

Image Transformations

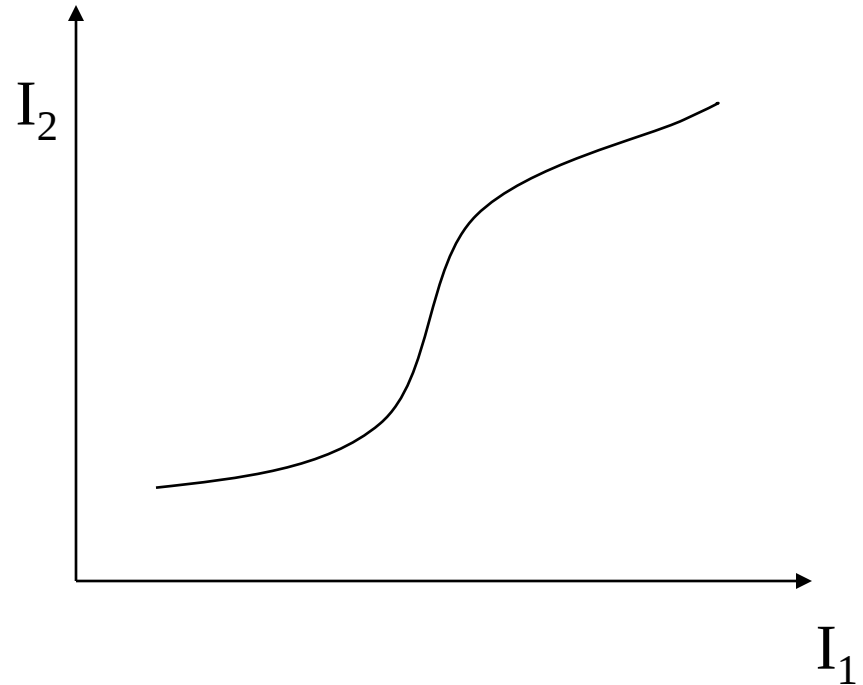
Outline

- Grey-level transformations
 - Histogram equalization
- Geometric transformations
 - Affine transformations
 - Interpolation
 - Warping and morphing.

Grey-level transformations

- Changes the image grey level in each pixel by a fixed mapping $f:\mathcal{R}\rightarrow\mathcal{R}$:

- $I_2(x, y) = f(I_1(x, y))$

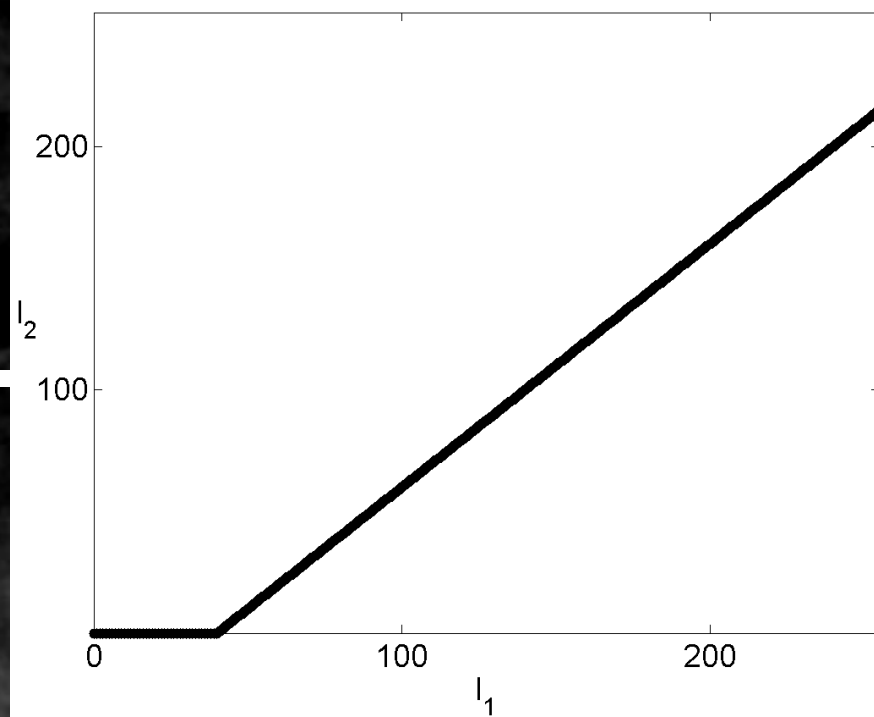


Linear – contrast stretching

- f is a linear function: $f(x) = \alpha x + \beta$
- We must preserve the range of grey level values as $\{0, 1, \dots, 255\}$.

