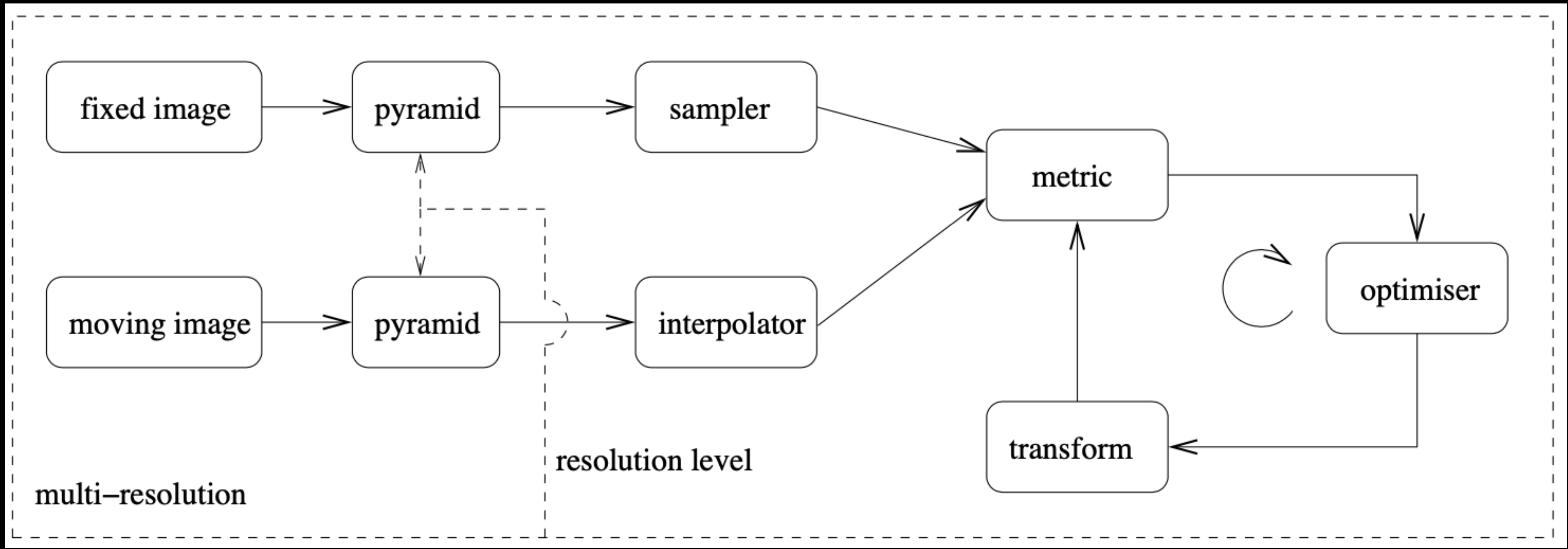
Image Analysis

Tim B. Dyrby
Rasmus R. Paulsen
DTU Compute

tbdy@dtu.dk

http://www.compute.dtu.dk/courses/02503

Lecture 10 – Advanced image registration



Klein et al 2010. (IEEE Trans Med Img)

https://elastix.lumc.nl

What can you do after today?

- Describe difference between a pixel and voxel
- Choose a general image-to-image registration pipeline
- Apply 3D geometrical affine transformations
- Define coordinate system of an object for 3D rotation
- Use the Homogeneous coordinate system to combine transformations
- Compute a suitable intensity-based similarity metric given the image modalities to register
- Compute the normalized correlation coefficient (NNC) between two images
- Compute Entropy
- Describe the concept of iterative optimizers
- Compute steps in the gradient descent optimization algorithm
- Apply the pyramidal principle for multi-resolution strategies
- Select a relevant registration strategy: 2D to 3D, Within- and between objects and moving images

Go to **www.menti.com** and use the code **4414 1532**

Associations to a mountain view

Mount Everest - Himalayas

0	0	0	0	0
A) Skiing	B) Hiking	C) paragliding	D) Danger	E) A parameter space

Image Registration pipeline

- The input images
  - Fixed image: Reference image
  - Moving image: Template image

