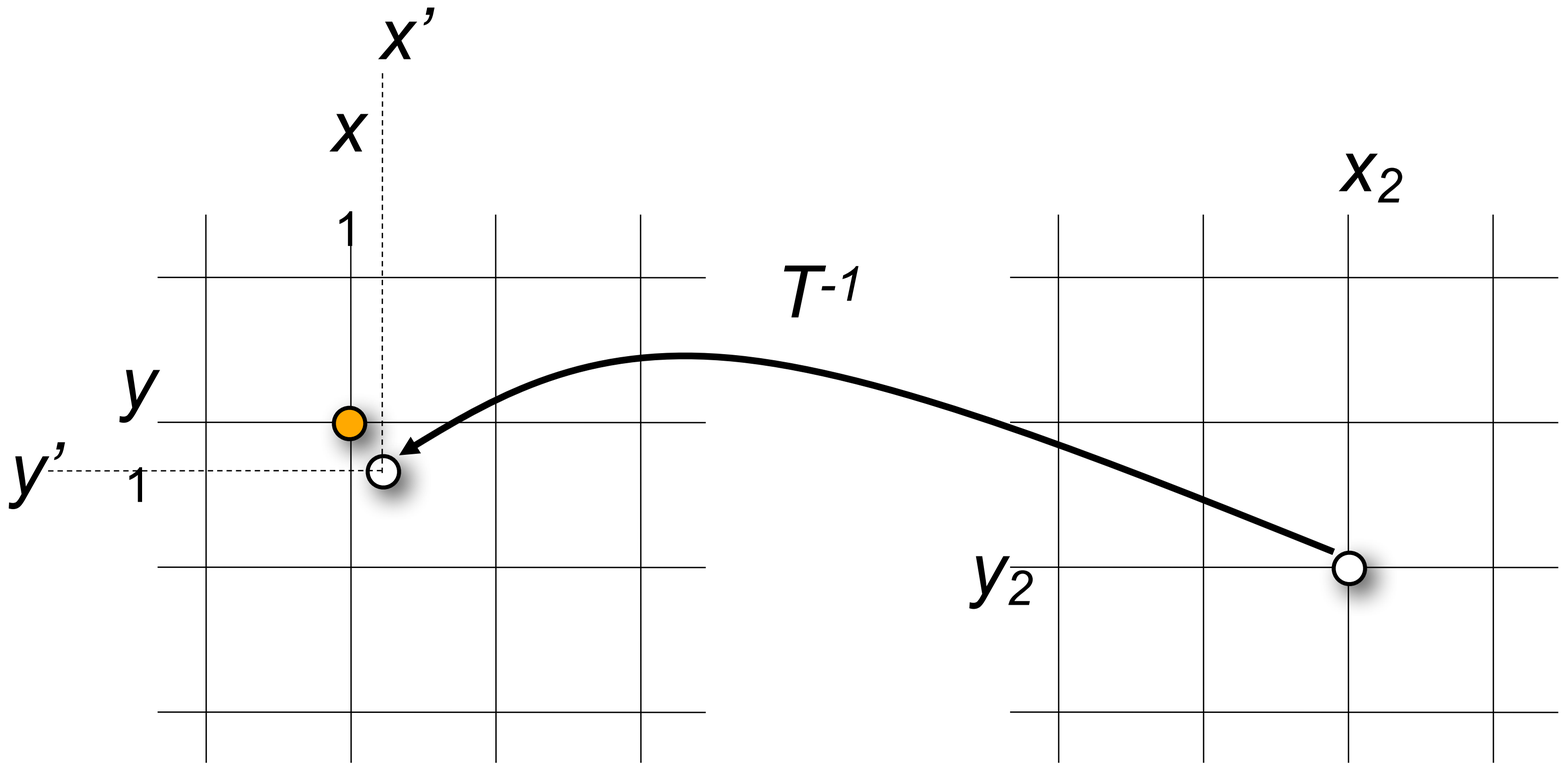
- **Usually,** $(x', y') \leftarrow T^{-1}(x, y)$ **are not integer coordinates.**
- **We estimate** $I_1(x', y')$ **by** *interpolation* **from surrounding positions.**



P' will be interpolated from P_1, P_2, P_3, and P_4



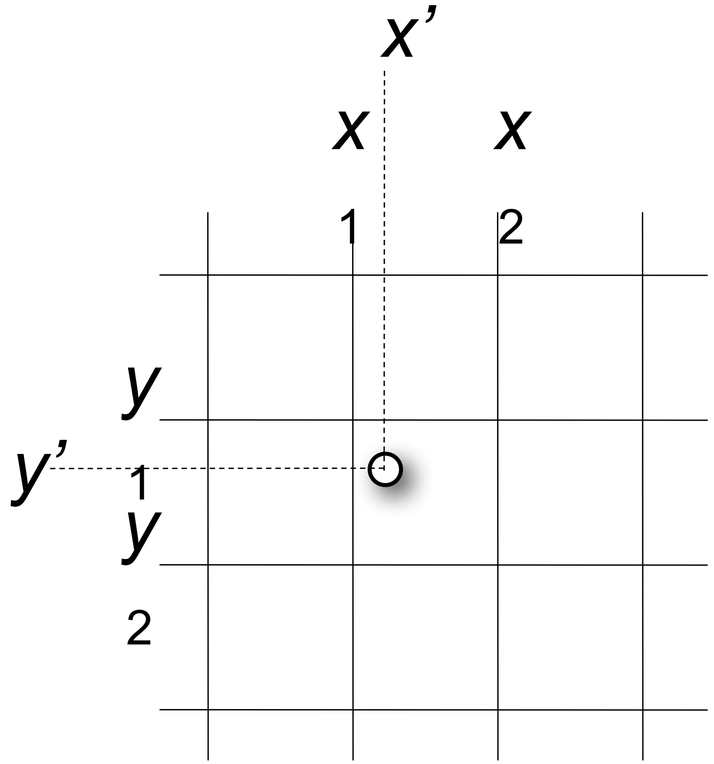
Nearest Neighbour Interpolation



Take the value at the nearest grid point:

$$I_1(x', y') = I_2(x_2, y_2)$$

Bilinear interpolation

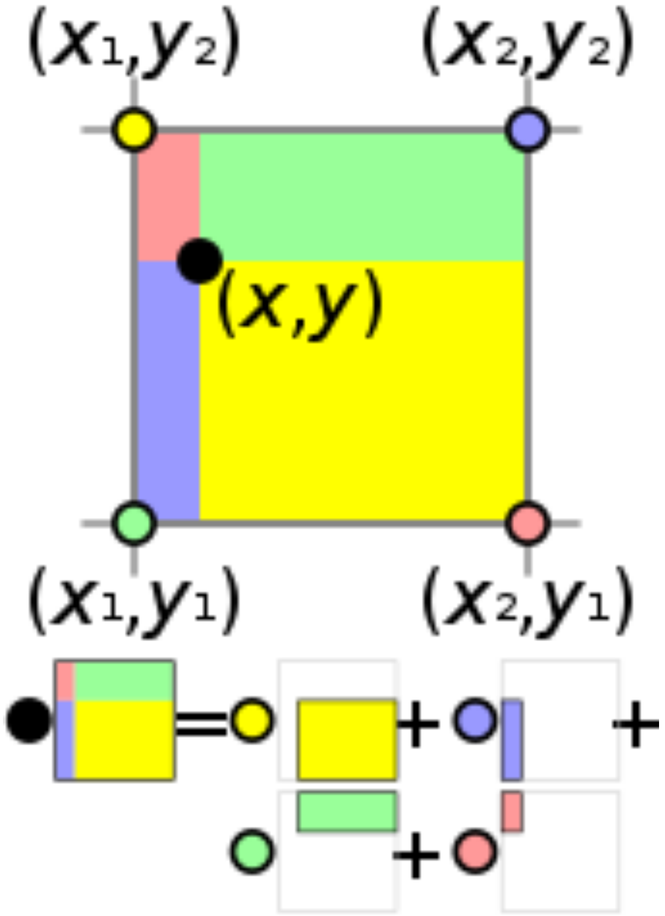


Weighted average from neighboring grid points:

$$I_1(x', y') = \Delta x\, \Delta y\, I_1(x_2, y_2)$$
$$+ \Delta x\, (1 - \Delta y)\, I_1(x_2, y_1)$$
$$+ (1 - \Delta x)\, \Delta y\, I_1(x_1, y_2)$$
$$+ (1 - \Delta x)\, (1 - \Delta y)\, I_1(x_1, y_1)$$