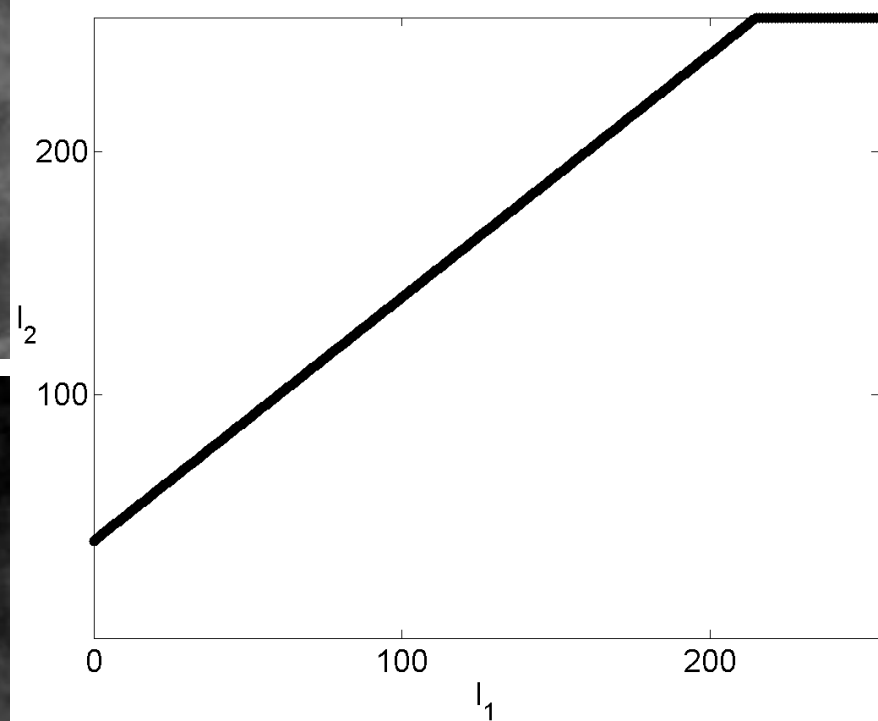


$$\alpha=1.0, \beta=-40$$

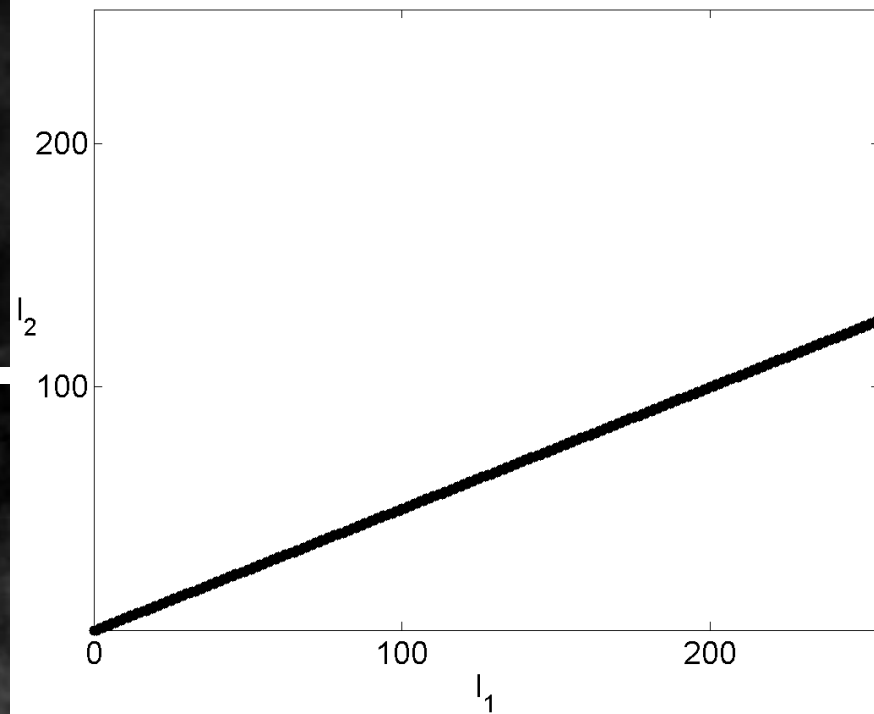




$$\alpha=1.0, \beta=40$$

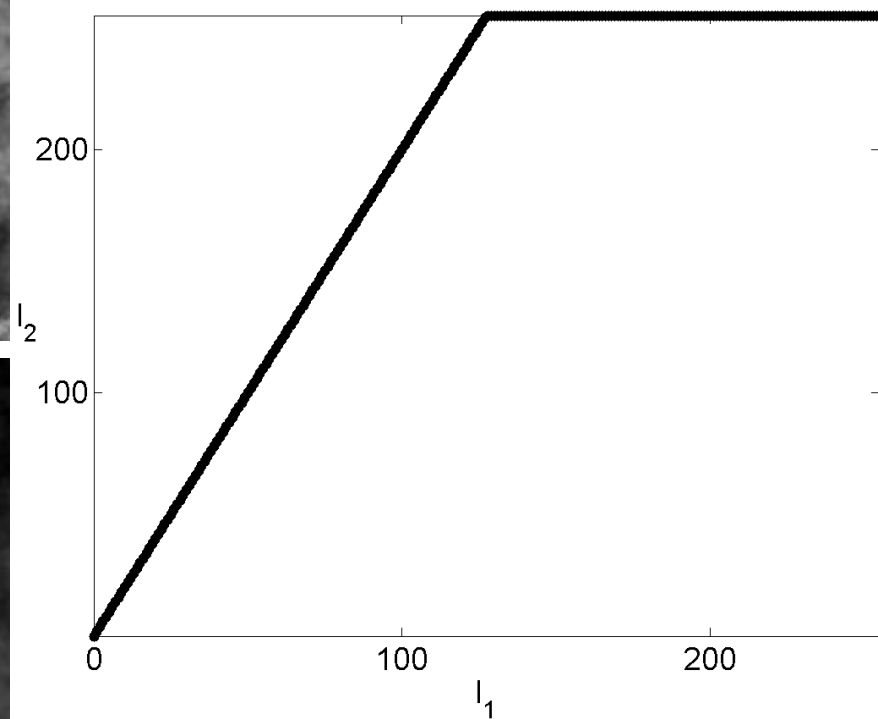


$$\alpha=0.5, \beta=0$$





$$\alpha=2.0, \beta=0$$

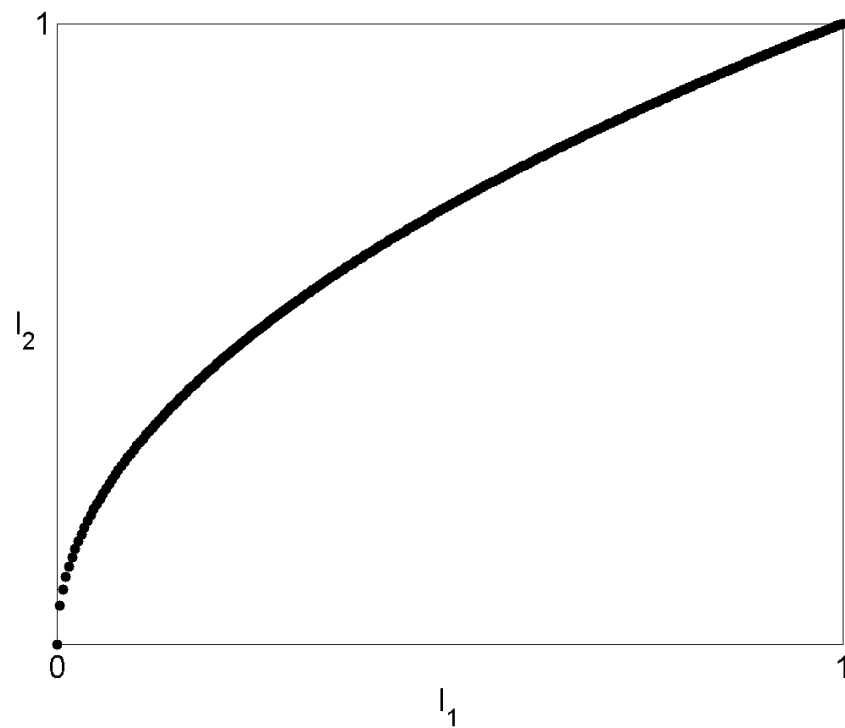


Non-linear – Gamma Correction

- Some non-linear grey-level transformations are useful.
- Gamma correction adjusts for differences between camera sensitivity and the human eye.
- $f(x) = Ax^\gamma$
- $A=255^{(1-\gamma)}$ ensures that the grey scale range is unchanged.

