# Image Warping and Alignment

AJIT RAJWADE

# Basics

- A **digital** image – a version of the visual stimulus sampled at discrete locations, with discretized values
- Can be regarded as a function $I = f(x,y)$ where $(x,y)$ are spatial (integer) coordinates in typically a rectangular domain $\Omega = [0,W-1] \times [0,H-1]$.
- Each ordered pair $(x,y)$ is called a **pixel**.
- The pixel is generally square (sometimes rectangular) in shape.

# Basics

- Pixel dimensions (height/width of the pixel) relate to the **spatial resolution** of the sensor in the camera that collects light reflected from a scene.

- Remember: the actual visual signal is analog, but digital cameras capture a **discrete** version of it, and also quantize the intensity values.

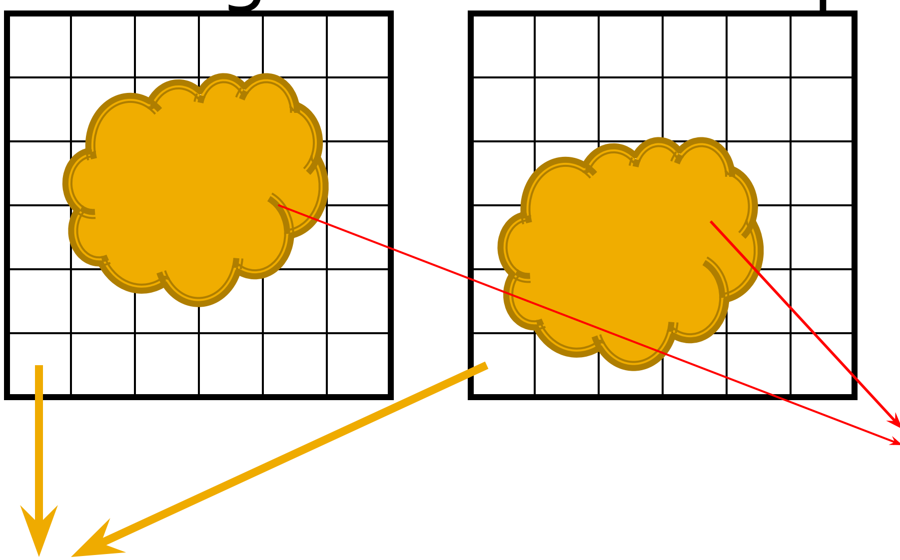
- That is they capture the visual stimulus only at specific points (x,y), usually evenly spaced from each other in both X and Y directions.

# Basics

- In a typical photographic grayscale image, the intensity values $f(x,y)$ lie in the range from 0 to 255 (8 bit integers).
- They are quantized versions continuous values corresponding to the actual light intensity that strikes a light sensor in the camera.

# Image Alignment

- Consider images $\mathbf{I}_1$ and $\mathbf{I}_2$ of a scene acquired through different viewpoints.

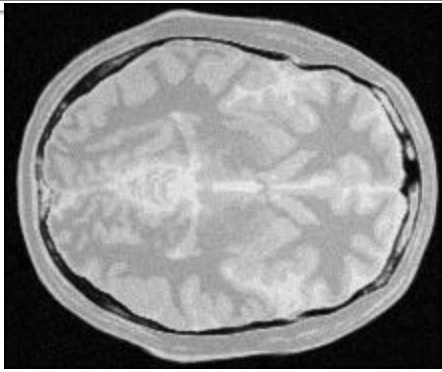


Pixels in **physical** correspondence (containing measurements of the same physical point, but not necessarily the same coordinate values in the image domain $\Omega$)

Pixels in **digital** correspondence (same coordinate values in the image domain $\Omega$, not necessarily containing measurements of the same physical point)

# Image Alignment

- $\mathbf{I}_1$ and $\mathbf{I}_2$ are said to be aligned if for every $(x,y)$ in the domain $\Omega$, the pixels at $(x,y)$ in $\mathbf{I}_1$ and $\mathbf{I}_2$ are in physical correspondence.
- If not, the images are said to be **misaligned** with respect to each other.
- Or we say there is **relative motion** between the images.
- Image alignment (also called **registration**) is the process of correcting for the relative motion between $\mathbf{I}_1$ and $\mathbf{I}_2$.



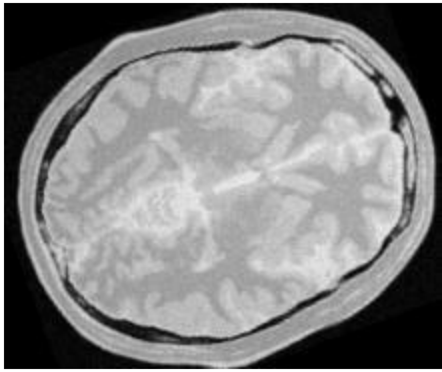


Image taken from the Brainweb database of the Montreal Neurological Institute

# Motion Models

- Let us denote the coordinates in $\mathbf{I}_1$ as $(x_1,y_1)$ and those in $\mathbf{I}_2$ as $(x_2,y_2)$.
- Translation: $\forall (x_1,y_1) \in \Omega, x_2 = x_1 + t_x, y_2 = y_1 + t_y$

$$\rightarrow \begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix}$$

- Rotation about point (0,0) anti-clockwise through angle $\theta$

$$x_2 = x_1 \cos\theta - y_1 \sin\theta$$

$$y_2 = x_1 \sin\theta + y_1 \cos\theta$$

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$$

2D Rotation matrix (orthonormal matrix)

# Motion Models

- Rotation about point $(x_c, y_c)$ anti-clockwise through angle $\theta$

$$x_2 = (x_1 - x_c)\cos\theta - (y_1 - y_c)\sin\theta + x_c$$

$$y_2 = (x_1 - x_c)\sin\theta + (y_1 - y_c)\cos\theta + y_c$$

$$\begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & x_c \\ \sin\theta & \cos\theta & y_c \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 - x_c \\ y_1 - y_c \\ 1 \end{pmatrix}$$

-Perform translation such that $(x_c, y_c)$ coincides with the origin (0,0).
-Rotate about the new origin.
-Translate back.
-The extra ones (third row) are called **homogeneous coordinates** – they facilitate using matrix multiplication to represent translations.

# Motion Models

- Rotation and translation:

$$x_2 = (x_1 - x_c)\cos\theta - (y_1 - y_c)\sin\theta + t_x$$

$$y_2 = (x_1 - x_c)\sin\theta + (y_1 - y_c)\cos\theta + t_y$$

$$\begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 - x_c \\ y_1 - y_c \\ 1 \end{pmatrix}$$

- Affine transformation: (rotation, scaling and shearing) besides translation

$$\begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} & t_x \\ A_{21} & A_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix}$$

Assumption: the 2 x 2 sub-matrix **A** is NOT rank deficient, otherwise it will transform two-dimensional figures into a line or a point

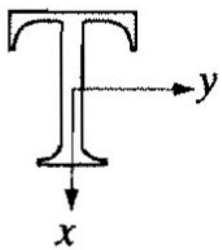
# Motion Models

- Scaling about the origin

Identity

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

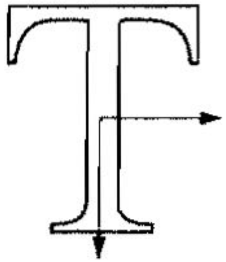
$$x = v$$
$$y = w$$




Scaling

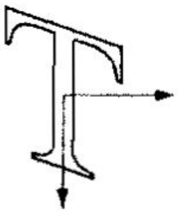
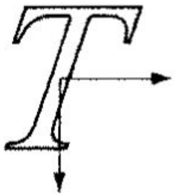
$$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$x = c_x v$$
$$y = c_y w$$



# Motion Models

- Shearing:

| | | | |
|---|---|---|---|
| Shear (vertical) | $\begin{bmatrix} 1 & 0 & 0 \\ s_v & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ | $x = v + s_v w$<br>$y = w$ |  |
| Shear (horizontal) | $\begin{bmatrix} 1 & s_h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ | $x = v$<br>$y = s_h v + w$ |  |

# Motion Models

- The 2D affine motion model (including translation in X and Y direction) includes 6 degrees of freedom.
- Note: this motion model accounts for in-plane motion only (example: not an appropriate model for “3D head profile view versus head frontal view”)
- Note: even with in-plane motion, there exist more complicated motion models, but we will stick to this one for now.