$$\Delta x = x' - x_1, \qquad \Delta y = y' - y_1$$

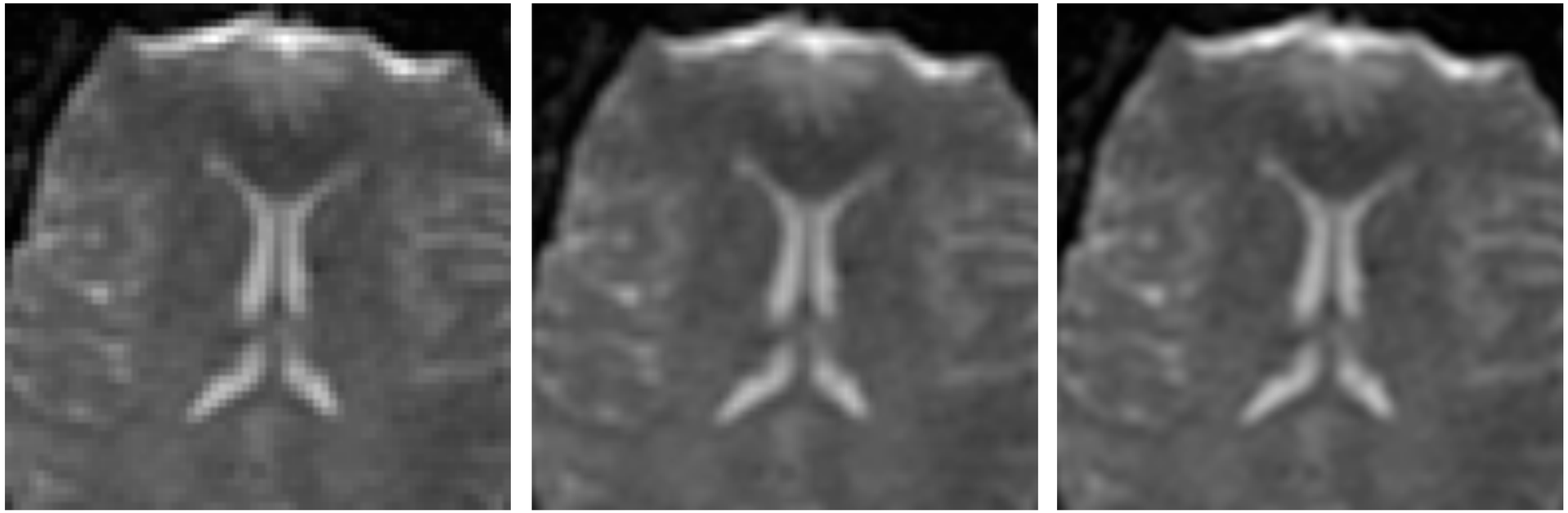
Bilinear interpolation



Higher Order Interpolation

- Quadratic interpolation fits a bi-quadratic function to a 3x3 neighbourhood of grid points
- Cubic interpolation fits a bi-cubic function to a 4x4 neighbourhood.

Interpolation Comparison



Nearest neighbour

Bilinear

Cubic

Polynomial Warps

- **Polynomial transformations, e.g., quadratic transformations:**

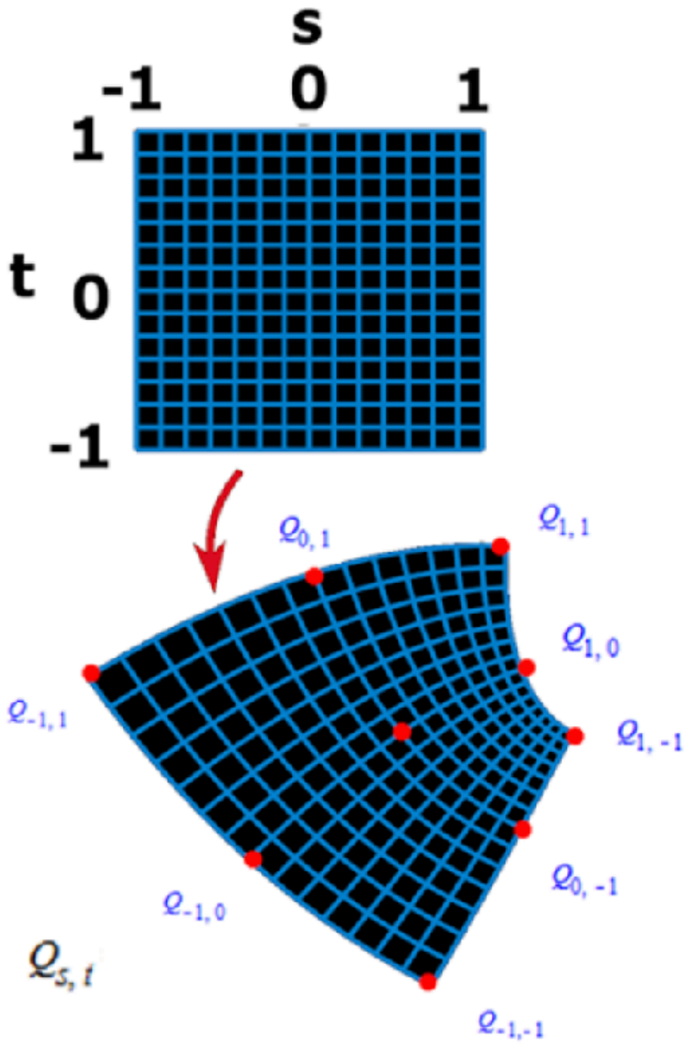
$$x' = a_0 + a_1 x + a_2 y + a_3 x^2 + a_4 xy + a_5 y^2$$

$$y' = b_0 + b_1 x + b_2 y + b_3 x^2 + b_4 xy + b_5 y^2$$

- **Affine transformations map lines to lines.**
- **They are *first-order* polynomial transformations.**
- **Higher-order polynomials can bend lines.**

Quadratic Warp

Quadratic Warp



Control Point Warps

- **Moves some pixels to specified locations.**
- **Interpolate the displacement at intermediate positions.**
- **Find a polynomial warp *P* that maps:**

$$(x_1, y_1) \rightarrow (x'_1, y'_1)$$
$$(x_2, y_2) \rightarrow (x'_2, y'_2)$$
$$\vdots$$
$$(x_m, y_m) \rightarrow (x'_m, y'_m)$$

Control point warps

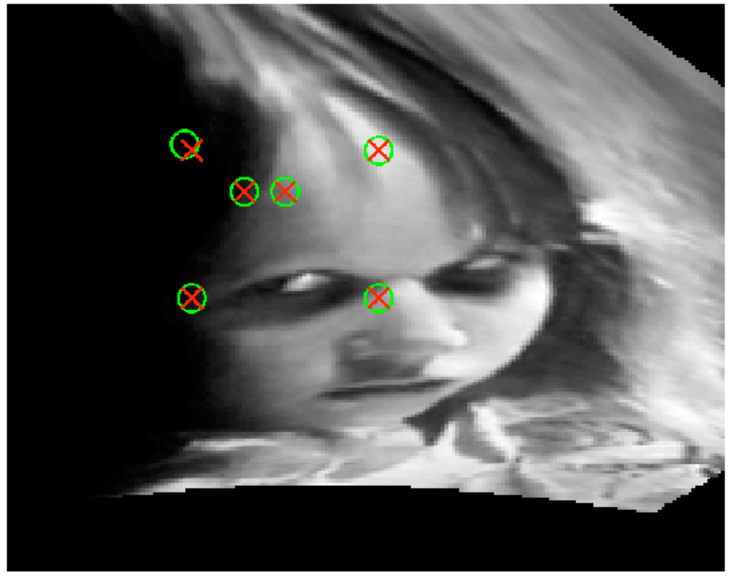
Least squares estimate of *P* is

$$\begin{pmatrix} x'_1 & y'_1 \\ x'_2 & y'_2 \\ \vdots & \vdots \\ x'_m & y'_m \end{pmatrix} = \begin{pmatrix} 1 & x_1 & y_1 & x_1^2 & x_1 y_1 & y_1^2 \\ 1 & x_2 & y_2 & x_2^2 & x_2 y_2 & y_2^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_m & y_m & x_m^2 & x_m y_m & y_1^2 \end{pmatrix} \begin{pmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \\ a_4 & b_4 \\ a_5 & b_5 \end{pmatrix}$$