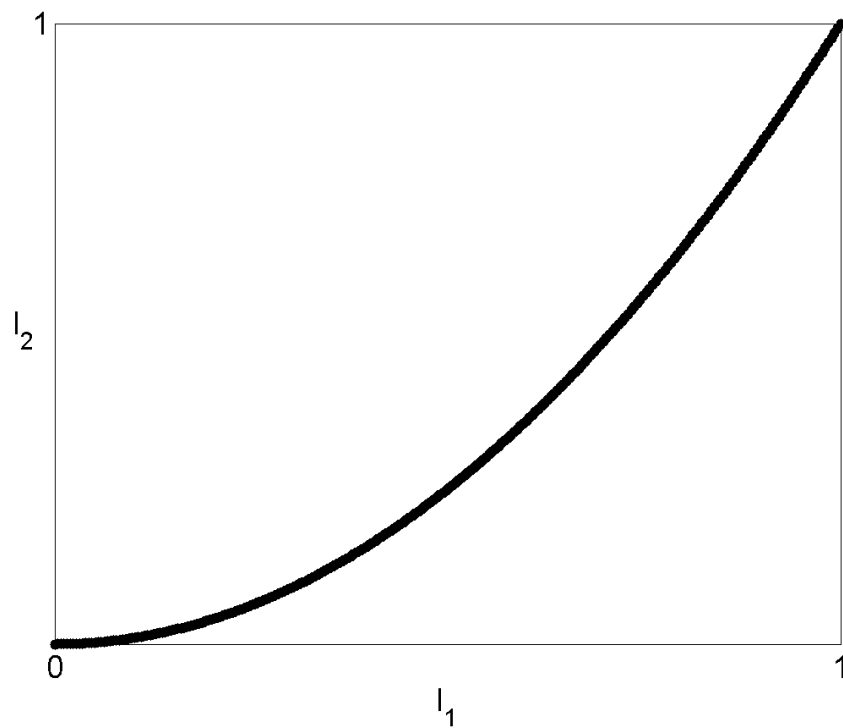


$$\gamma=0.5$$





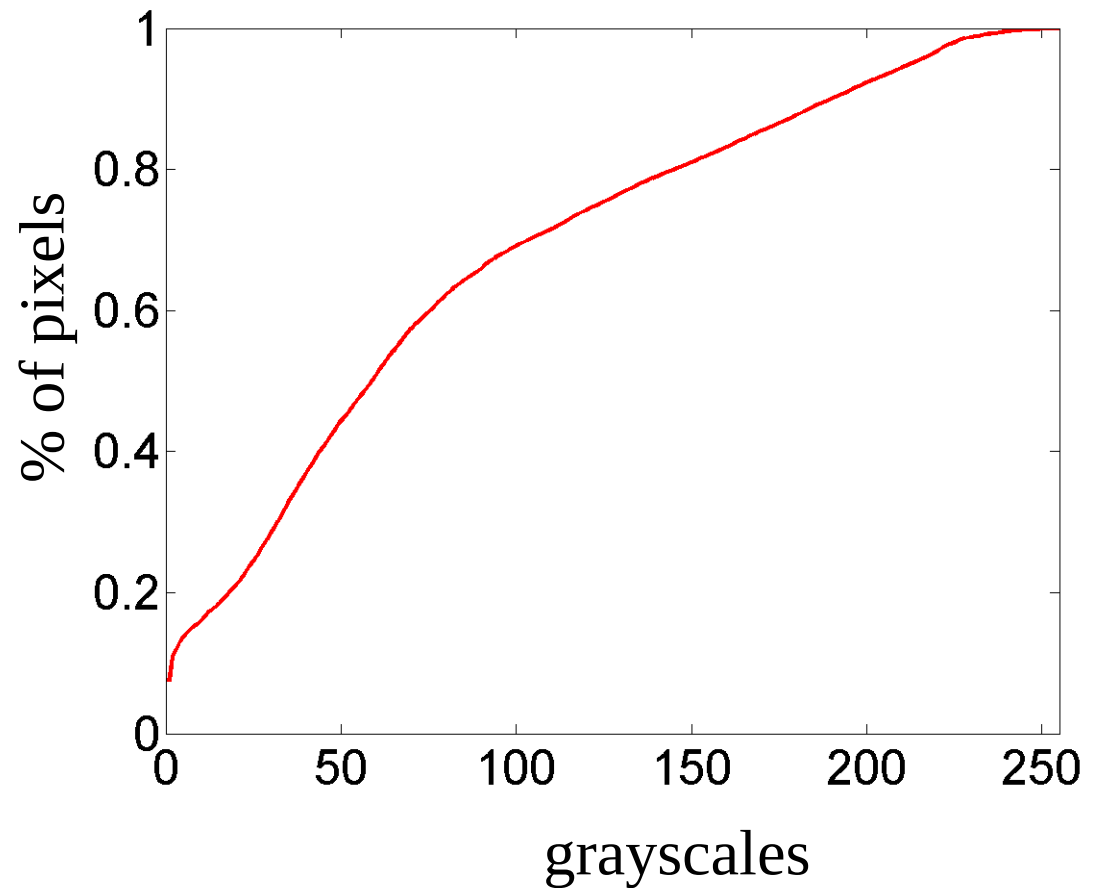
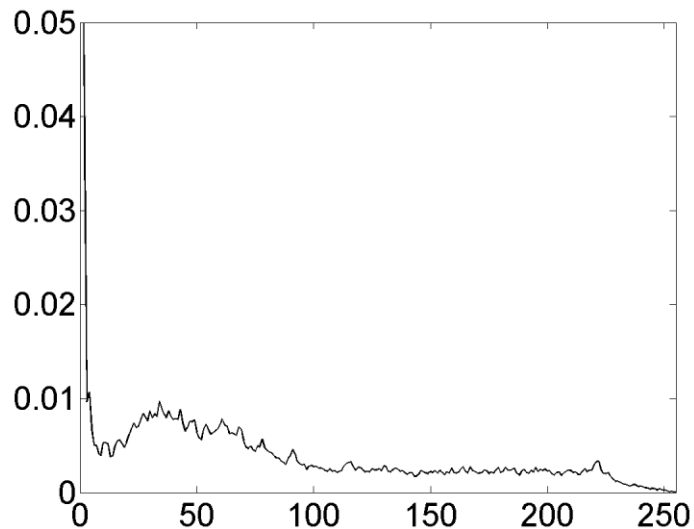
$$\gamma=2$$



Histogram Equalization

- Tries to use all the grey levels equally often.
- The grey level histogram is then flat.
- Use the cumulative histogram for f .

Cumulative histogram



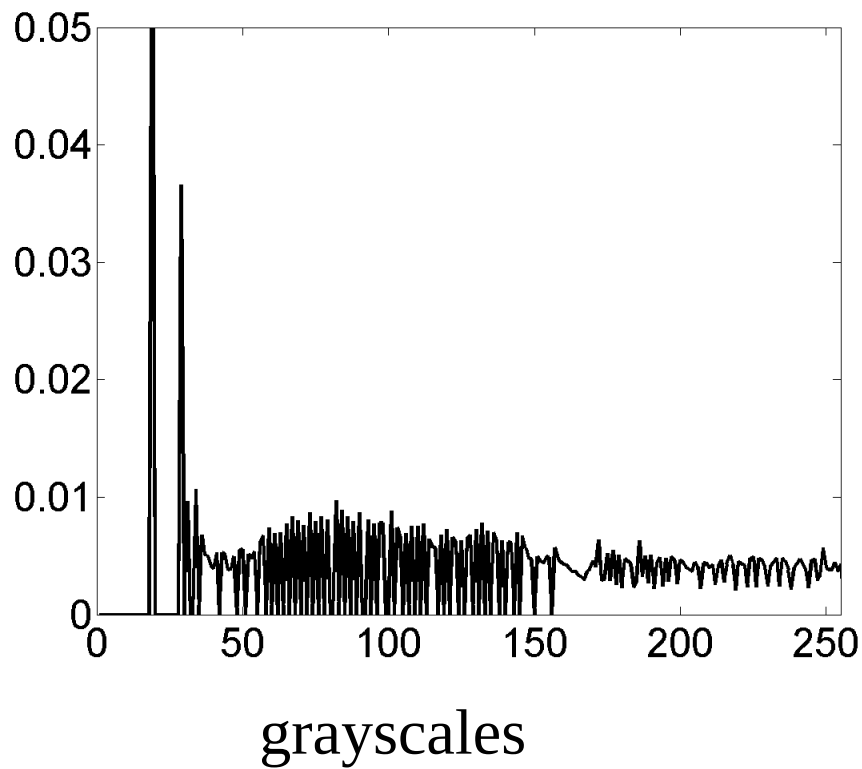
Histogram Equalization

```
function eqIm = histEq(image)
[X,Y] = size(image);
% Construct the cumulative histogram
for i=0:255
    cumHist(i) = sum(sum(image <= i)) / (X*Y);
end
% Use it to transform the grey level in each pixel.
eqIm = fix(255*cumHist(image));
```