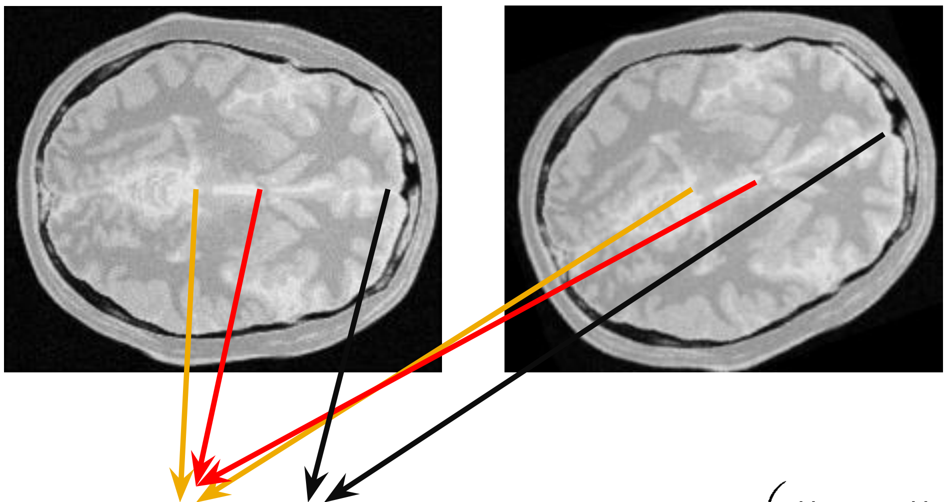
# Motion Models

- Composition of multiple types of motion is given by the multiplication of their corresponding matrices.
- So, if you first scale (matrix **S**) and then rotate (matrix **R**), then the resultant transformation = **RS**.
- Note: most motion compositions are not commutative (**RS** is not equal to **SR**).

# Motion Models

- In actual coding, you will typically not use matrices.
- Rather you will implement the formula as is.
- So why is matrix-based motion representation useful?
- Because it allows for a compact way to represent the composition of many different types of motion.

# Alignment with control points



Solve for unknown parameters using least-squares framework (i.e. pseudo-inverse)

Apply the motion based on these parameters to the first image

Control points: pairs of physically corresponding points – maybe marked out manually, or automatically using geometric properties of the image.

Number of control points $k$ MUST be >= $u/2$, where $u$ = number of unknown parameters in the motion model (each point has two coordinates – x and y). The number of control points is several times smaller than the number of image pixels.

$$\begin{pmatrix} x_{21} & x_{22} & . & . & x_{2k} \\ y_{21} & y_{22} & . & . & y_{2k} \\ 1 & 1 & . & . & 1 \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} & t_x \\ A_{21} & A_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{11} & x_{12} & . & . & x_{1k} \\ y_{11} & y_{12} & . & . & y_{1k} \\ 1 & 1 & . & . & 1 \end{pmatrix}$$

# Alignment with control points

- Not always feasible, if it requires manual intervention
- Error-prone
- There are methods for finding matching control points automatically (eg: the popular SIFT technique), but we will not cover them in this course.

# Alignment with mean squared error

- Mean squared error is given by:

$$MSSD = \frac{1}{N}\sum_{x,y\in\Omega}(I_1(x,y) - I_2(x,y))^2, N = \#\text{pixels in field of view (see defn. in later slides)}$$

- Find motion parameters as follows:

$$\boldsymbol{T}^* = \arg\min MSSD_T(I_1(\boldsymbol{v}), I_2(\boldsymbol{T}\boldsymbol{v}))$$

$$\boldsymbol{T} = \begin{pmatrix} A_{11} & A_{12} & t_x \\ A_{21} & A_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix}, \boldsymbol{v} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Find transformation matrix T which produces the least value of MSSD

# Alignment with mean squared error

- For simplicity, assume there was only rotation and translation.
- Then we have

$$\boldsymbol{T}^* = \arg\min MSSD_{\boldsymbol{T}}(I_1(\boldsymbol{v}), I_2(T\boldsymbol{v}))$$

$$\boldsymbol{T} = \begin{pmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{pmatrix}, \boldsymbol{v} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

# Alignment with mean-squared error

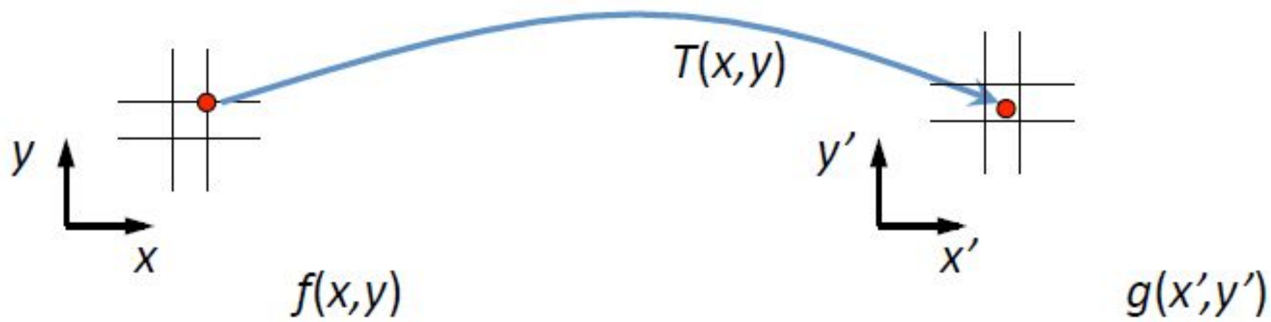
- There are many ways to do this minimization. The simplest but least efficient way is to do a brute-force search.
- Sample $\theta$, $t_x$ and $t_y$ uniformly from a certain range (example: $\theta$ from -45 to +45, $t_x$ or $t_y$ from -30 to +30).
- Apply this motion to $\mathbf{I}_1$ keeping $\mathbf{I}_2$ fixed (alternatively to $\mathbf{I}_2$ keeping $\mathbf{I}_1$ fixed, if the matrix is invertible), and compute the MSSD.
- Each time, compute the MSSD. Pick the parameter values corresponding to minimum MSSD.

# Image Warping

- **Forward warping method:**

✔ Apply the transformation **T** to every coordinate vector **v** = [**x y**] in the original image to yield new coordinate **Tv**.

✔ Copy the intensity value from **v** in the original image to the new location in the warped image. Careful handling is needed if **Tv** is not an integer (highly likely).