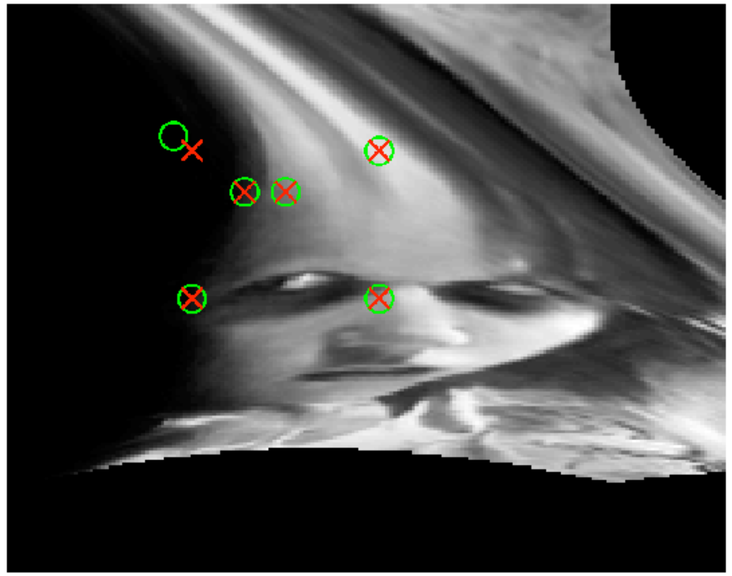
$$A = XP \qquad (m \geq 6)$$

$$P = (X^T X)^{-1} X^T A$$

Example – Two Pixel Displacement



Five Pixel Displacement



Over-determined

Applications

- Special effects
  - Film industry
  - Computer games

- Image registration
  - Medical imaging
  - Security

Image Morphing

Image Morphing

Image Morphing

Image Morphing

Image Morphing

Image Morphing

Image Morphing

Image Morphing

Image Morphing

Image Morphing

Image Morphing

Image Morphing

Image Morphing

Image Morphing

Image Morphing

Image Morphing

Image Morphing

Image Morphing

Image Morphing

Image Morphing

Image Morphing

Image Morphing

How do we get the *i*-th image in the sequence?

Landmarks

Image Registration

- Determines the transformation that aligns two similar images

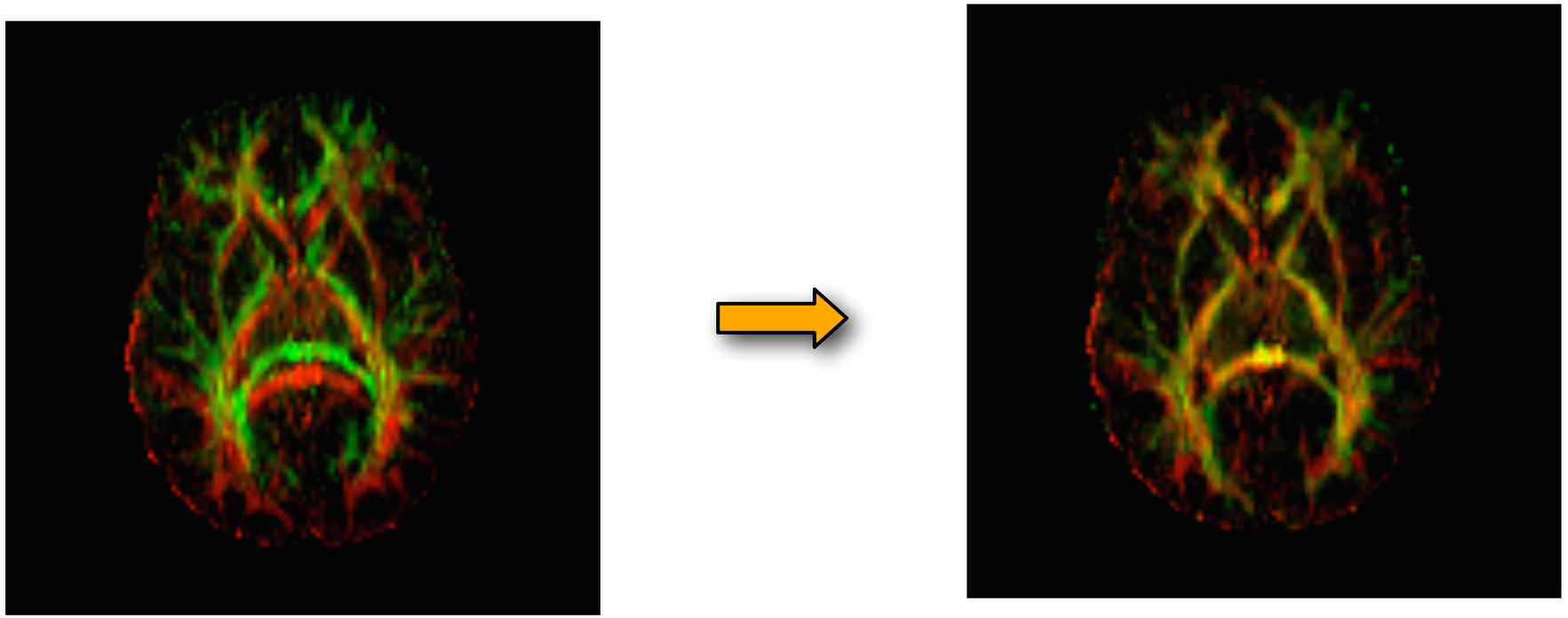


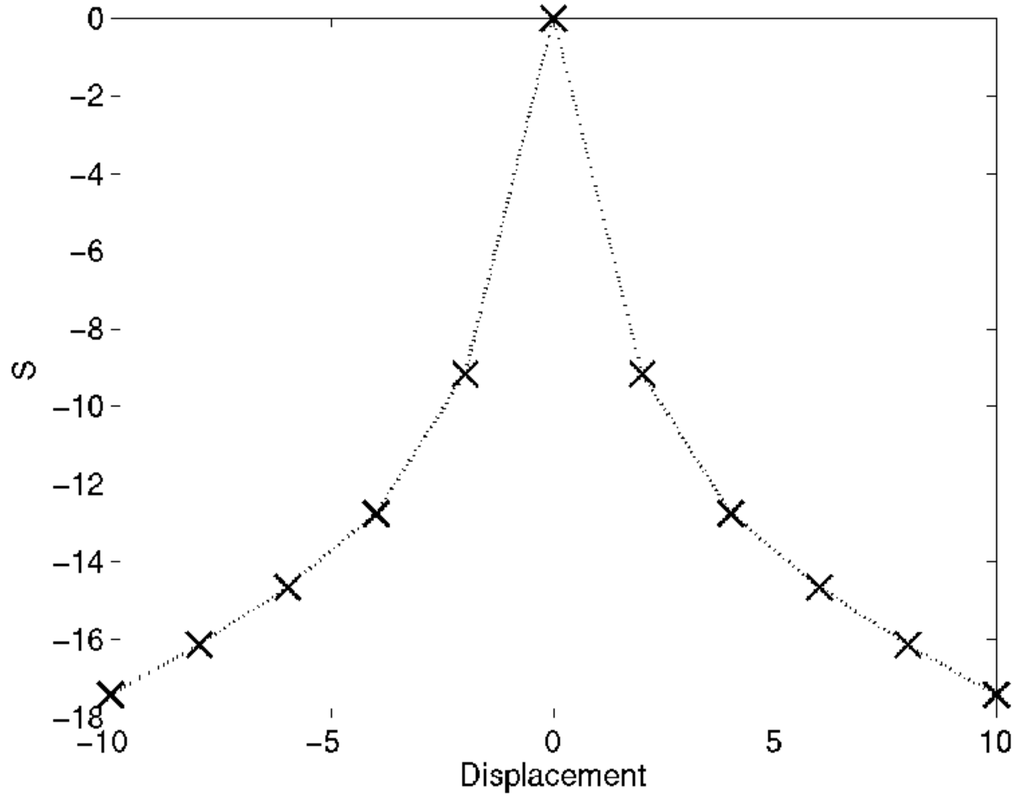
Image Similarity

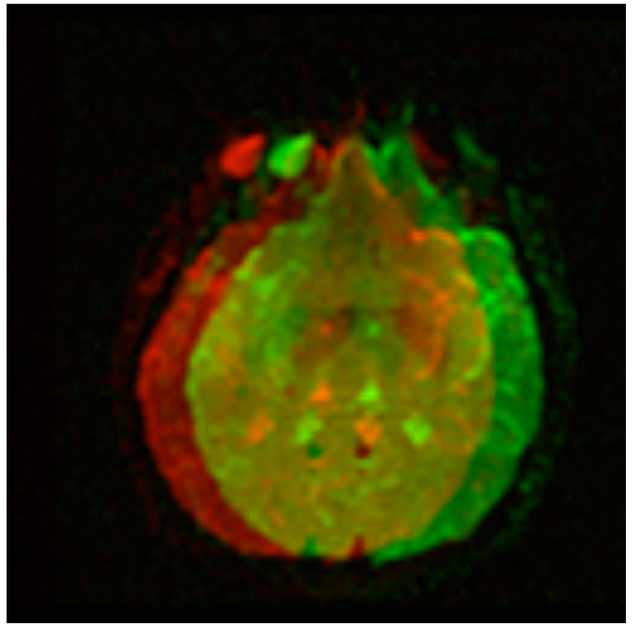
- **We find the transformation that maximises the similarity of the images.**
- **A simple similarity measure is:**