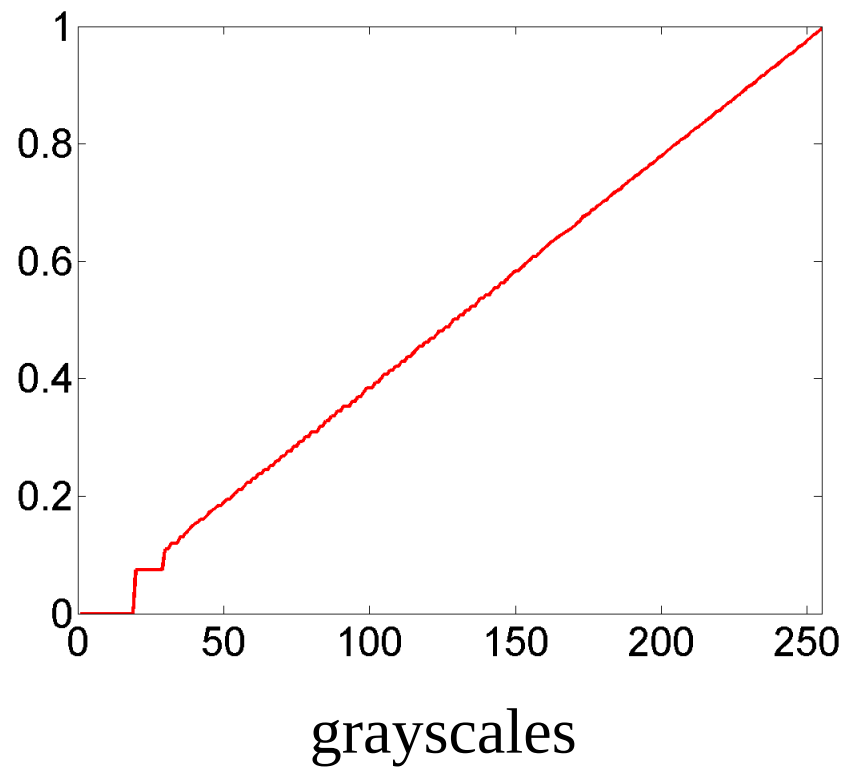
Equalized Image



Equalized histograms



Histogram before
equalization



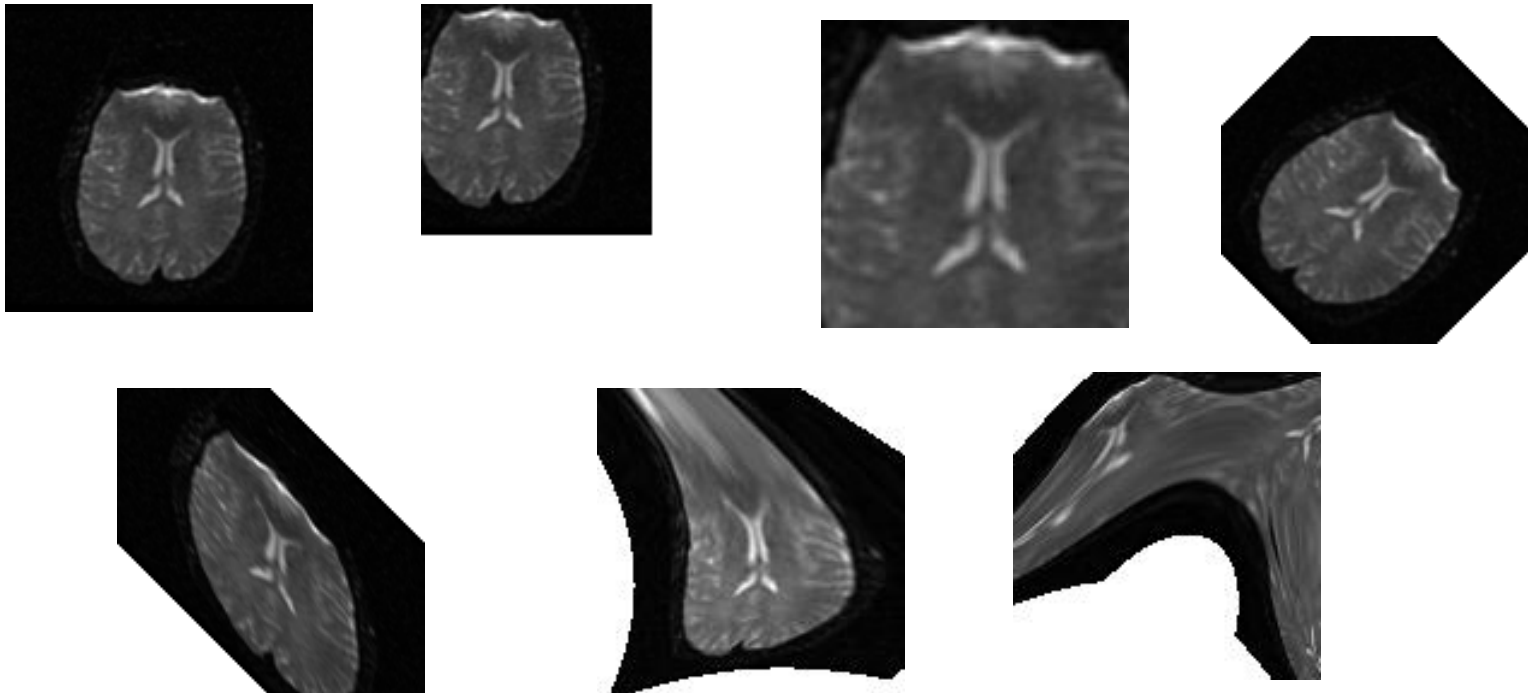
Cumulative histogram
after equalization

Unassessed Exercise

- Write matlab functions for linear grey-level transformations and gamma correction. Try moderate and extreme settings on an image and observe the effects they have. Plot the grey-level histograms before and after the grey-level transformations.

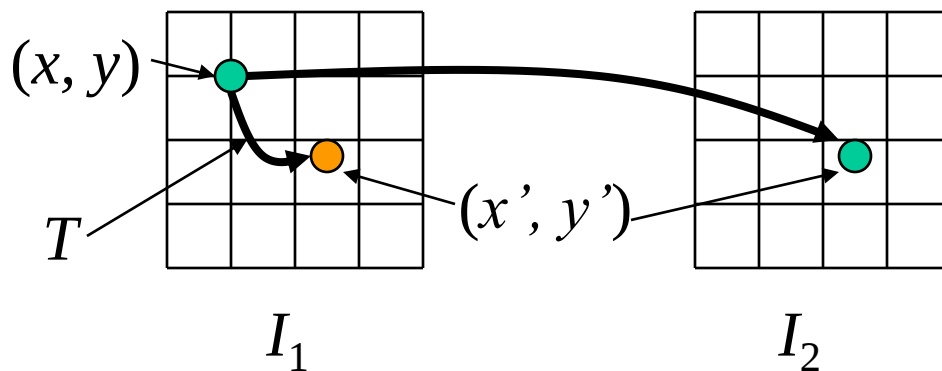
Geometric Transformations

- Change the location of image features



Geometric Transformations

- I_1 is the original image, I_2 is the warped image.
- $I_2(x', y') = I_1(x, y)$
- $(x', y') = T(x, y)$



Affine Transformations

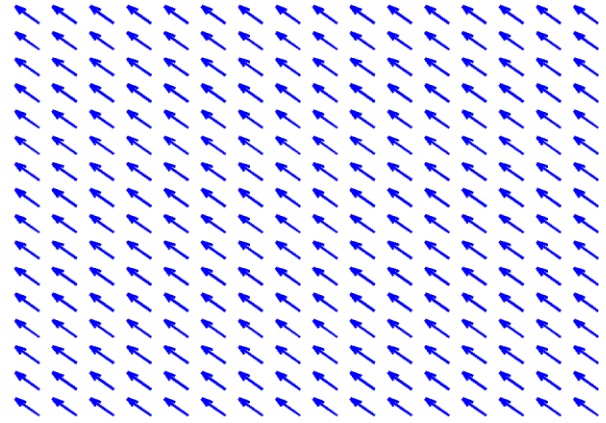
- $x' = ax + by + t_x$
- $y' = cx + dy + t_y$

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

- Translation, scaling, rotation, shear.