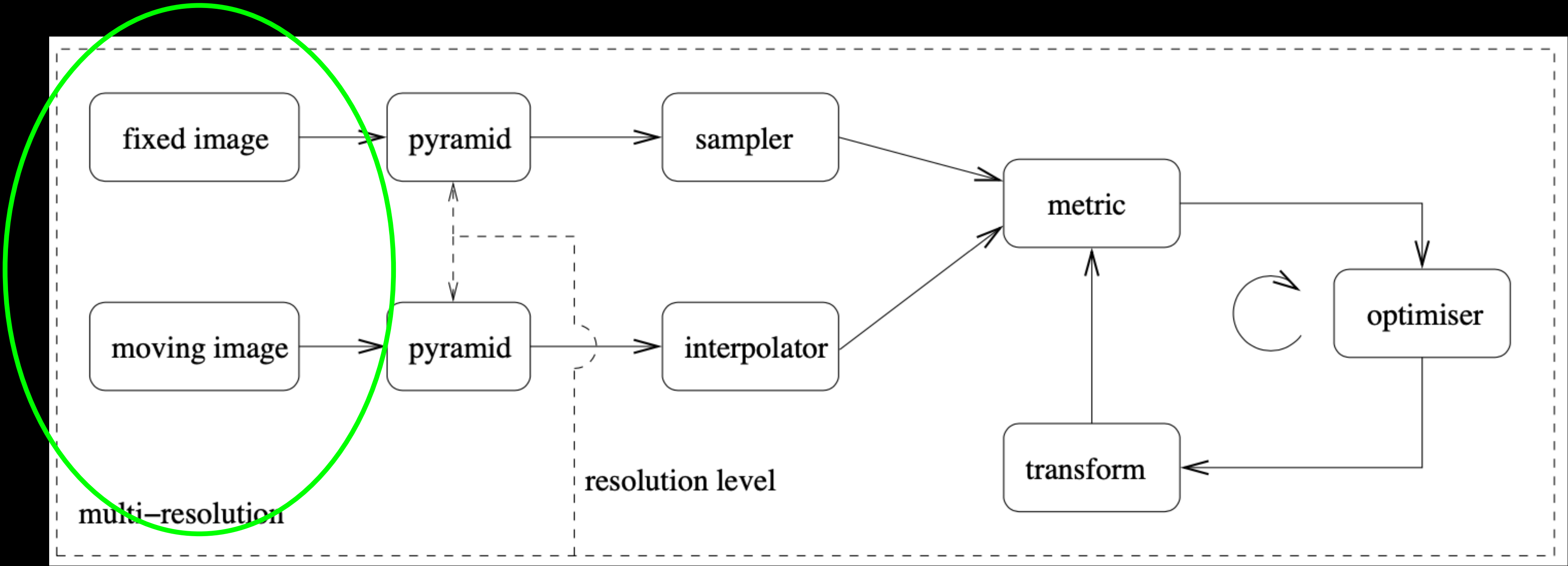
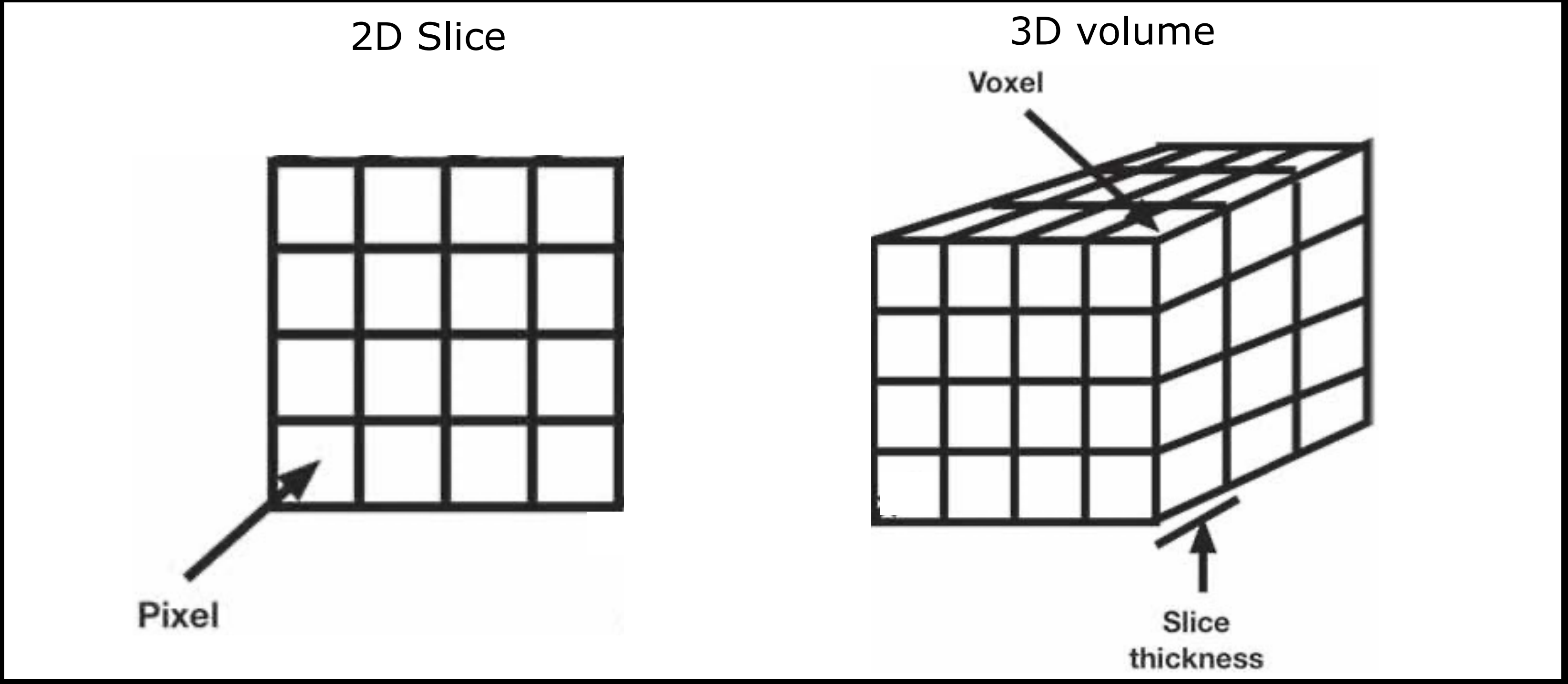