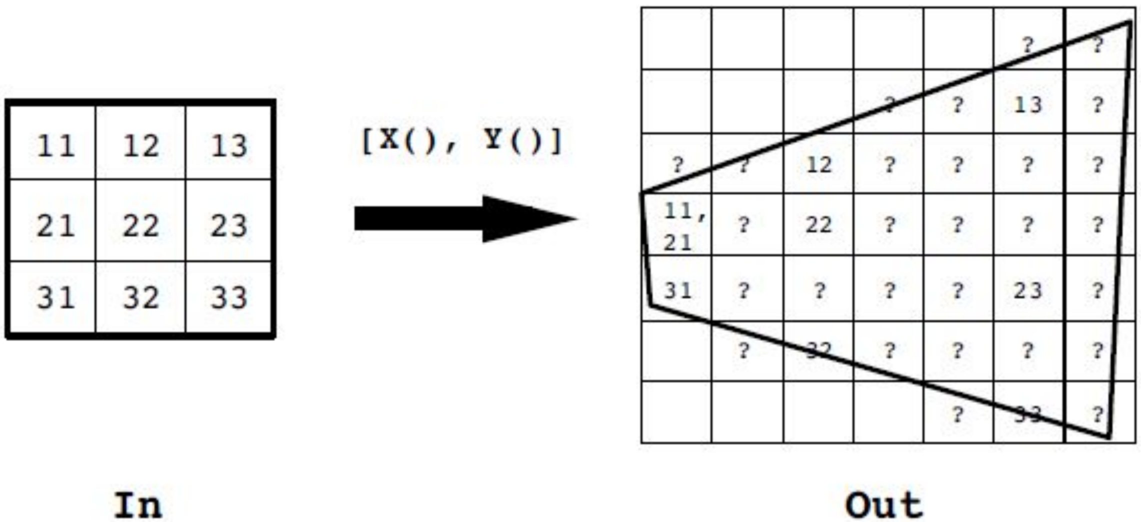


# Image Warping

- Forward warping:

-Can leave the destination image with some holes if you scale up.

-Can lead to multiple answers in one pixel if you scale down.

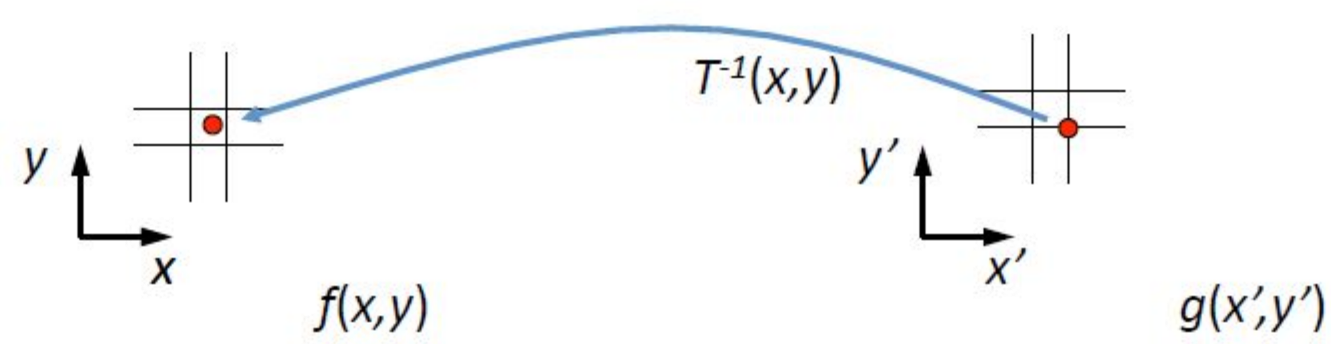


# Image warping

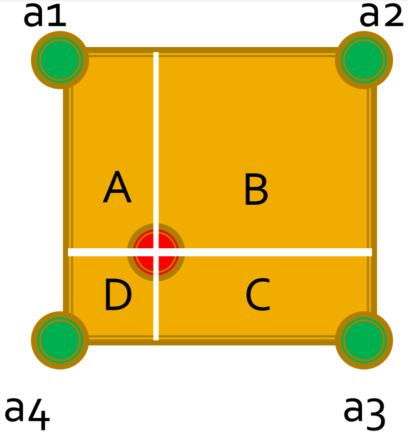
- Reverse warping:

✔ For every coordinate **v** = [x y] in the destination image, copy the intensity value from coordinate $\mathbf{T}^{-1}\mathbf{v}$ in the original image.

✔ In case of non-integer value, perform interpolation (nearest neighbor or bilinear)



# Interpolation




**Nearest neighbor method:**

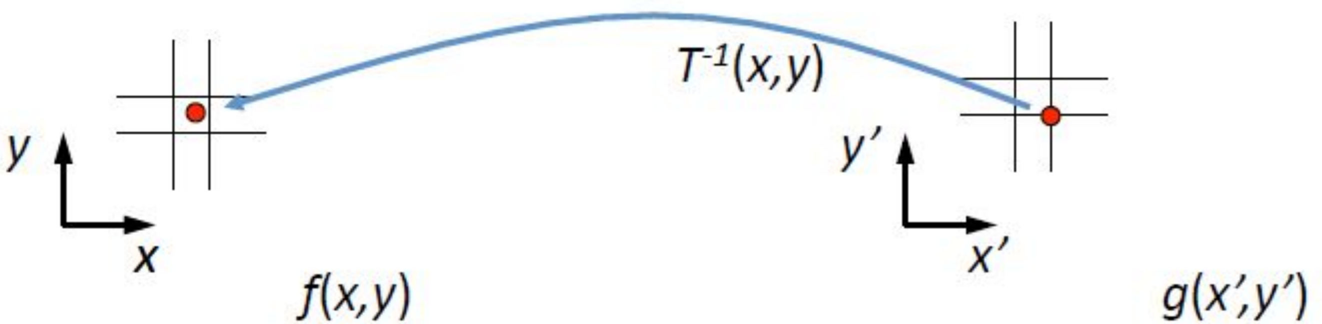
- Use value $a_4$ (as the pixel that is nearest to the red point contains value $a_4$)

**Bilinear method:**

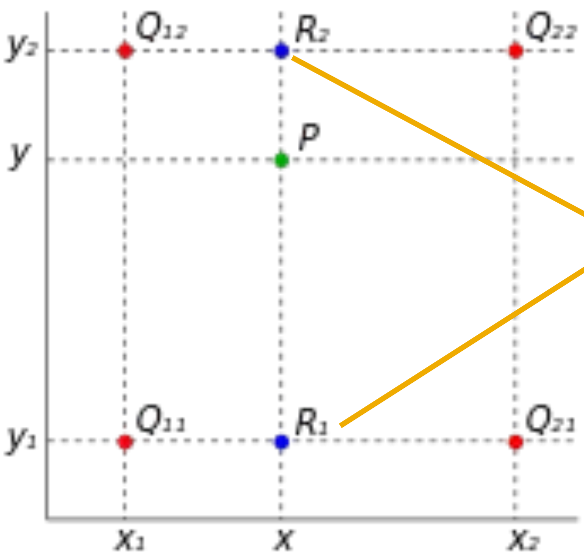
- Use the following value, a weighted combination of the four neighboring pixel values, with more weight to nearer values:

$$(Ba_4 + Aa_3 + Ca_1 + Da_2)/(A+B+C+D) = (Ba_4 + Aa_3 + Ca_1 + Da_2)$$

as $A+B+C+D=1$ for unit area pixels



# Bilinear interpolation in more detail




We interpolate first in the X direction:

$$f(x, y_1) = \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21})$$

$$f(x, y_2) = \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22})$$

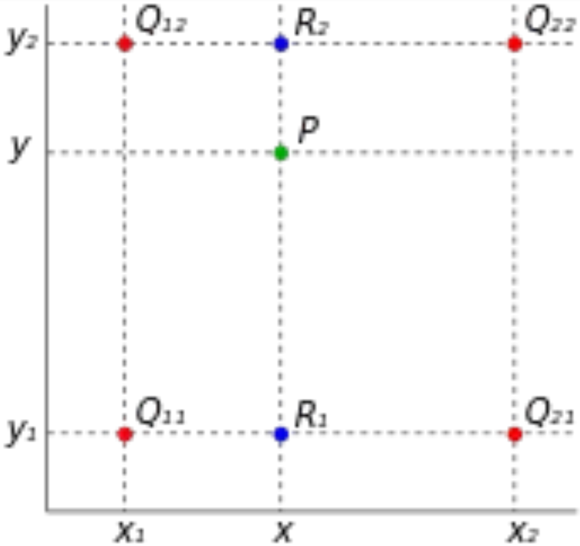
We then interpolate first in the Y direction:

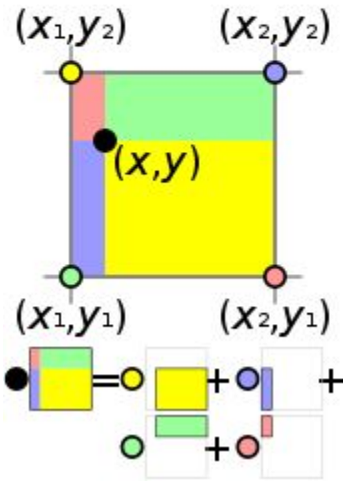
$$f(x, y) = \frac{y_2 - y}{y_2 - y_1} f(x, y_1) + \frac{y - y_1}{y_2 - y_1} f(x, y_2)$$

$$= \frac{y_2 - y}{y_2 - y_1} \left( \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \right) + \frac{y - y_1}{y_2 - y_1} \left( \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \right)$$

In both cases, notice that higher weight is given to the pixels that are closer to P.