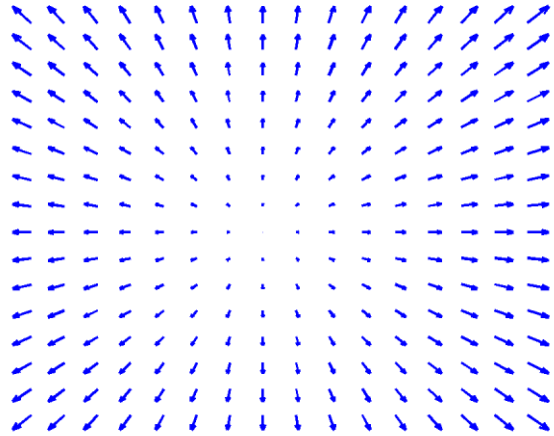
Translation

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} t_x \\ t_y \end{pmatrix} = \begin{pmatrix} -W / 4 \\ -H / 4 \end{pmatrix}$$



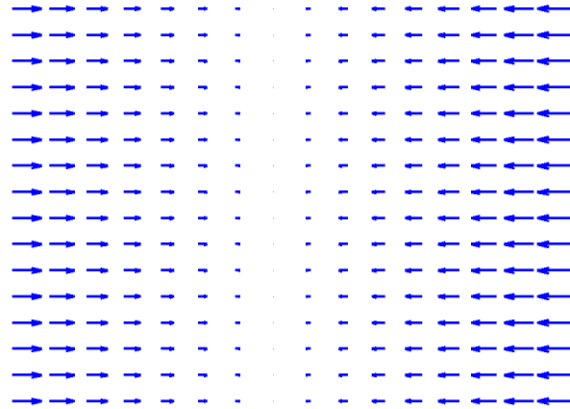
Scaling

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}, \quad \begin{pmatrix} t_x \\ t_y \end{pmatrix} = \begin{pmatrix} -W/2 \\ -H/2 \end{pmatrix}$$



Stretch

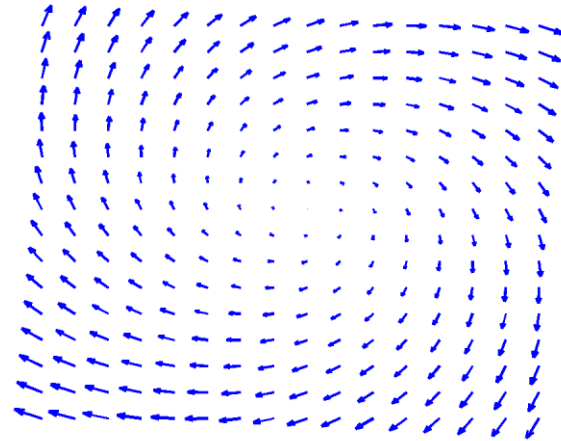
$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 0.5 & 0 \\ 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} t_x \\ t_y \end{pmatrix} = \begin{pmatrix} W / 4 \\ 0 \end{pmatrix}$$



Rotation

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

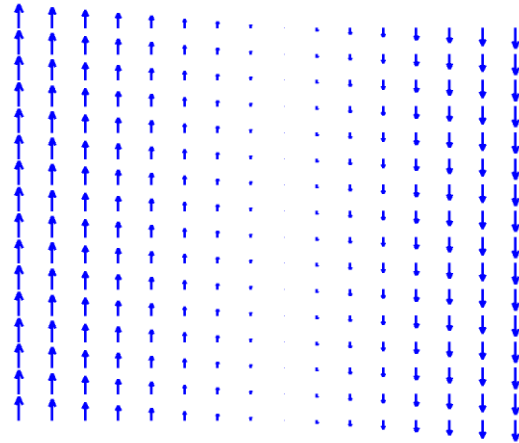
$$\begin{pmatrix} t_x \\ t_y \end{pmatrix} = \begin{pmatrix} -(W(\cos \theta - 1) + H \sin \theta) / 2 \\ (W \sin \theta - H(\cos \theta - 1)) / 2 \end{pmatrix}$$



Shear (along y axis)

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

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



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:

$$\begin{aligned} x' &= ax + by + t_x \\ y' &= cx + dy + t_y \end{aligned} \quad \Rightarrow \quad \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

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

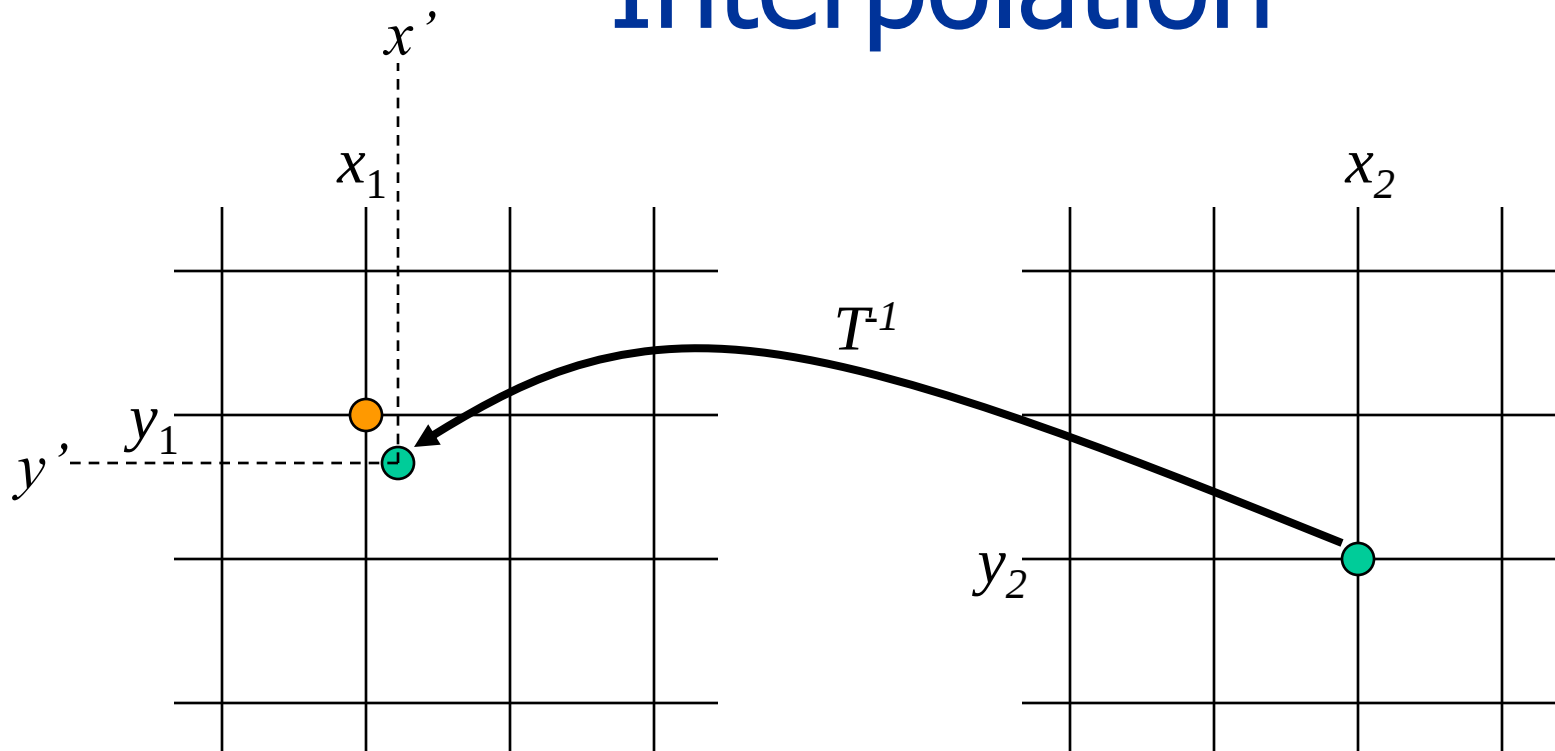
- Always use the inverse transformation:

```
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
end
```