Image volumes

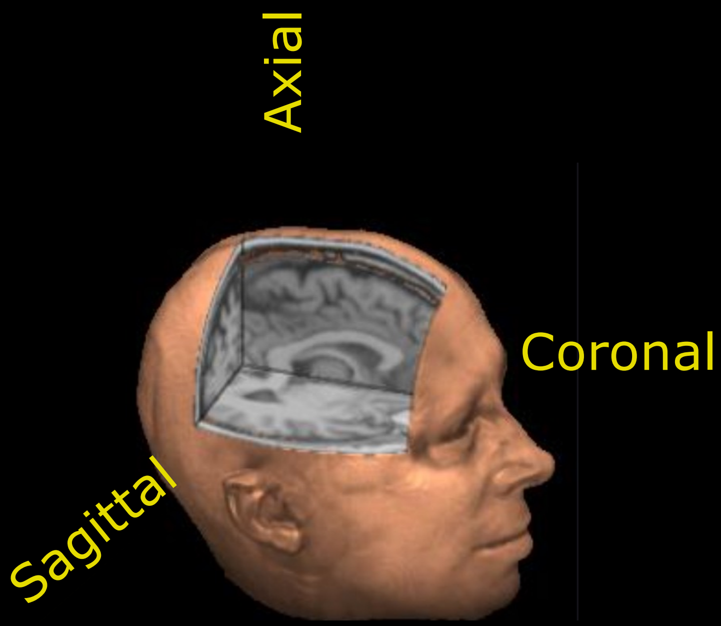
- Image slice: 2D (NxM) matrix of pixels
- Image volumes: 3D (NxMxP) matrix of voxels
  - An element is a volume pixel i.e. voxel
- Pixel vs voxel intensity
  - Integrated information within an area or volume