# Bilinear interpolation in more detail




$$f(x,y) = \frac{y_2-y}{y_2-y_1}\left(\frac{x_2-x}{x_2-x_1}f(Q_{11})+\frac{x-x_1}{x_2-x_1}f(Q_{21})\right)+\frac{y-y_1}{y_2-y_1}\left(\frac{x_2-x}{x_2-x_1}f(Q_{12})+\frac{x-x_1}{x_2-x_1}f(Q_{22})\right)$$

$$= \frac{y_2-y}{y_2-y_1}\frac{x_2-x}{x_2-x_1}f(Q_{11})+\frac{y_2-y}{y_2-y_1}\frac{x-x_1}{x_2-x_1}f(Q_{21})+\frac{y-y_1}{y_2-y_1}\frac{x_2-x}{x_2-x_1}f(Q_{12})+\frac{y-y_1}{y_2-y_1}\frac{x-x_1}{x_2-x_1}f(Q_{22})$$

This formula will remain unchanged if you first interpolated in the Y direction and then in the X direction. Verify this yourself.

# Bilinear interpolation in more detail

- Here we are approximating the image intensity in the form of the following bilinear function:

$$f(x,y) = a_0 + a_1 x + a_2 y + a_3 xy$$

- Here $a_0$, $a_1$, $a_2$, $a_3$ are scalar coefficients. The value of f(x,y) is known at the four corners of a pixel.

- The function would have been linear if the term in xy were absent (i.e. if $a_3 = 0$).

# Bilinear interpolation in more detail

- How do we determine the coefficients $a_0, a_1, a_2, a_3$ ?

$$\begin{pmatrix} 1 & x_1 & y_1 & x_1y_1 \\ 1 & x_1 & y_2 & x_1y_2 \\ 1 & x_2 & y_1 & x_2y_1 \\ 1 & x_2 & y_2 & x_2y_2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} f(x_1, y_1) \\ f(x_1, y_2) \\ f(x_2, y_1) \\ f(x_2, y_2) \end{pmatrix} = \begin{pmatrix} f(Q_{11}) \\ f(Q_{12}) \\ f(Q_{21}) \\ f(Q_{22}) \end{pmatrix}$$

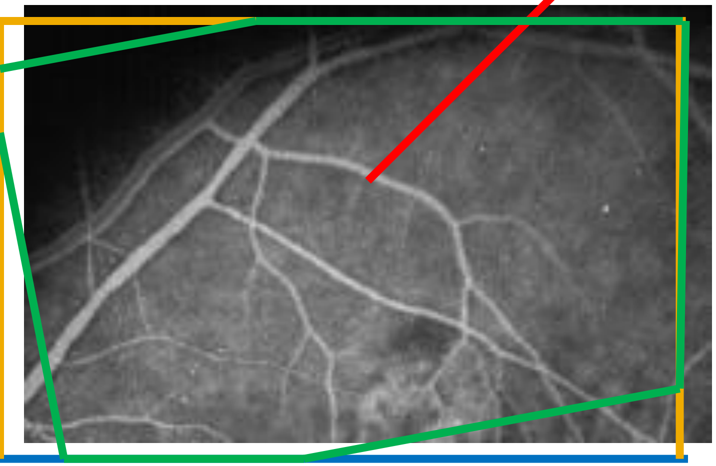
- These coefficients can be obtained by inverting the 4 x 4 matrix.
- It can be shown (through tedious calculations) that the result of this is equivalent to the formula we derived earlier.

# Alignment with mean squared error

- In the ideal case, the MSSD between two perfectly aligned images is 0. In practice, it will have some small non-zero value even under more or less perfect alignment due to sensor noise or slight mismatch in pixel grids.

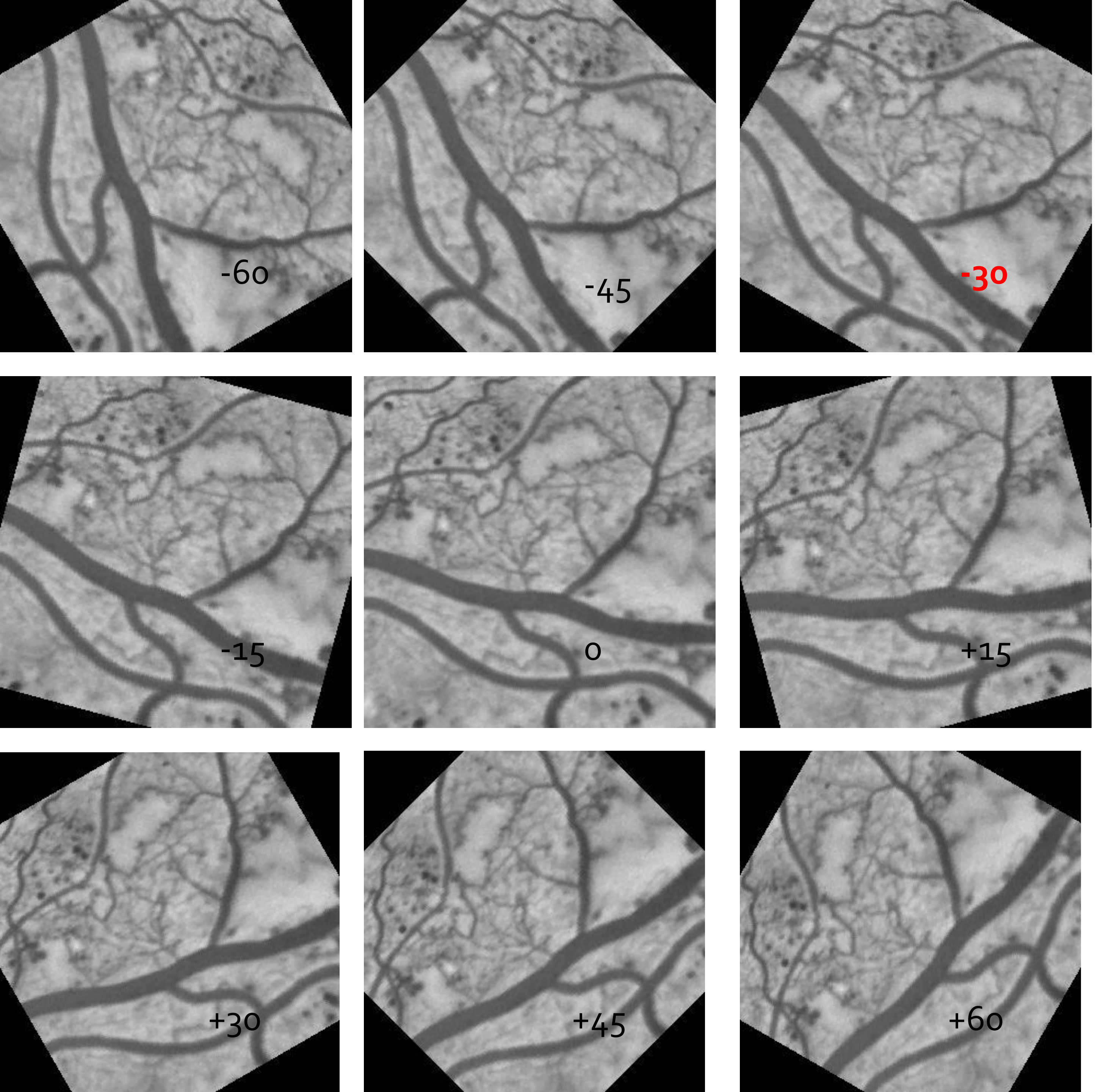
# Careful: field of view issues!