Interpolation

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

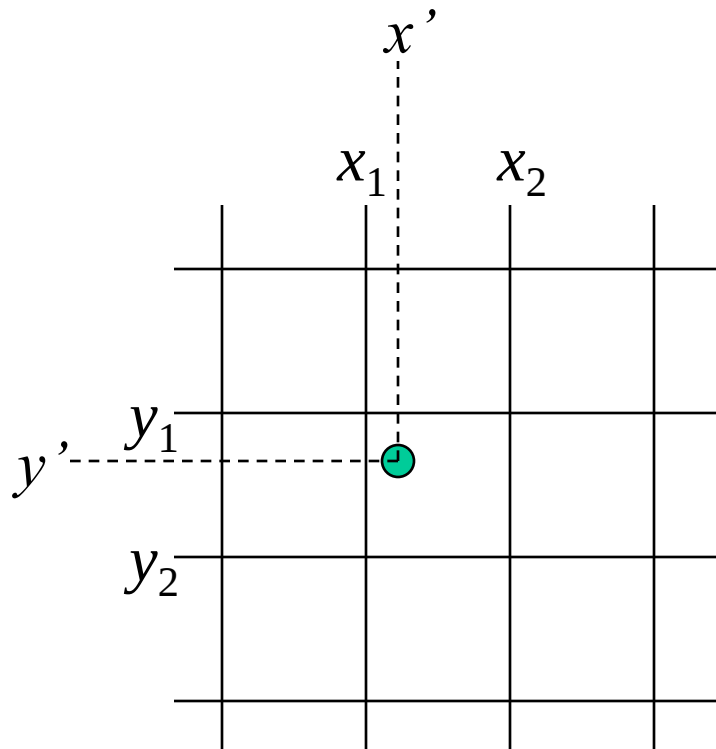
Nearest Neighbour Interpolation



Take the value at the nearest grid point:

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

Bilinear interpolation



Weighted average from neighbouring grid points:

$$\begin{aligned} I_1(x', y') = & \Delta x \Delta y I_1(x_2, y_2) \\ & + \Delta x (1 - \Delta y) I_1(x_2, y_1) \\ & + (1 - \Delta x) \Delta y I_1(x_1, y_2) \\ & + (1 - \Delta x) (1 - \Delta y) I_1(x_1, y_1) \end{aligned}$$

$$\Delta x = x' - x_1, \quad \Delta y = y' - y_1.$$

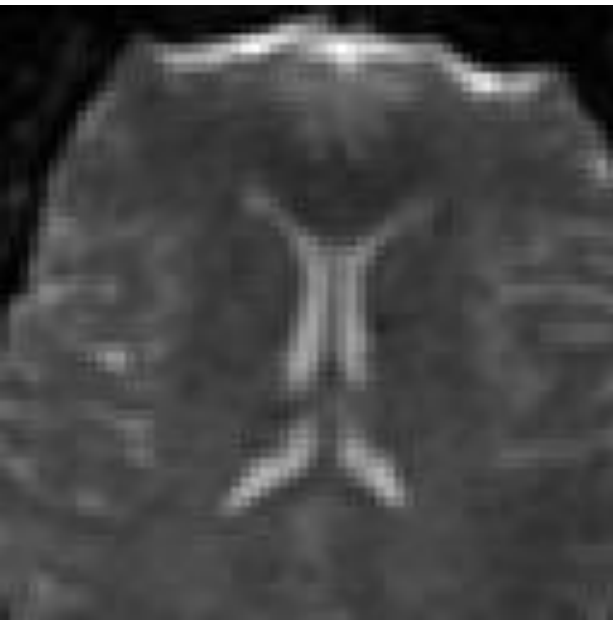
Higher order interpolation

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

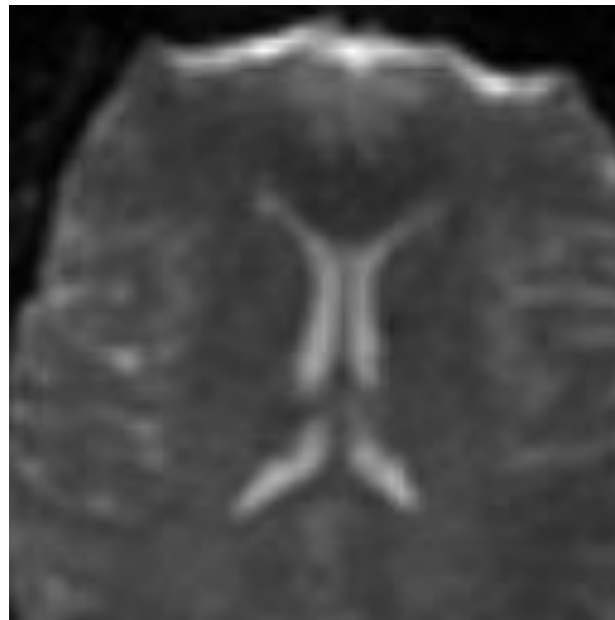
interp2

- The matlab function `interp2` does nearest neighbour, linear and cubic interpolation.
- Look it up and try it out!

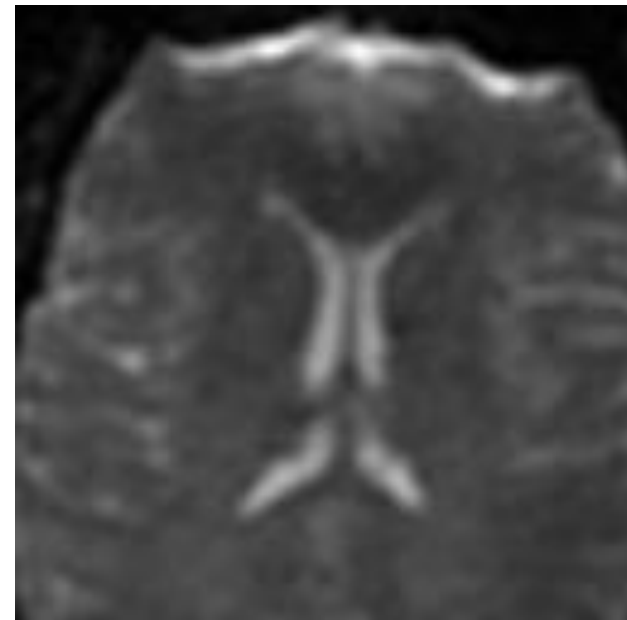
Interpolation comparison



Nearest
neighbour



Bilinear



Cubic

Warps

- Polynomial transformations, e.g., quadratic transformations:
- $x' = a_0 + a_1x + a_2y + a_3x^2 + a_4xy + a_5y^2$
- $y' = b_0 + b_1x + b_2y + b_3x^2 + b_4xy + b_5y^2$
- Affine transformations map lines to lines.
- They are *first-order* polynomial transformations.
- Higher-order polynomials can bend lines.

Quadratic warp example



Control point warps

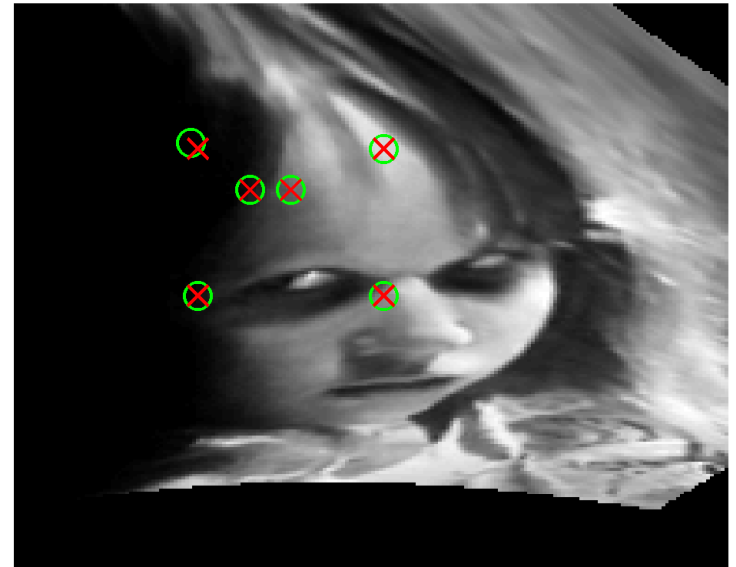
- Moves some pixels to specified locations.
- Interpolate the displacement at intermediate positions.
- Find a polynomial warp P that maps:
 $(x_1, y_1) \rightarrow (x'_1, y'_1)$
 $(x_2, y_2) \rightarrow (x'_2, y'_2)$
:
 $(x_m, y_m) \rightarrow (x'_m, y'_m)$