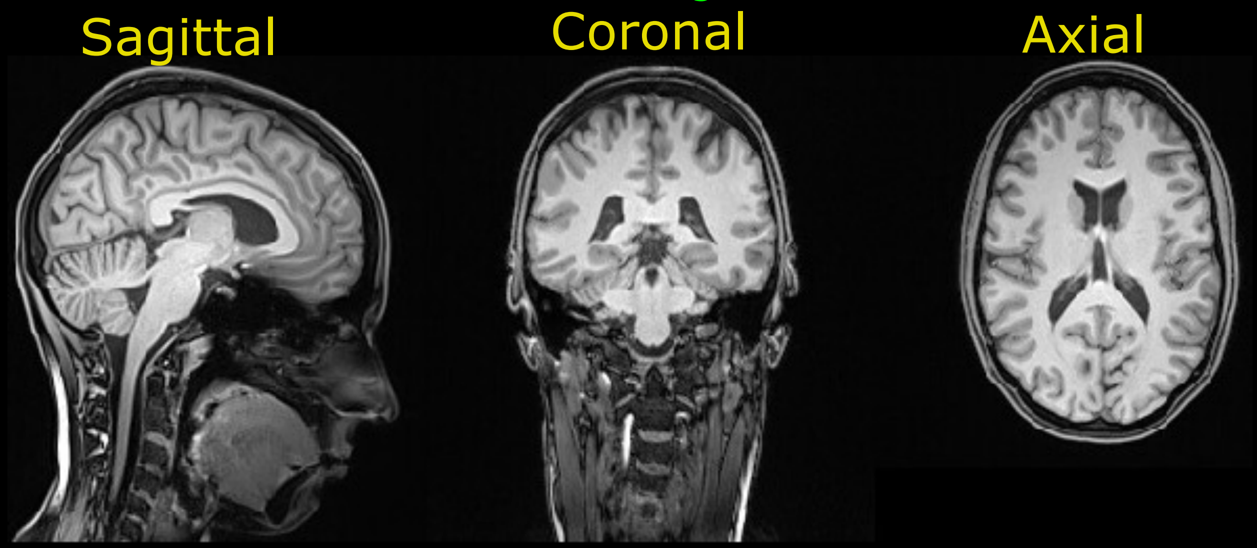
3D image viewing

- Three orthogonal views
  - Fine structural details at slice level
  - Hard to get 3D surface insight
- Rendering of surfaces
  - Surface insight
  - Limited types of surfaces to visualise

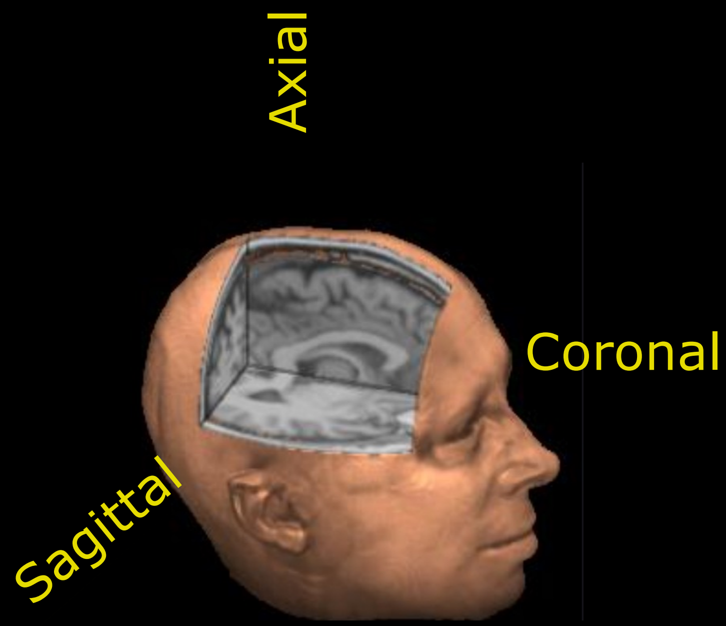
3D rendering



Slices three orthogonal views



3D rendering



3D image viewing

- Three orthogonal views
  - Fine structural details at slice level
  - Hard to get 3D surface insight
- Rendering of surfaces
  - Surface insight
  - Limited types of surfaces to visualise