Fixed image (also called reference image)

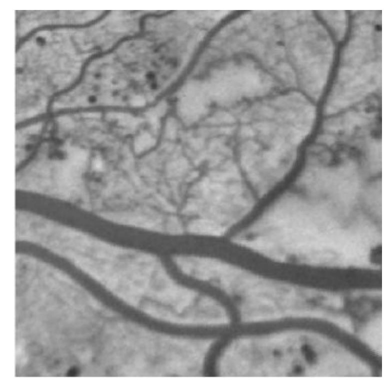


Region of overlap (also called **field of view**) when the moving image is warped

Note: compute MSSD only over region of overlap.




Change in region of overlap (also called **field of view**), as the moving image is warped. The field of view in any of these figures consists of all those pixels not marked black. It represents the set of pixels (x,y) where both $I_1(x,y)$ and $I_2(x,y)$ are well defined.

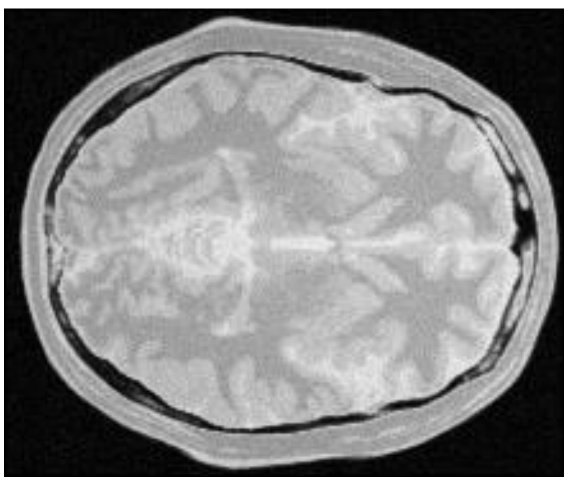


# Alignment with mean squared error

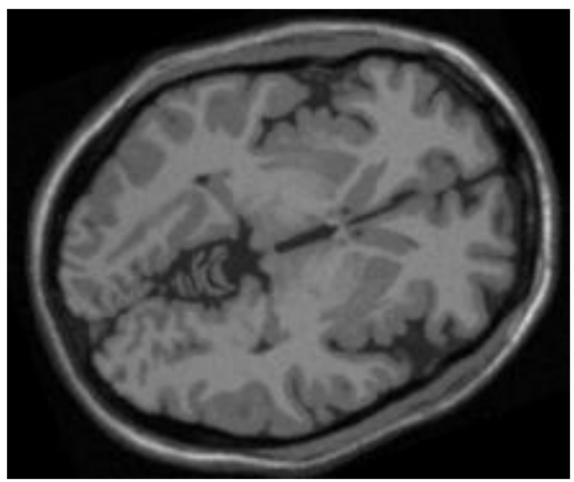
- MSSD is called an “image similarity metric”.
- MAJOR ASSUMPTION: Physically corresponding pixels have the same intensity!

# Image alignment: Intensity changes in images

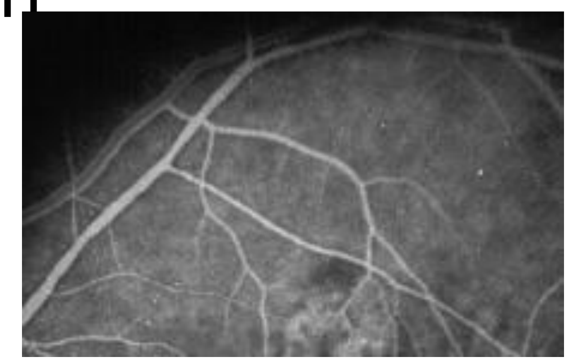
- Images acquired by different sensors (MR and CT, camera with and without flash, etc.)
- Changes in lighting condition

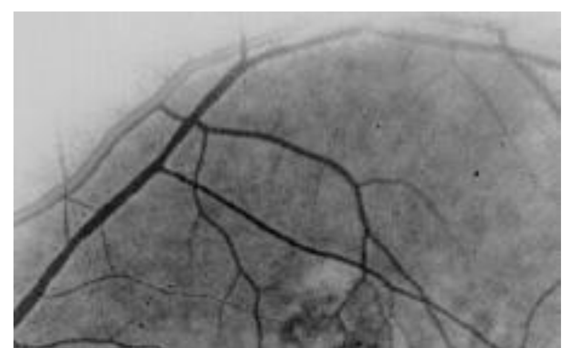


MR-PD



MR-T1





# Image alignment: Intensity changes in images

- If following relationship (function g) exists and we **knew** it, the solution is easy:

$$I_1(x_1, y_1) = g(I_2(x_2, y_2)), \forall (x_1, y_1), (x_2, y_2) \in \Omega$$

Physically corresponding points

$$\text{transformed} - \text{MSSD} = \frac{1}{N} \sum_{x,y \in \Omega} (g(I_2(x, y)) - I_1(x, y))^2$$

$N = \#\,\text{pixels in the field of view (see earlier for definition of field of view)}$

# Image alignment: Intensity changes in images

- What if the relationship exists in the following linear form, but we knew it only partially?

$$I_1(x_1, y_1) = aI_2(x_2, y_2) + b, \forall (x_1, y_1), (x_2, y_2) \in \Omega$$

Physically corresponding points

$$NCC = \left| \frac{\sum_{(x,y)\in\Omega} (I_1(x,y) - \bar{I}_1)(I_2(x,y) - \bar{I}_2)}{\sqrt{\sum_{(x,y)\in\Omega} (I_1(x,y) - \bar{I}_1)^2 \sum_{(x,y)\in\Omega} (I_2(x,y) - \bar{I}_2)^2}} \right|$$

Normalized cross-correlation, also called correlation-coefficient

$\bar{I}_1, \bar{I}_2$ : average value of images $I_1, I_2$

We are taking the absolute value here, to take care of cases where one image has positive values and the other has negative values. We are assuming a linear relationship between the intensities of I1 and I2 but we do not assume knowledge of the scalar coefficients *a* and *b*.

# Image alignment: Intensity changes in images

$$NCC = \left| \frac{\sum_{(x,y)\in\Omega} (I_1(x,y) - \bar{I}_1)(I_2(x,y) - \bar{I}_2)}{\sqrt{\sum_{(x,y)\in\Omega} (I_1(x,y) - \bar{I}_1)^2 \sum_{(x,y)\in\Omega} (I_2(x,y) - \bar{I}_2)^2}} \right|$$

$\bar{I}_1, \bar{I}_2$ : average value of images $I_1, I_2$

Normalized cross-correlation, also called correlation-coefficient.
The NCC is like the absolute normalized dot product between mean-deducted images.

$$\boldsymbol{T}^* = \arg\max NCC_{\boldsymbol{T}}(I_1(\boldsymbol{v}), I_2(T\boldsymbol{v}))$$

$$\boldsymbol{T} = \begin{pmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{pmatrix}, \boldsymbol{v} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

# Image alignment: intensity changes in images?

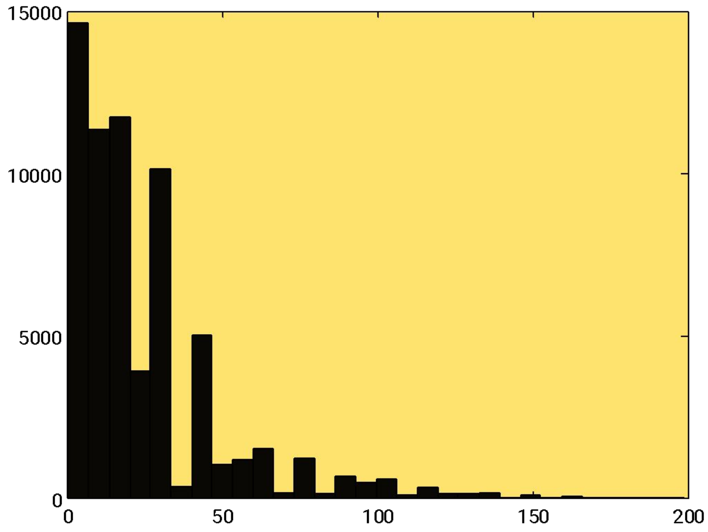
- Assume there exists a functional relationship between intensities at physically corresponding locations in the two images.