$$S(I_1, I_2) = -\left(\sum_{\mathbf{x} \in I_1} (I_1(\mathbf{x}) - I_2(\mathbf{x}))^2 \right)^{1/2}$$

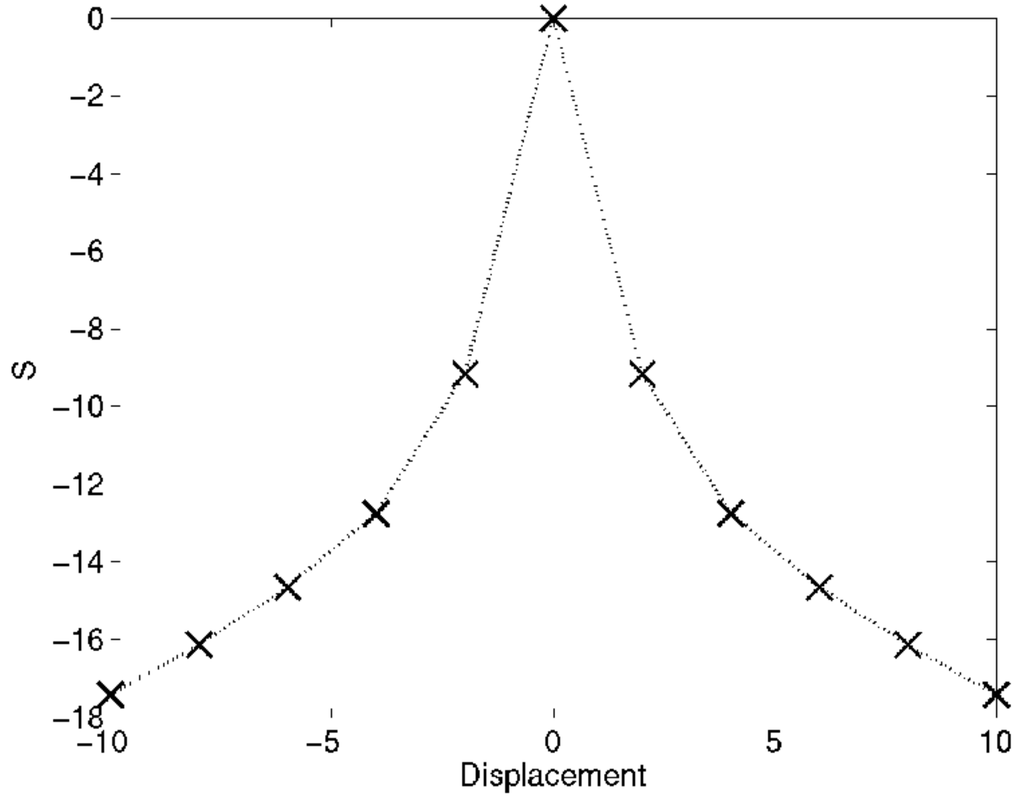
- **Many others are used including the *cross-correlation* and *mutual information*.**