Slices three orthogonal views

www.dreamstime.com/illustration/truck-top-view.html

Image volumes

- Stacked slices: 2D to 3D
  - Object cut into slices, imaged and stacked
  - Still pixels – not voxel
- Registration challenges
  - Geometrical distortions between slices

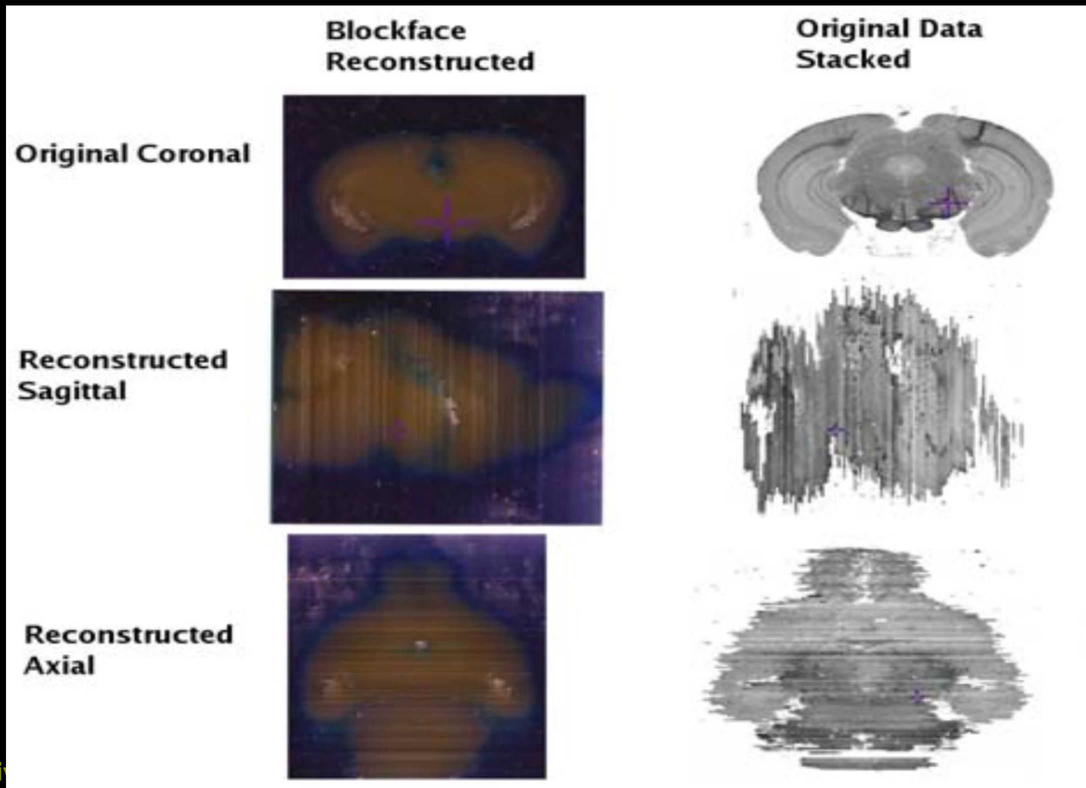
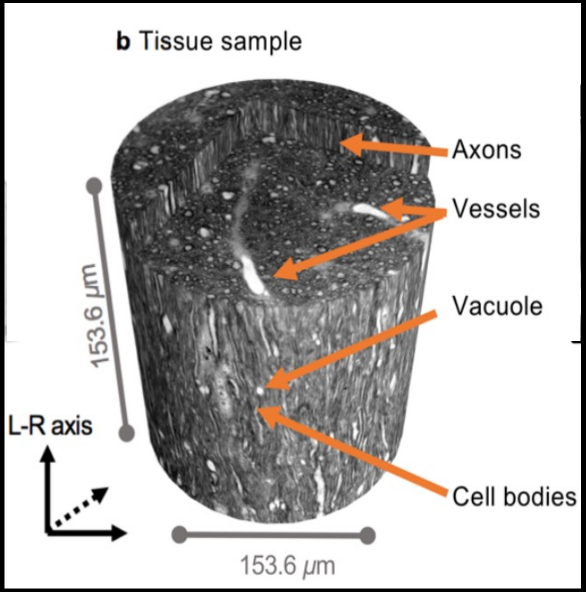