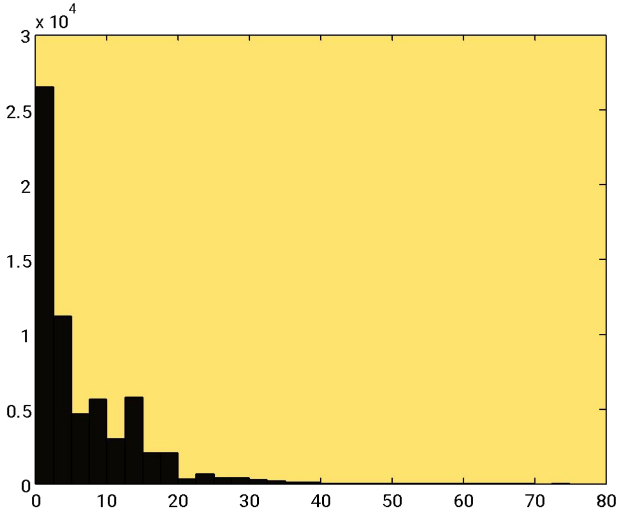
- But suppose we didn't know it (most practical scenario) and couldn't find it out.

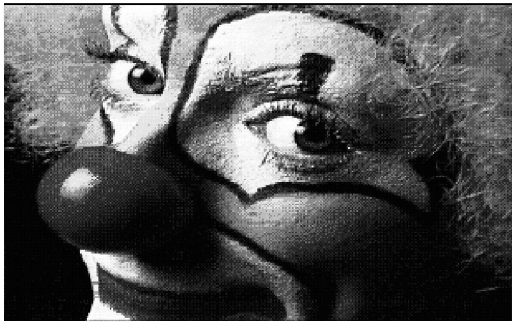
- We will use image histograms!

# Image Histogram

- In a typical digital image, the intensity levels lie in the range [0,$L$-1].
- The (normalized) histogram of the image is a discrete function of the form $p(r_k) = n_k/HW$, where $r_k$ is the $k$-th intensity value, and $n_k$ is the number of pixels with that intensity. (H,W = image dimensions)
- Sometimes, we may consider a range of intensity values for one entry in the histogram, in which case $r_k = [r^{min}_k, r^{max}_k]$ represents an intensity bin, and $n_k$ is the number of pixels whose intensity falls within this bin.
- Note $p(r_k) >= 0$ always, and all the $p(r_k)$ values sum up to 1.



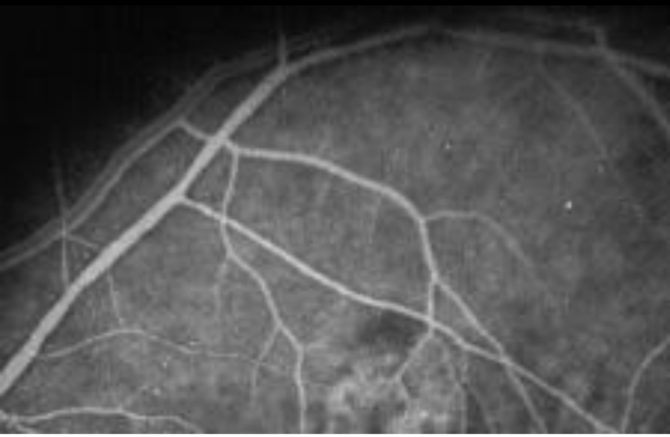



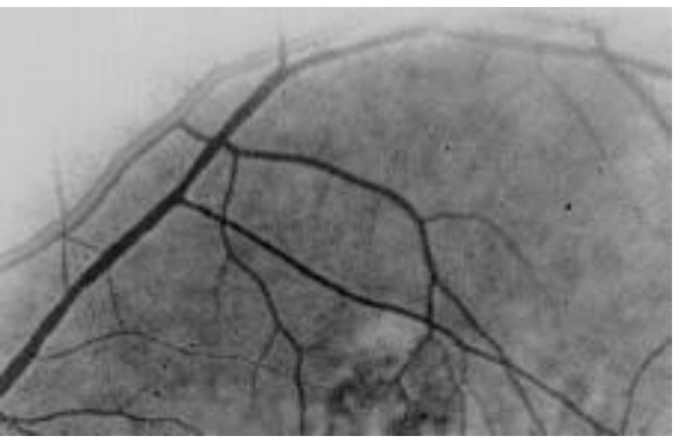
These are unnormalized histograms of two different images. That is, the entries of the histogram represent frequencies in this case without division by HW = #pixels.

# Joint Image Histogram

- A joint image histogram is a function of the form $p(r_{k1}, r_{k2})$ where $r_{k1}$ and $r_{k2}$ represent intensity bins from the two images $I_1$ and $I_2$ respectively.

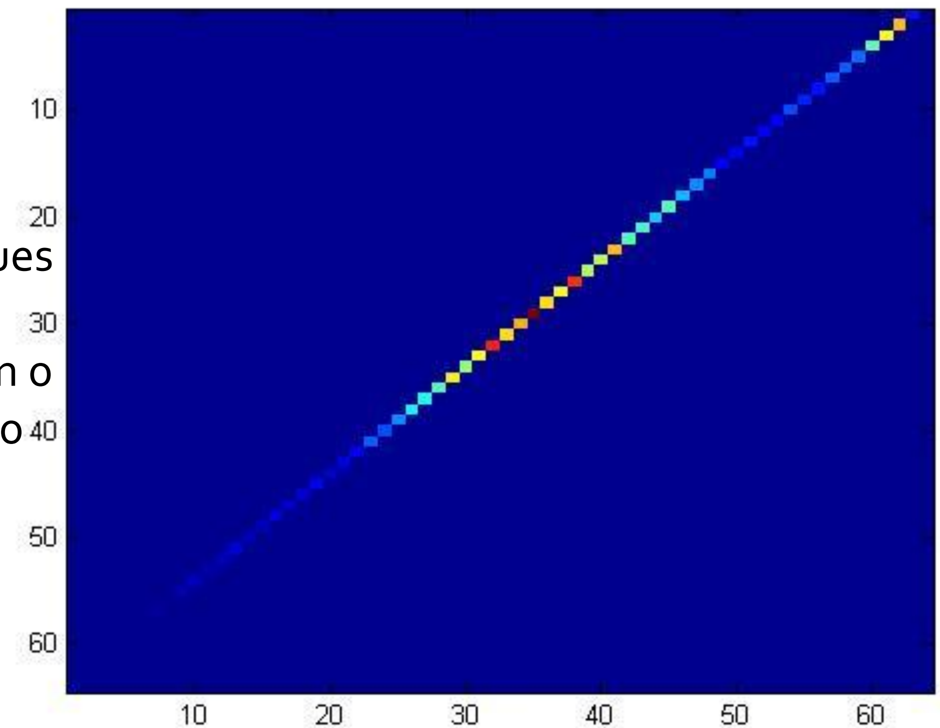
- $p(r_{k1}, r_{k2})$ = number of pixels $(x,y)$ such that $I_1(x,y)$ and $I_2(x,y)$ lie in bins $r_{k1}$ and $r_{k2}$ respectively, divided by HW.