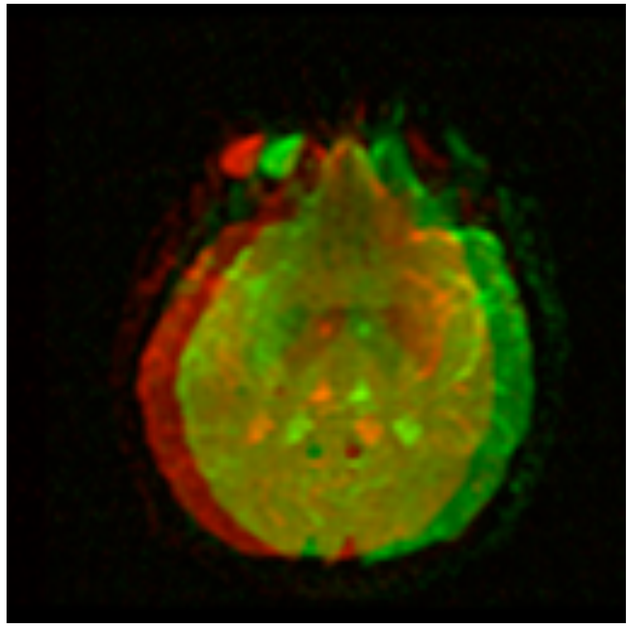
Registration Example



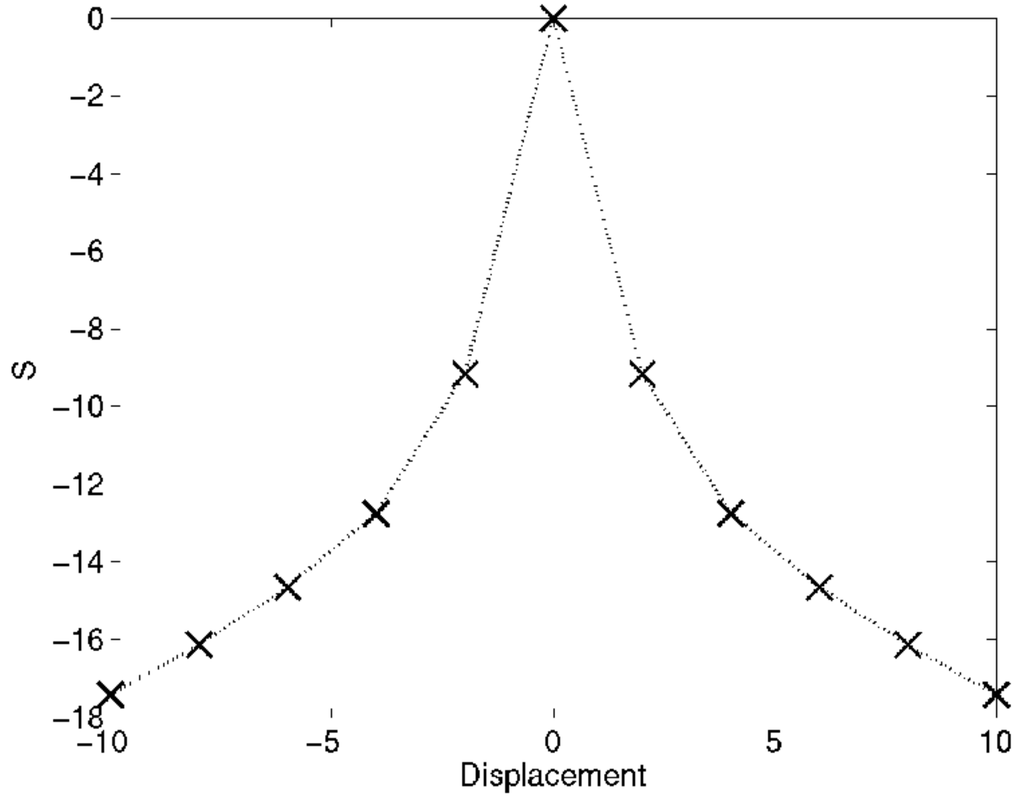


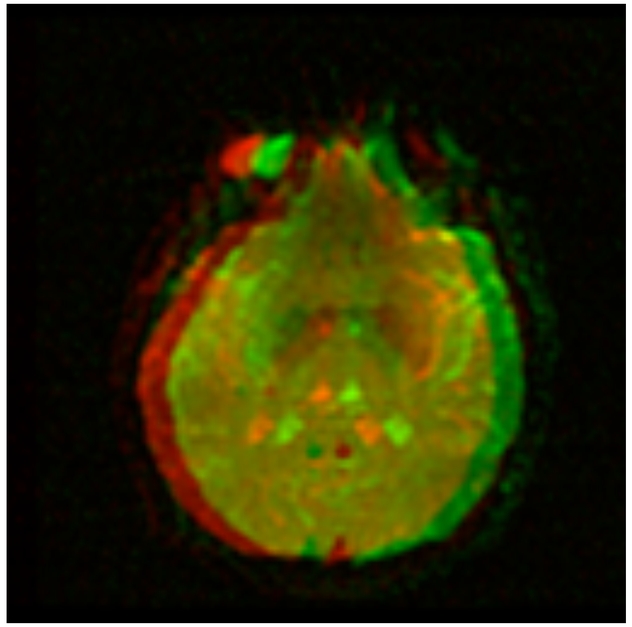
Registration Example



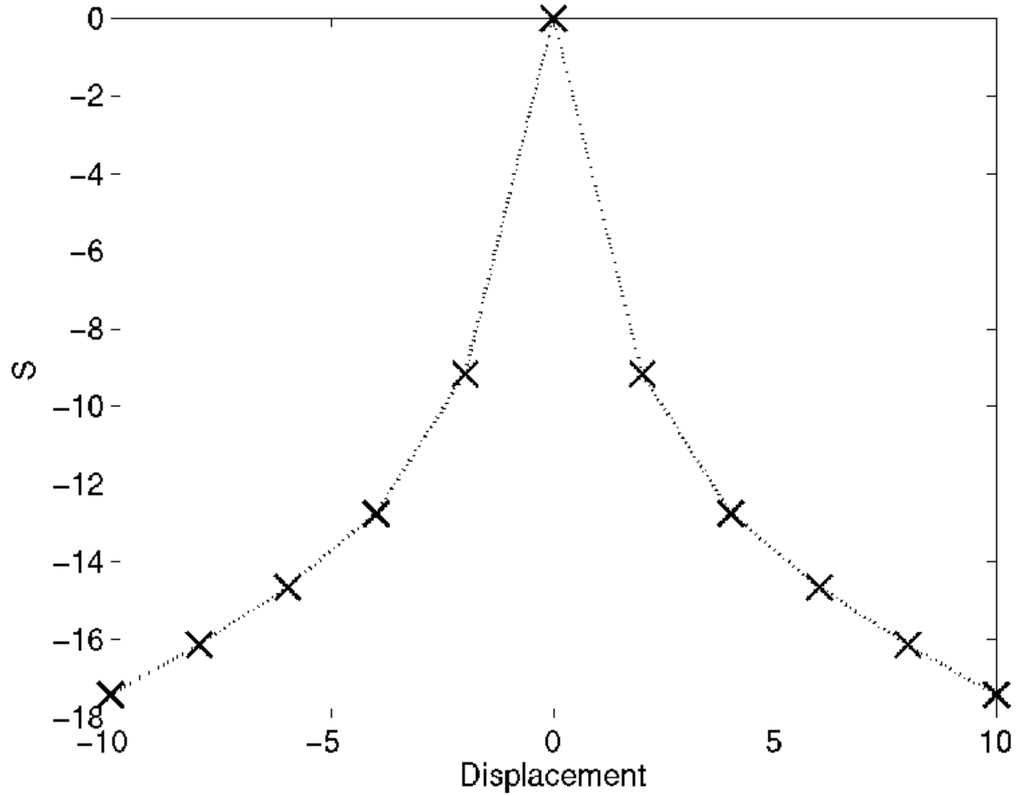


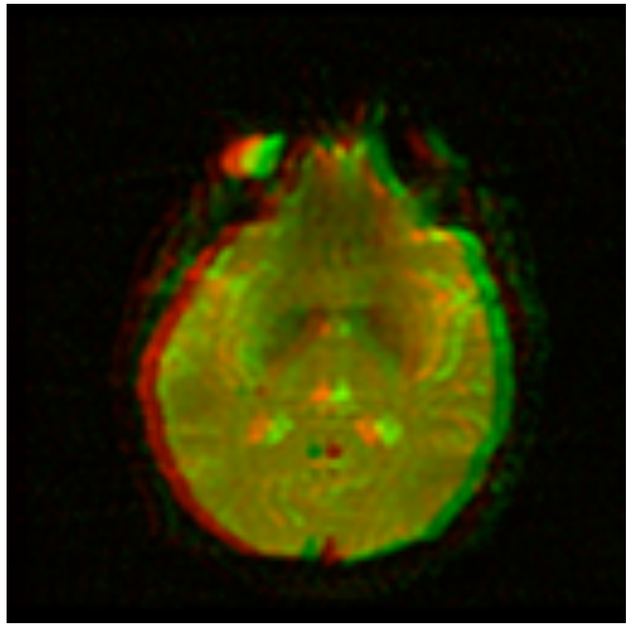
Registration Example



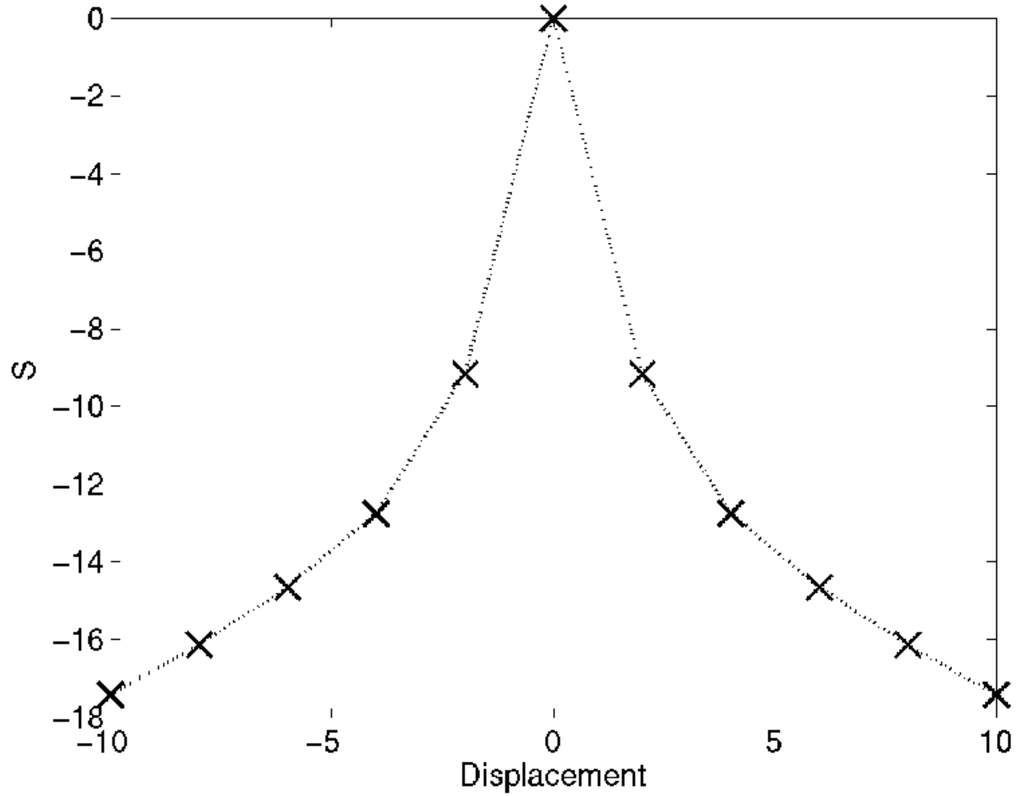


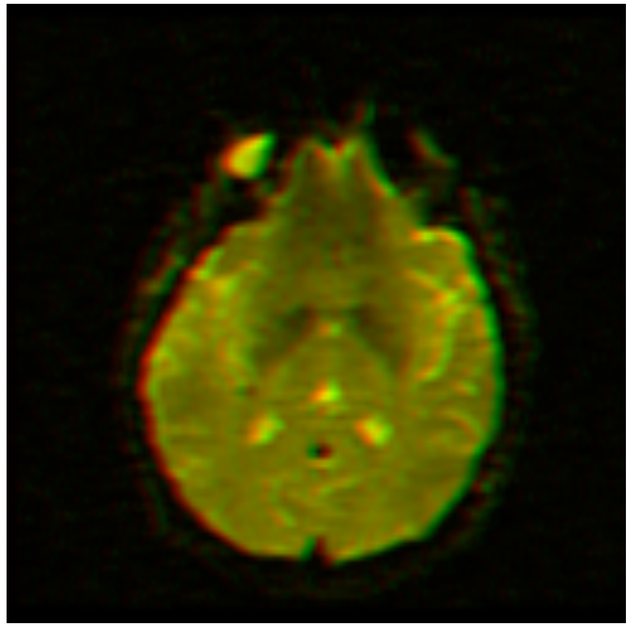
Registration Example



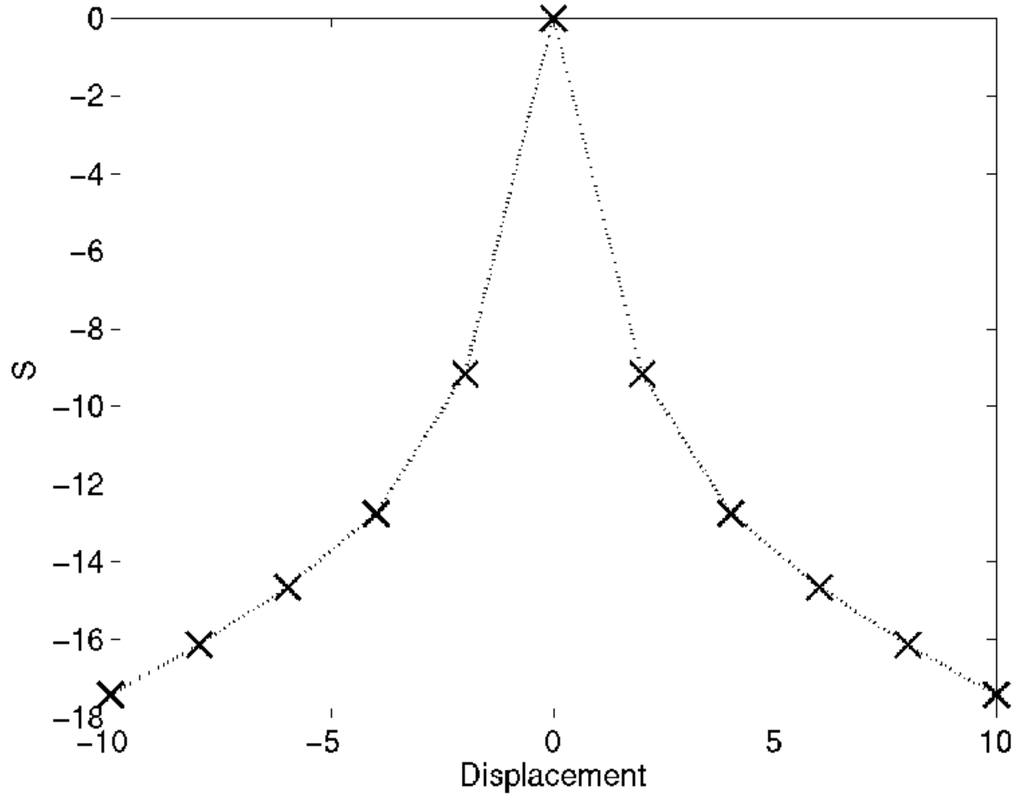