Control point warps

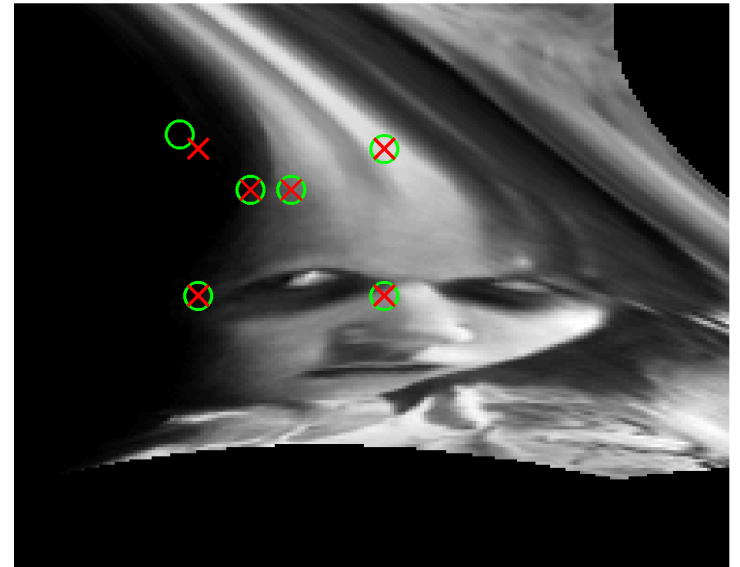
$$\begin{pmatrix} x'_1 & y'_1 \\ x'_2 & y'_2 \\ \vdots & \vdots \\ x'_m & y'_m \end{pmatrix} = \begin{pmatrix} 1 & x_1 & y_1 & x_1^2 & x_1 y_1 & y_1^2 \\ 1 & x_2 & y_2 & x_2^2 & x_2 y_2 & y_2^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_m & y_m & x_m^2 & x_m y_m & y_m^2 \end{pmatrix}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

- $A = X P$
- Least squares estimate of P is ($m \geq 6$):
- $P = (X^T X)^{-1} X^T A$.

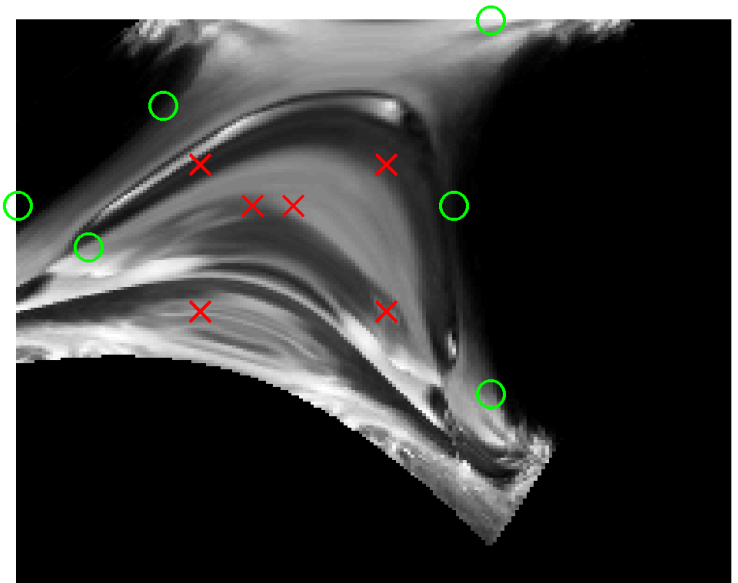
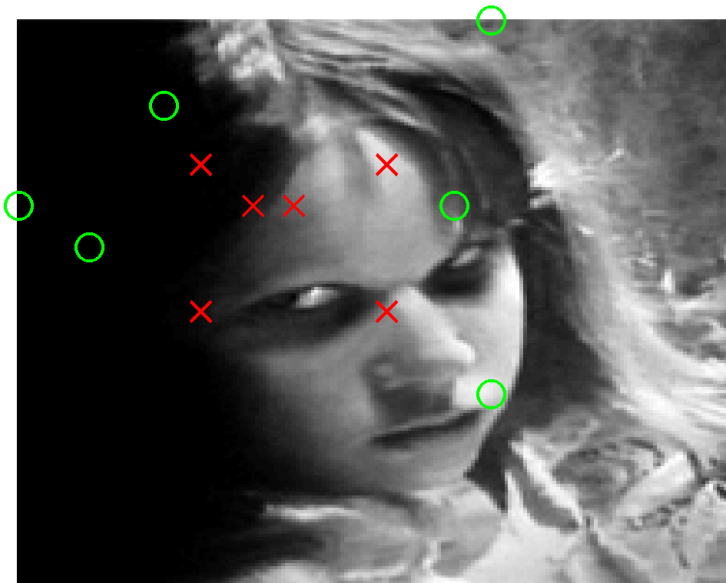
Example – Two pixel displacement



Five Pixel Displacement



Extreme warp



Overdetermined



Applications

- Special effects
 - Film industry
 - Computer games
- Image registration
 - Medical imaging
 - Security