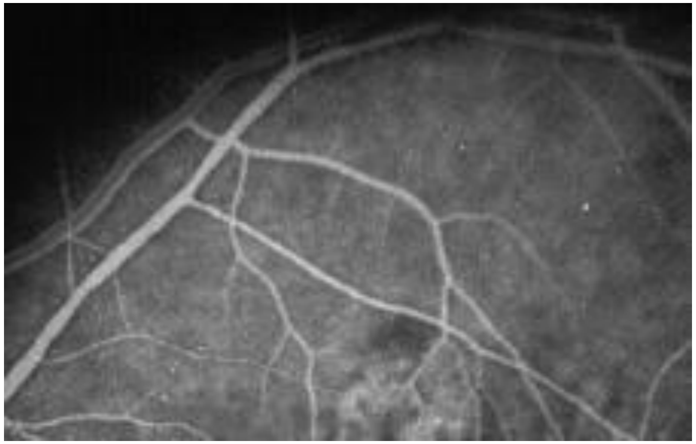
$I_2$

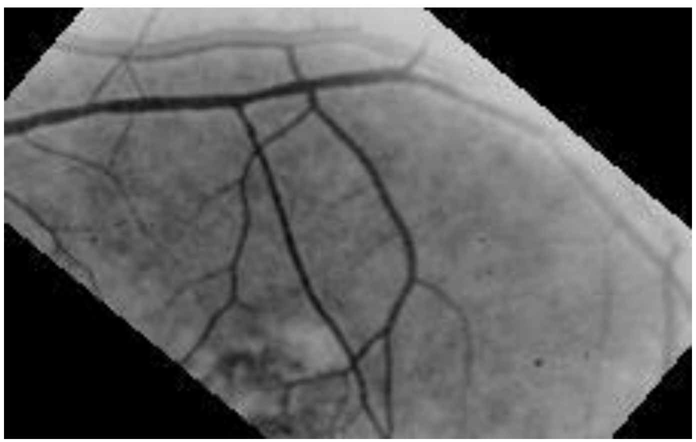
Values in $I_2$ from 0 to 60

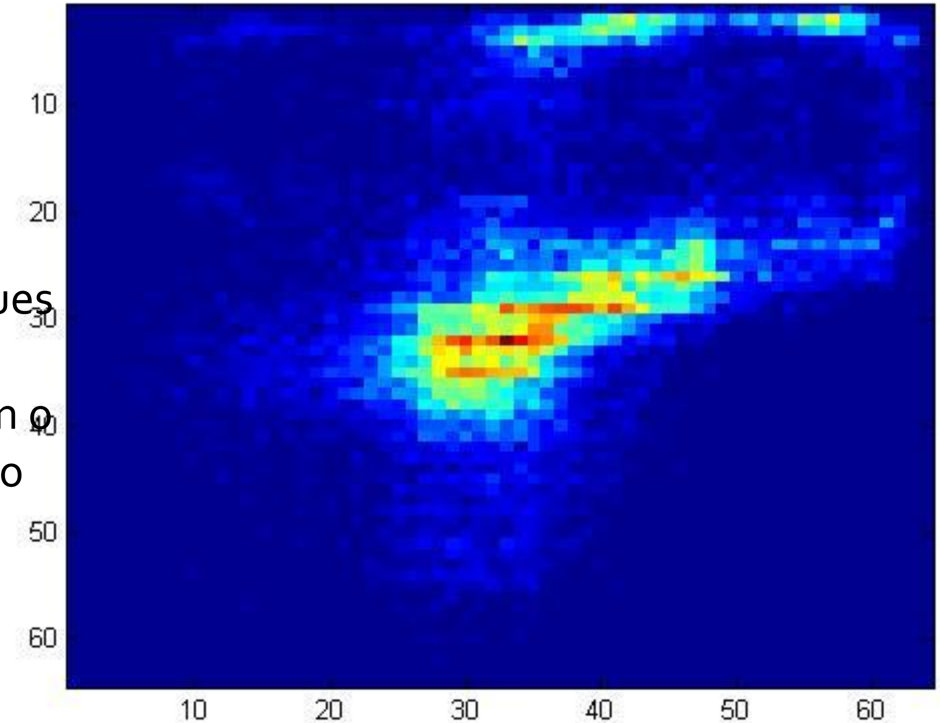



Values in $I_1$ (from 0 to 60)

Registered images: joint histogram plot looks “streamlined”








Values in $I_2$ from 0 to 60

Values in $I_1$ (from 0 to 60)

Misaligned images: joint histogram plot looks “dispersed”

We need a method to quantify how dispersed a joint histogram actually is.

# Measure of dispersion

- Consider a discrete random variable X with normalized histogram p(X) [also called the probability mass function].
- The entropy of X is a measure of the uncertainty in X, given by the following formula:

$$H(X) = -\sum_{x \in DX} p(X = x) \log_2 p(X = x)$$

$$DX = \text{discrete set of values that } X \text{ can take}$$

- Note: entropy is a function of the **normalized histogram** of X.
- **Not** a function of the **actual values** of X.
- The entropy is always non-negative.

# Entropy

- The entropy is maximum if X has a discrete uniform distribution, i.e. $p(X=x_1) = p(X=x_2)$ for all values $x_1$ and $x_2$ in DX. The maximum value is log(|DX|).

- The entropy is minimum (zero) if the normalized histogram of X is a Kronecker delta function, i.e. $p(X= x_1) = 1$ for some $x_1$, and $p(X= x_2) = 0$ for all $x_2 \neq x_1$.

# Joint entropy

- The joint entropy of two random variables X and Y is given as follows:

$$H(X,Y) = -\sum_{x \in DX} \sum_{y \in DY} p(X = x, Y = y) \log_2 p(X = x, Y = y)$$

- Maximum entropy:

-Uniform distribution on X and Y: entropy value log(|DX||DY|) where DX, DY are the set of discrete values that X and Y can acquire

- Minimum entropy:

-Kronecker delta (i.e. a PMF will all probability concentrated on only one entry with others being 0): entropy value 0 = non uncertainty