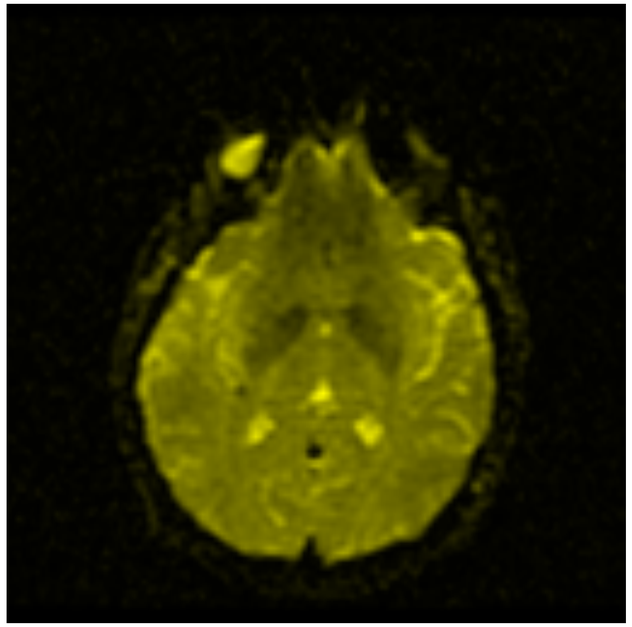
Registration Example



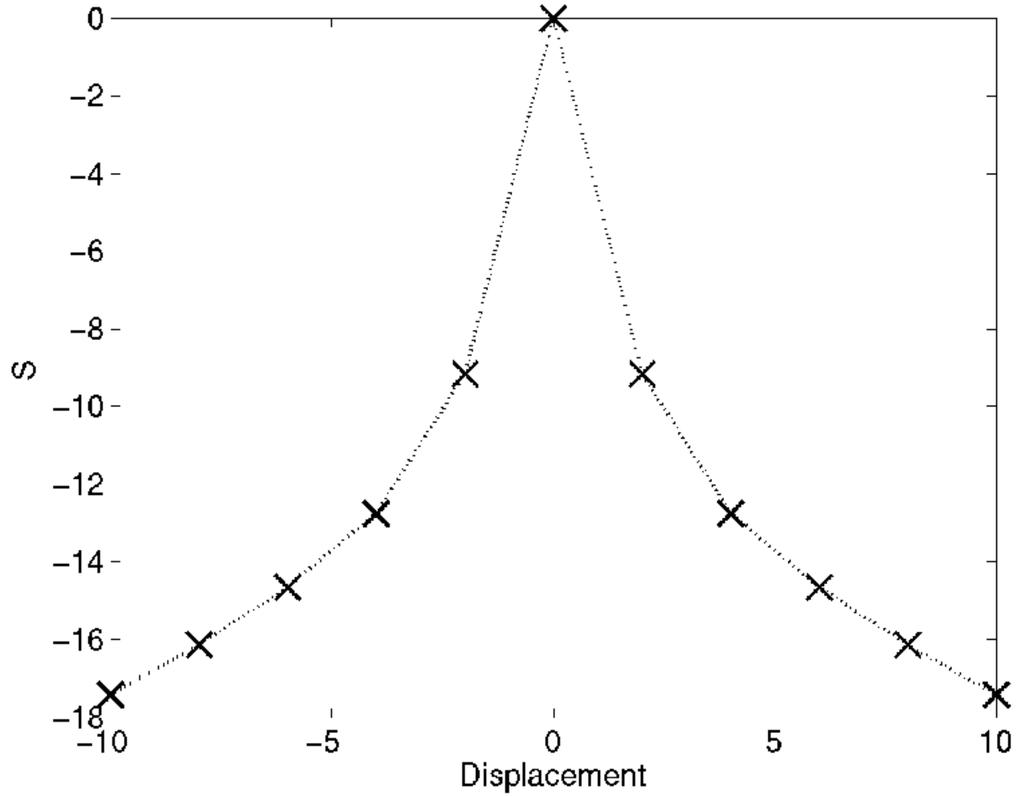


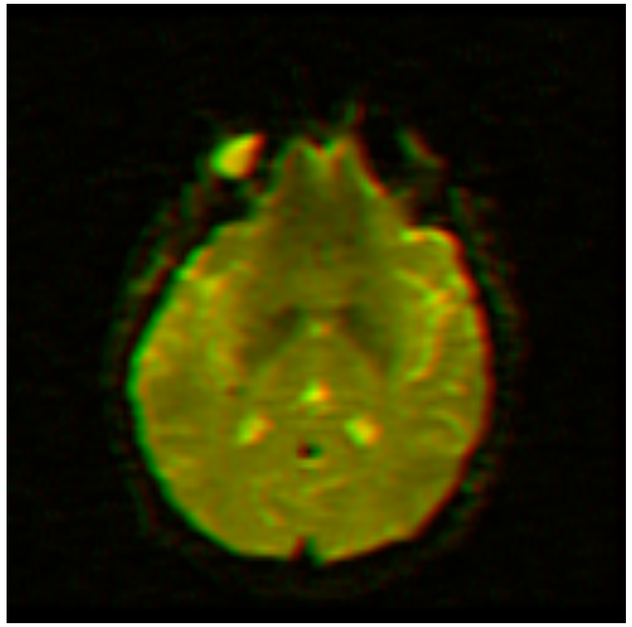
Registration Example



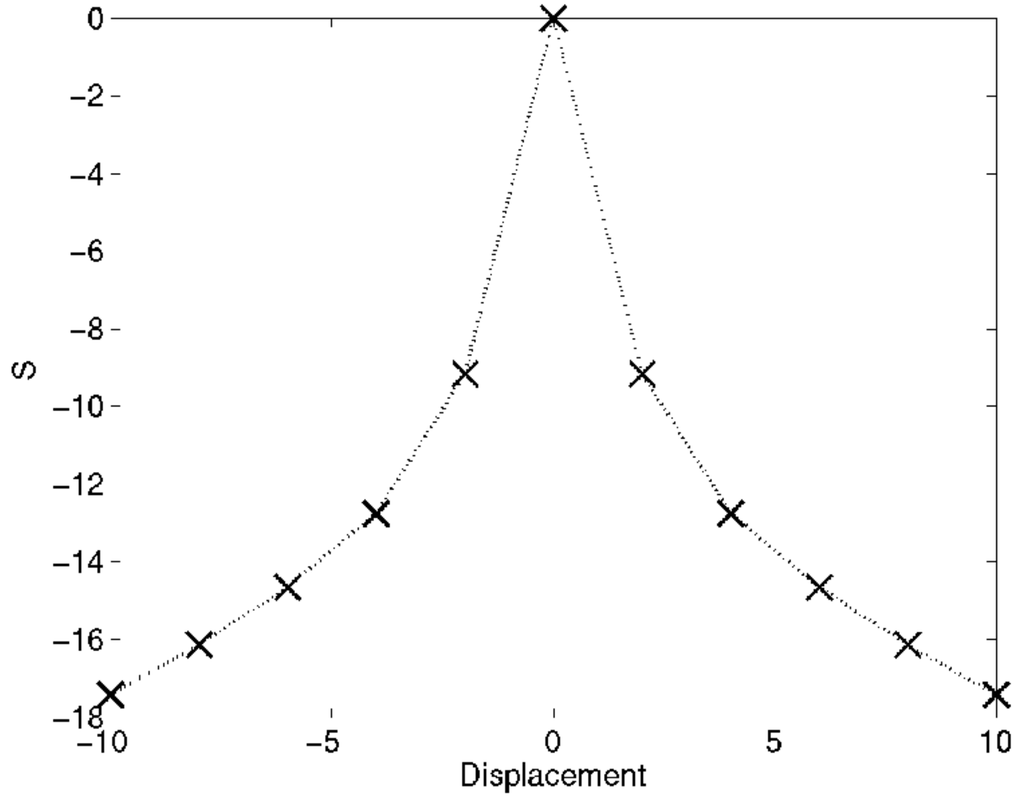


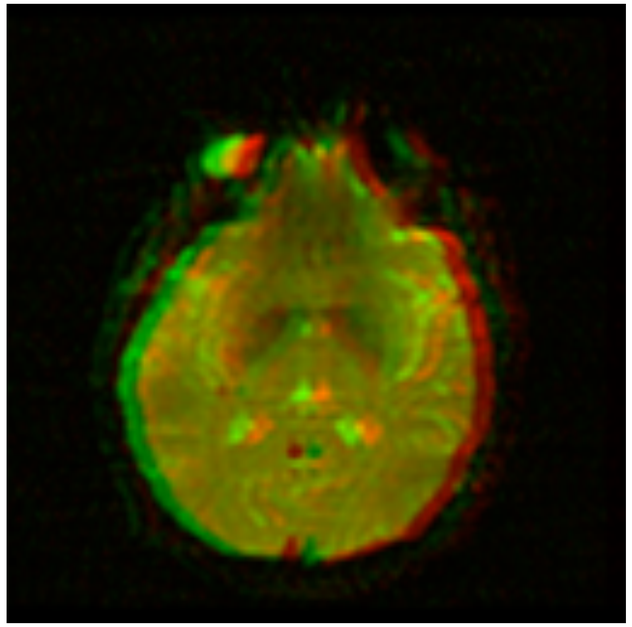
Registration Example



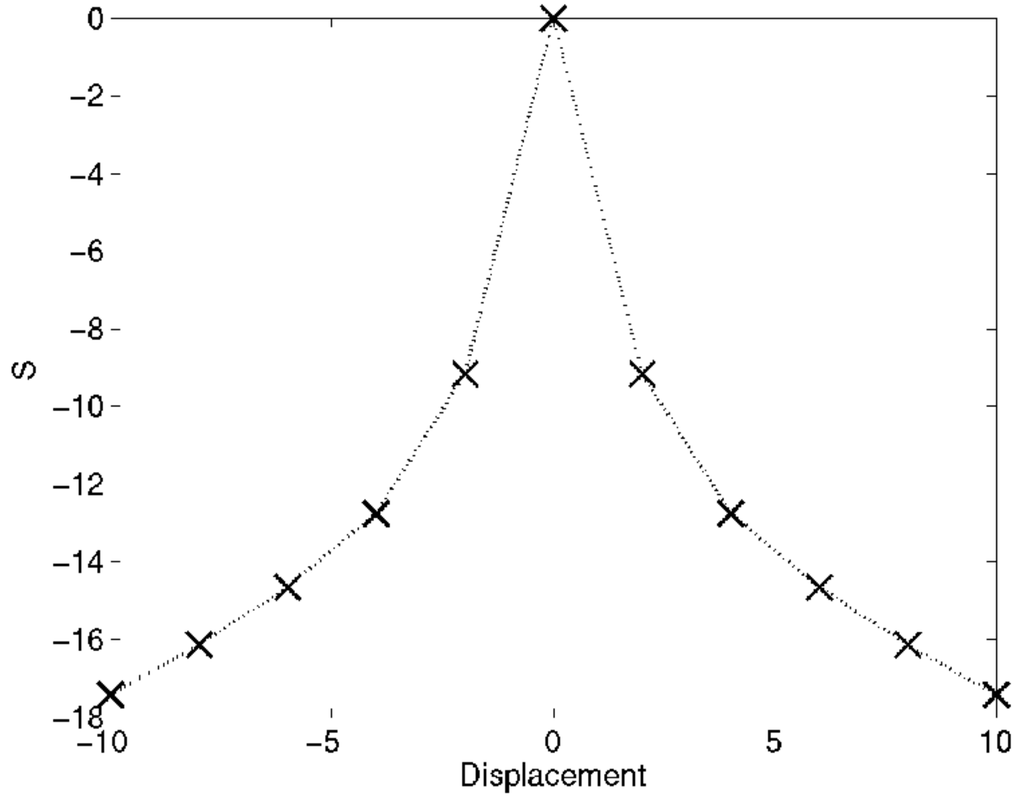


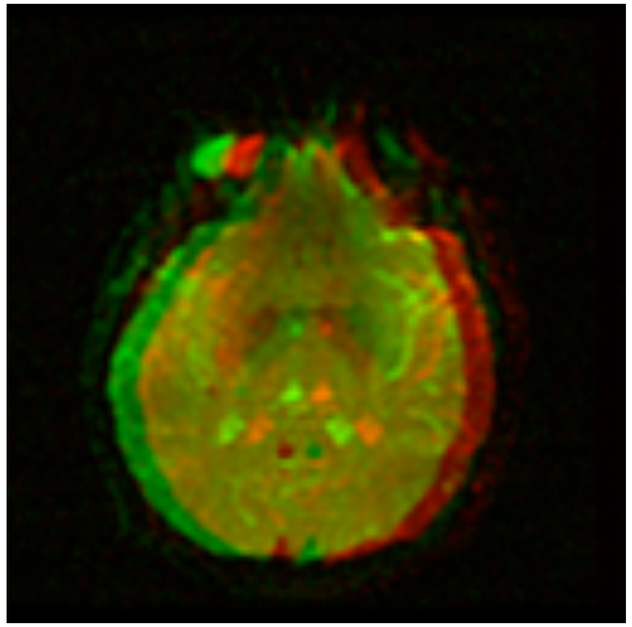
Registration Example



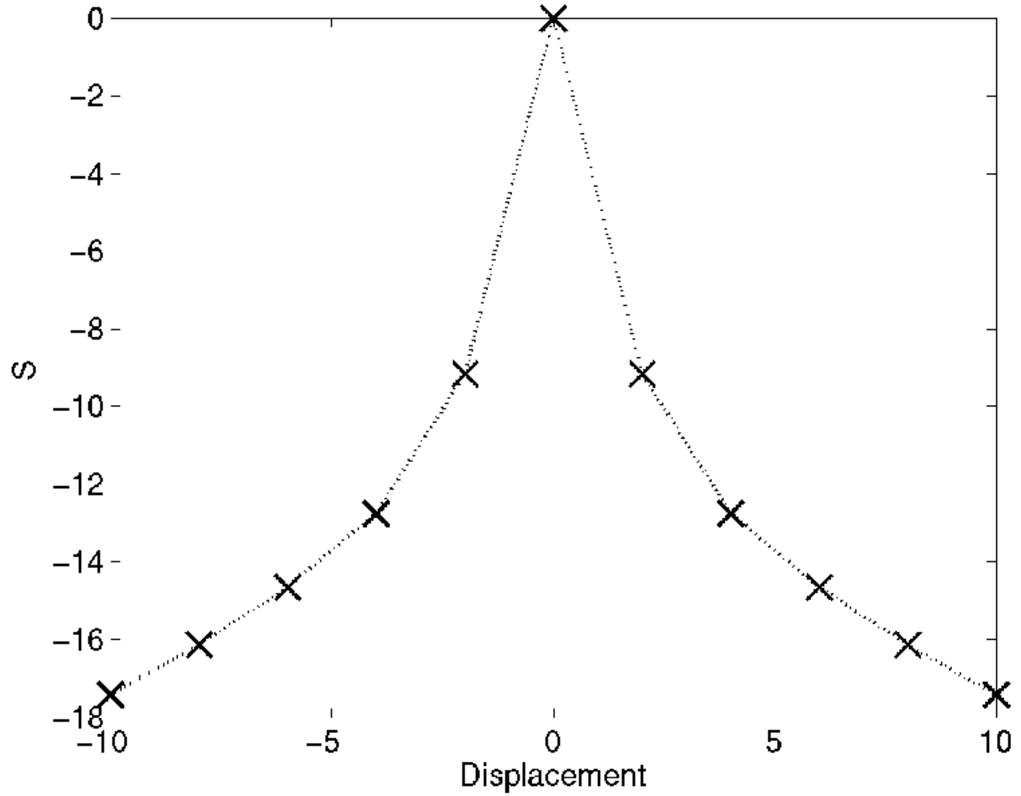