Image Morphing



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



Landmarks

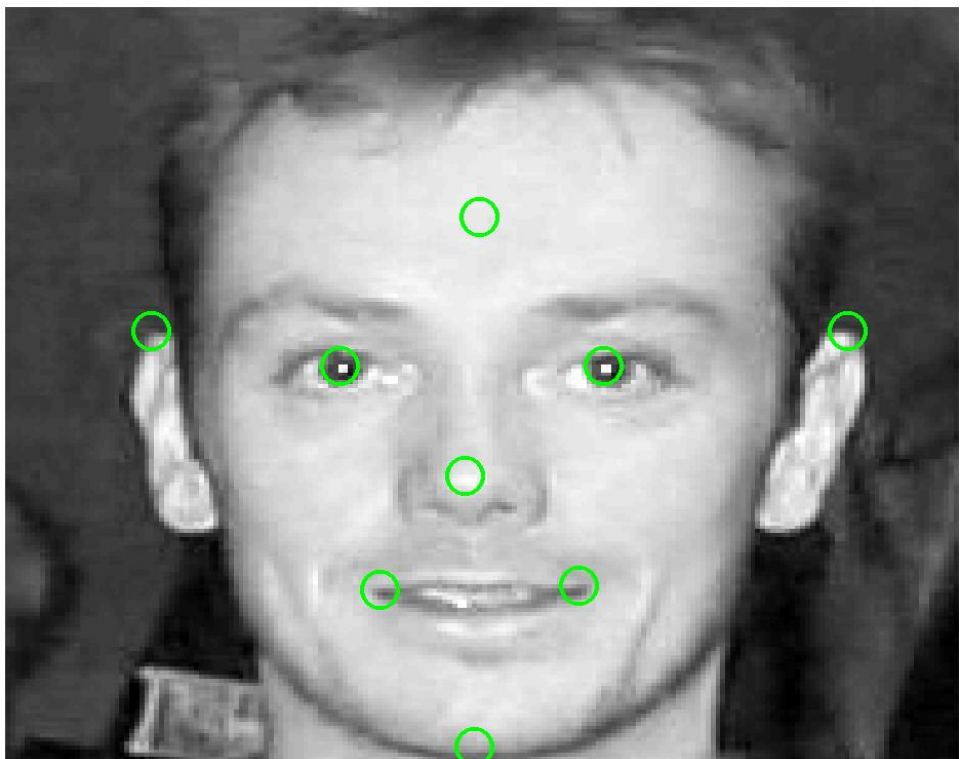


Image Registration

- Determines the transformation that aligns two similar images

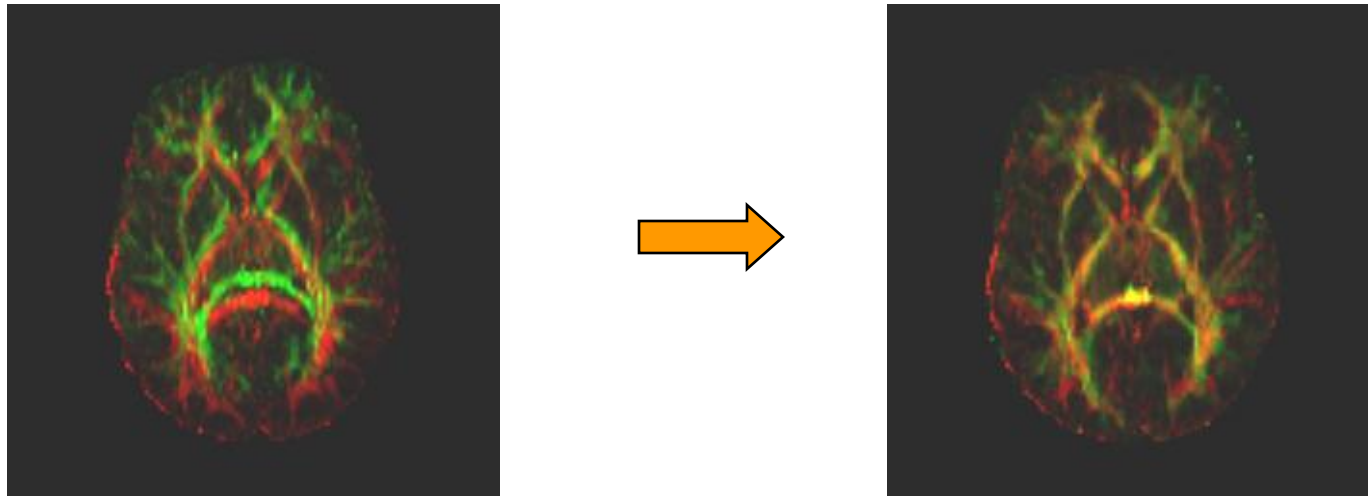


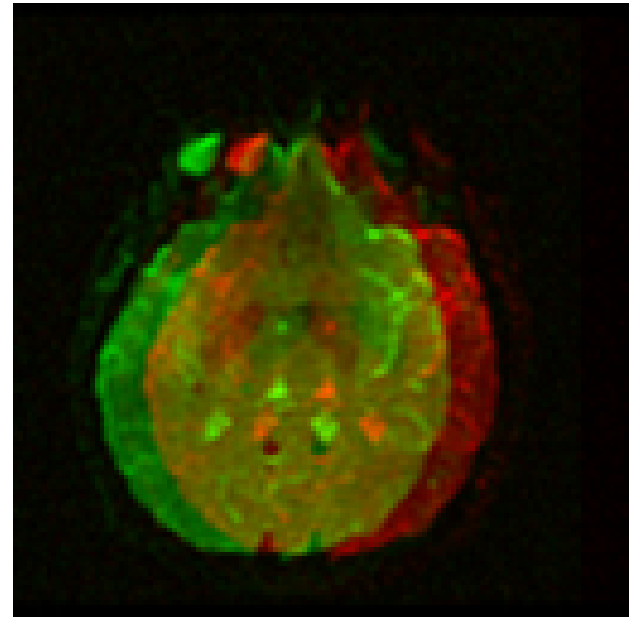
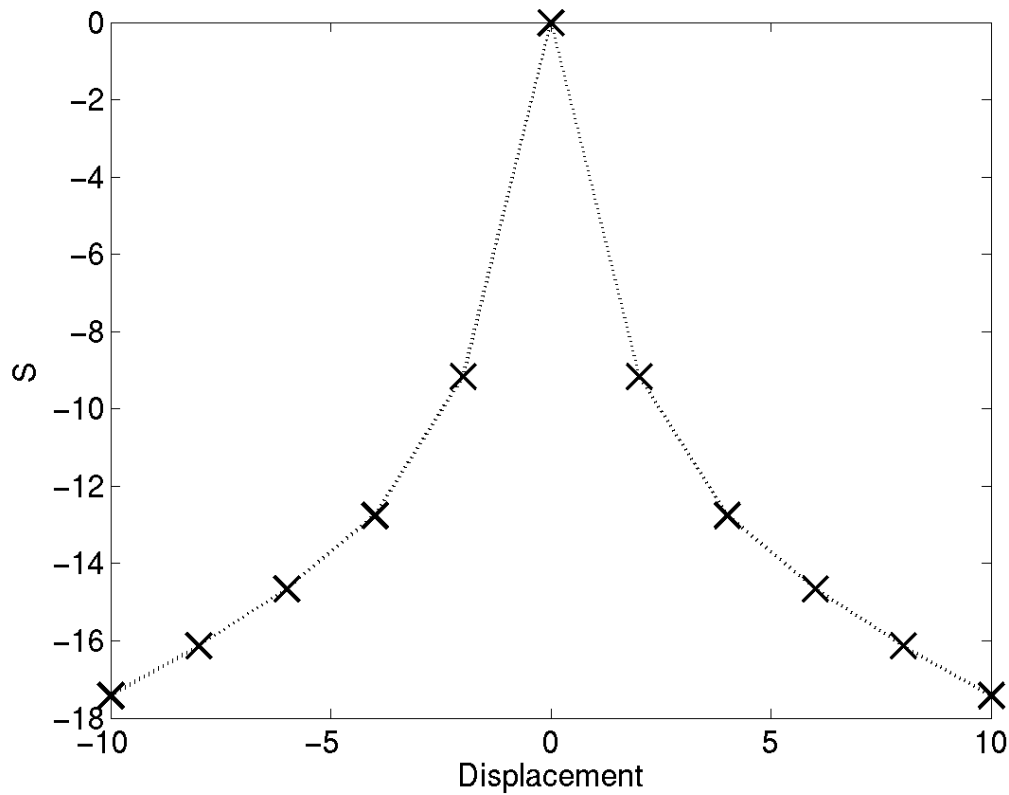
Image Similarity

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

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

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

Registration example



Registration

- We have to search for the transformation that 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

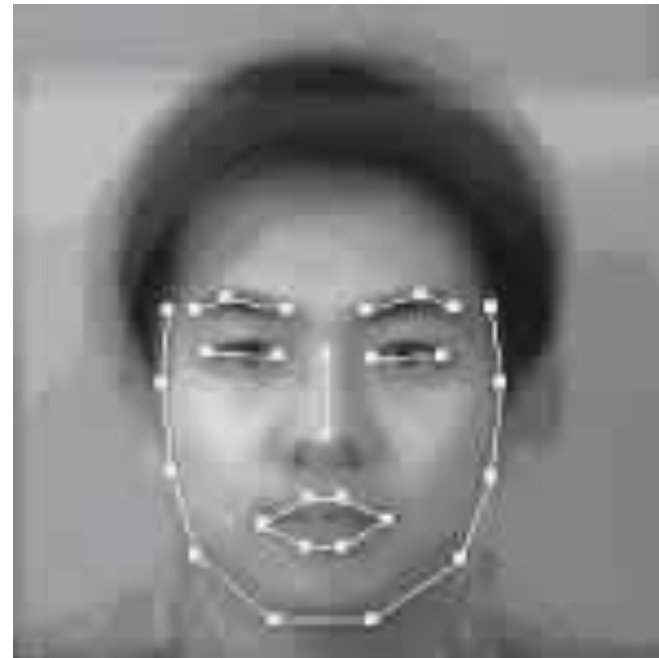
Beier & Neely Siggraph'92

(Quick overview)

Explicit Correspondences



Individual face



Average face

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

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

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