# Joint entropy

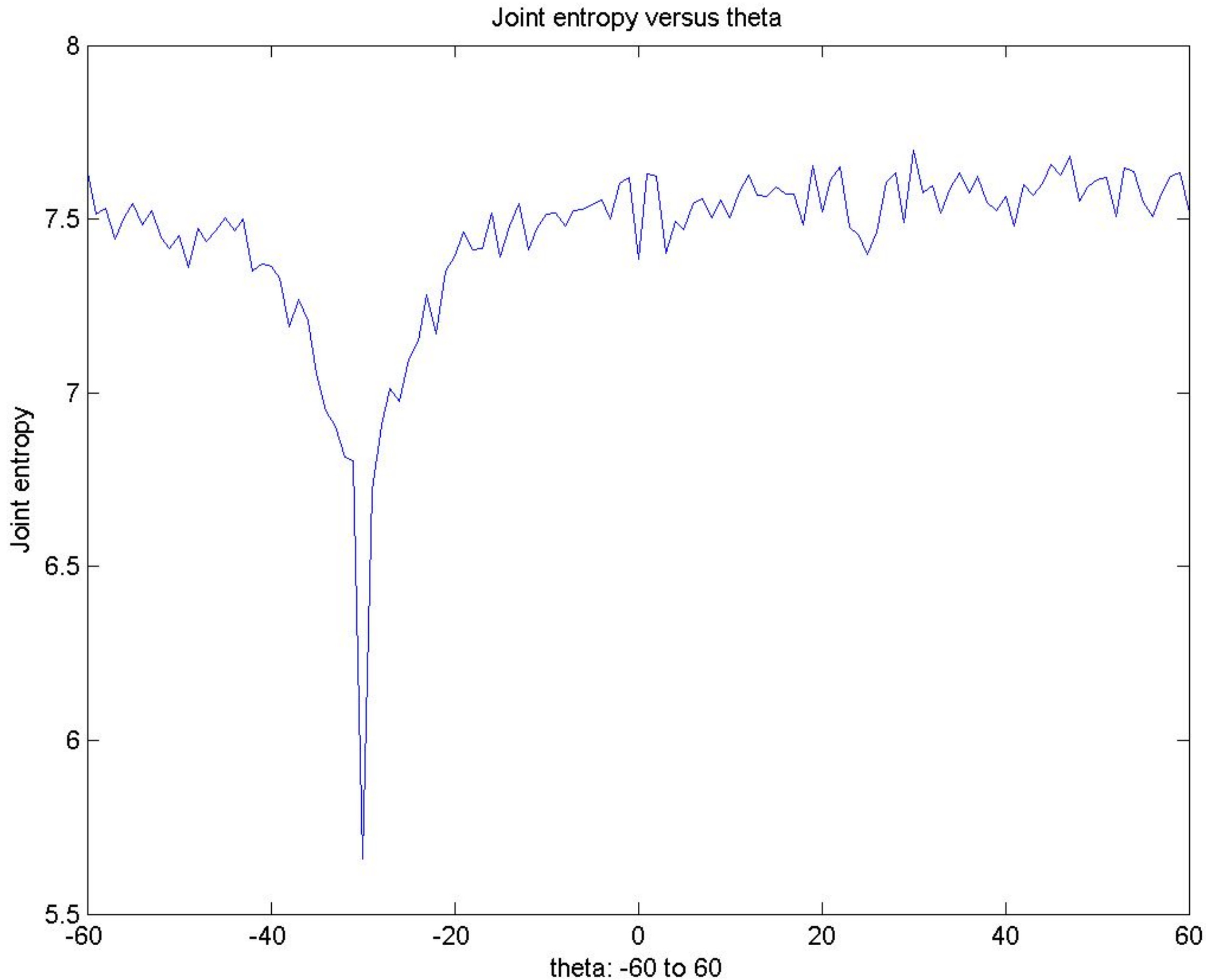
- Minimizing joint entropy is one method of aligning two images with different intensity profiles.

$$\boldsymbol{T}^* = \arg\min_{\boldsymbol{T}} H(I_1(\boldsymbol{v}), I_2(\boldsymbol{T}\boldsymbol{v}))$$

$$\boldsymbol{T} = \begin{pmatrix} A_{11} & A_{12} & t_x \\ A_{21} & A_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix}, \boldsymbol{v} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

I1

I2: obtained by squaring the intensities of I1, and rotating I1 anticlockwise by 30 degrees.




I2 treated as moving image, I1 treated as fixed image. Joint entropy minimum occurs at -30 degrees.

# Components of an Image Alignment Algorithm

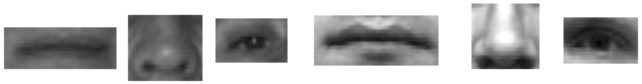
- Choice of a metric to optimize (joint entropy or mean squared error)
- Choice of a motion model (only translation, translation + rotation, affine, etc.)
- Choice of an interpolation algorithm to generate the warped image
- Choice of an optimization algorithm (here, we just used brute force search)

# Image Alignment: applications, related problems

- Template matching
- Image Panoramas

# Template Matching

- Look for the occurrence of a template (a smaller image) inside a larger image.
- Example: eyes within face image

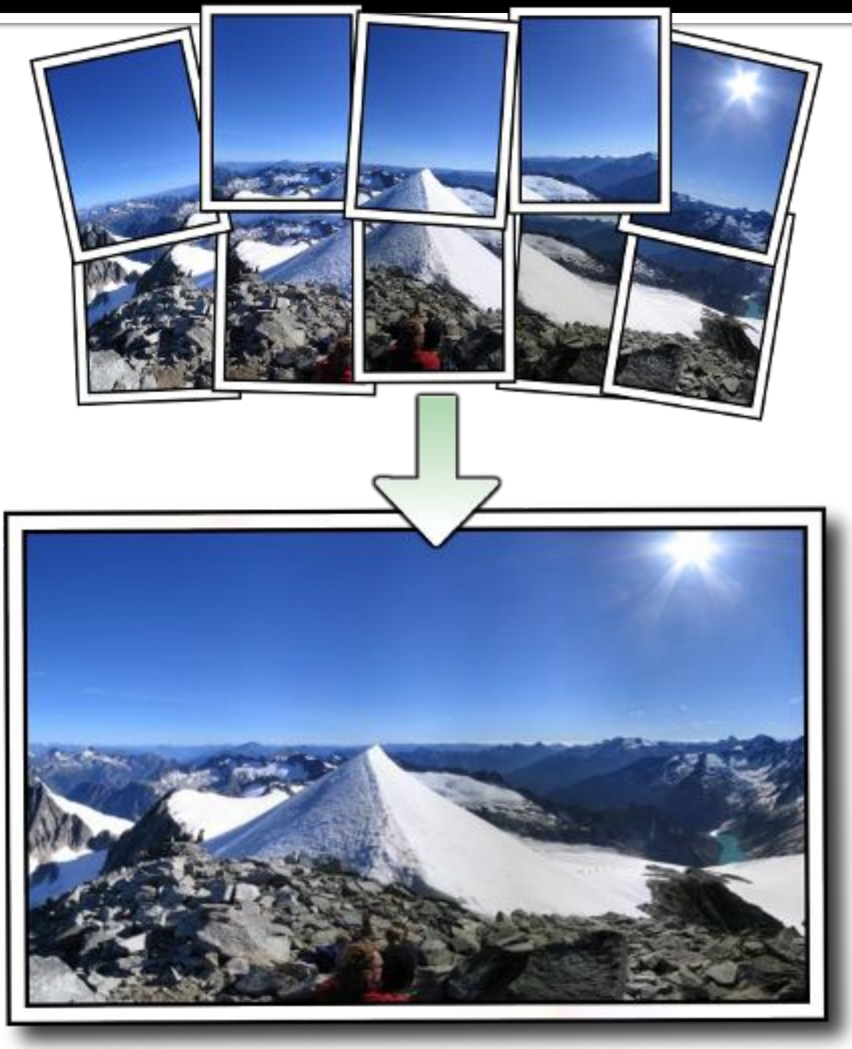


Templates

# Template matching

- Let T = template image (smaller) of size h x w and J = larger image of size H x W
- For every pixel (x,y) in J, consider a portion $z_{xy}$ = J(x:x+w-1,y:y+h-1)
- Select the portion $z_{xy}$ with smallest MSSD compared to T
- In some variants, the larger image J may be rotated with respect to T.
- In such cases, repeat the above procedure for every rotation of J from (say) -90 to +90 degrees.
- That is for every θ from -90 to +90 degrees, let $J_\theta$ be a rotated version of J. Now consider all $z_{xy}$ in $J_\theta$ and report the (x,y, θ) triple that produces the least MSSD with respect to T.

# Image Panoramas



http://cs.bath.ac.uk/brown/autostitch/autostitch.html

A camera has a limited field of view. A scene may have much larger "area" than what can be captured from a single camera.

So one can acquire multiple images, each from a different viewpoint, and you need to stitch these together to form a panorama or mosaic. The stitching involves more complicated motion models (called homography) than what you have studied in this course.

# What we learnt..

- Affine motion model
- Forward and reverse image warping
- Field of view during image alignment
- Measures for Image alignment: sum of squared differences, normalized cross-correlation, joint entropy

# What we didn't learn

- Complicated motion models: higher degree polynomials, non-rigid models (example: motion of an amoeba, motion of the heart during the cardiac cycle, facial expressions, etc.)

- Efficient techniques for optimizing the measure for image alignment