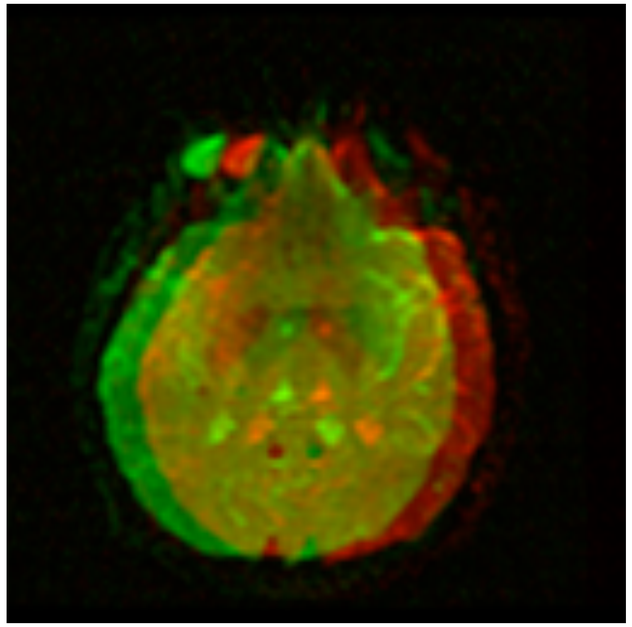
Registration Example



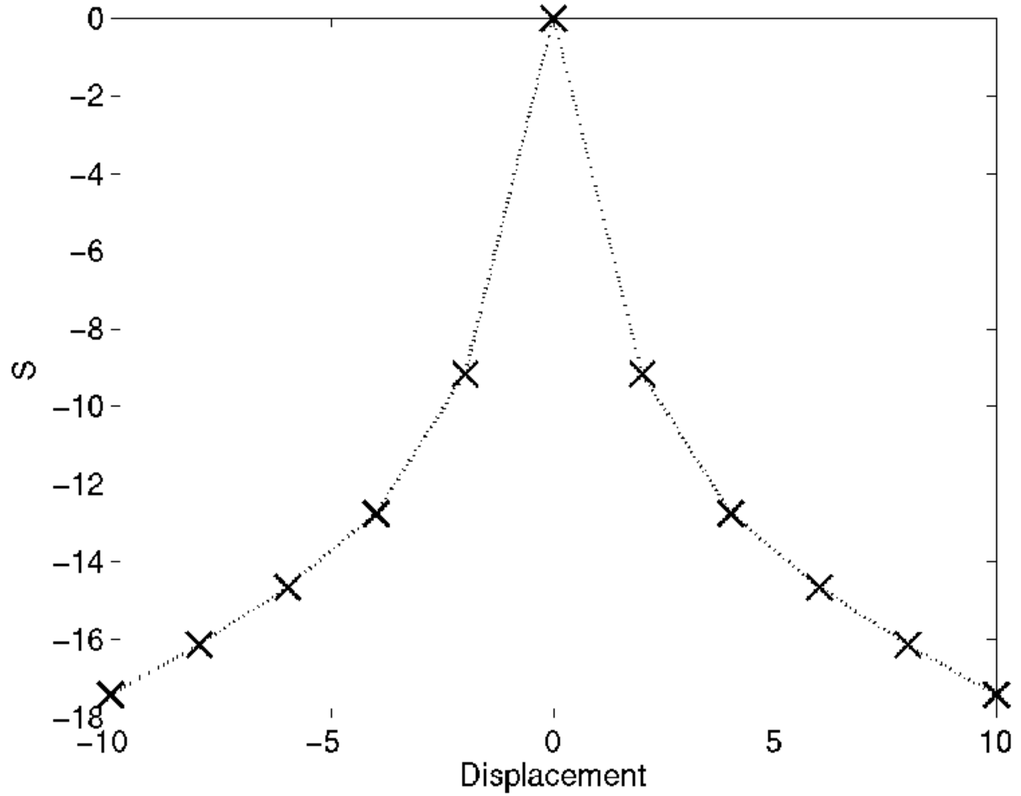


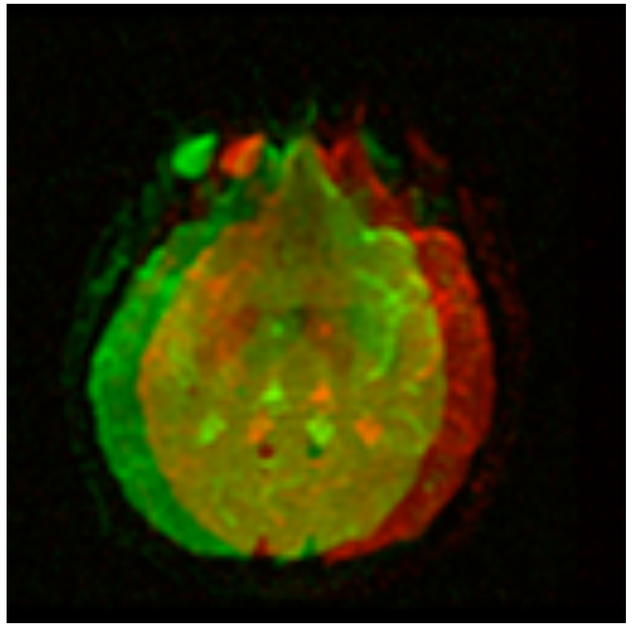
Registration Example





Registration Example





Registration

- **We have to search for the transformation the minimizes S.**
- **Simple in the 1D example.**
- **For more complex transformations, we use *numerical optimization* (see the Matlab function `fminunc` for example).**

Feature-Based Image Metamorphosis

Beier and Neely

[*Feature-Based Image Metamorphosis*, SIGGRAPH, 1992]

Explicit Correspondences

Individual face

Average face

$$u = \frac{(X - P) \cdot (Q - P)}{\| Q - P \|^2} \quad (1)$$

$$v = \frac{(X - P) \cdot Perpendicular\,(Q - P)}{\| Q - P \|} \quad (2)$$

$$X' = P' + u \cdot (Q' - P') + \frac{v \cdot Perpendicular\,(Q' - P')}{\| Q' - P' \|} \quad (3)$$