Image volumes

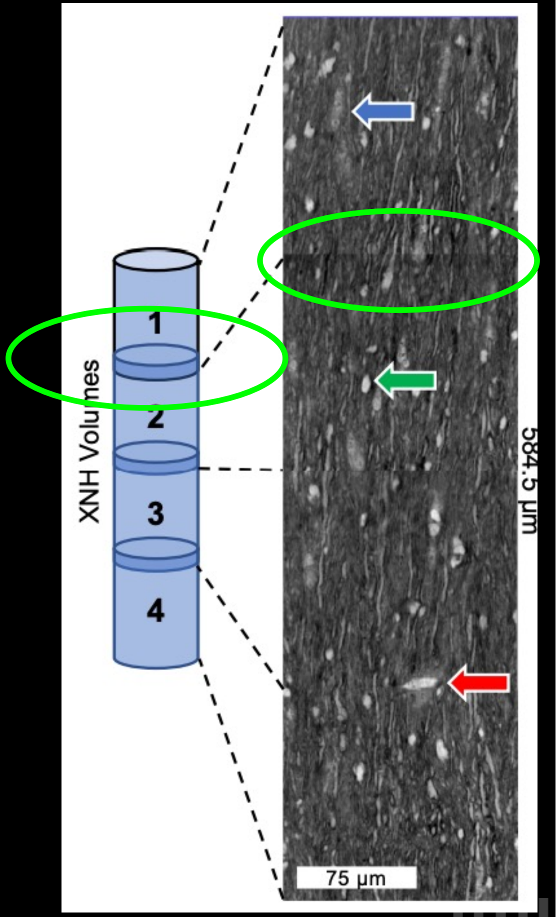
- Intact sample
  - No sample cutting
- Registration challenges:
  - Stacking 3D volumes

Synchrotron x-ray imaging
Tissue sample 1mm
75 nm isotropic resolution voxels

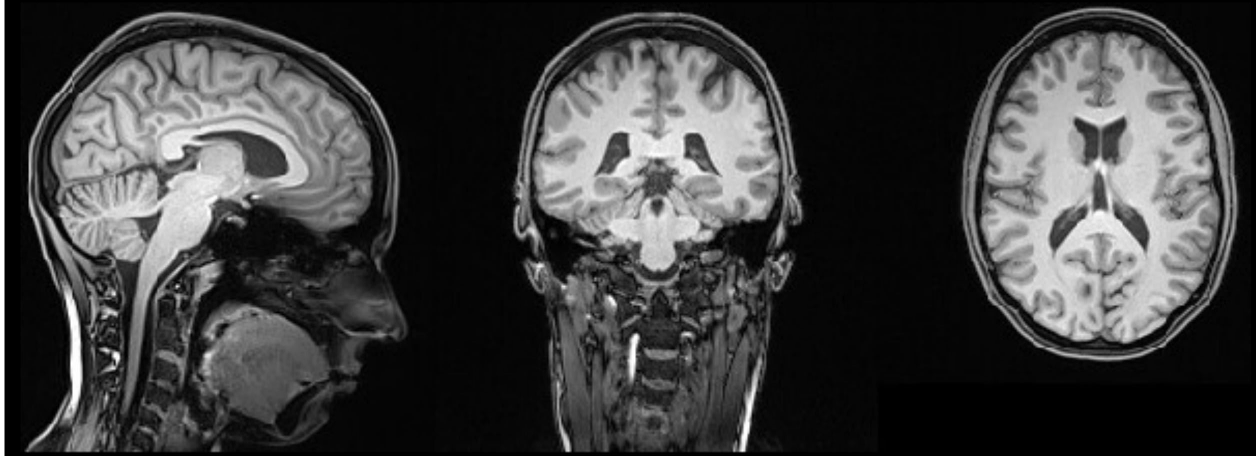


Andersson et al, 2020 (PNAS)

Stacked 3D volumes



MRI
Whole brain
1 mm isotropic resolution voxels



Rotating sample in x-ray tomography

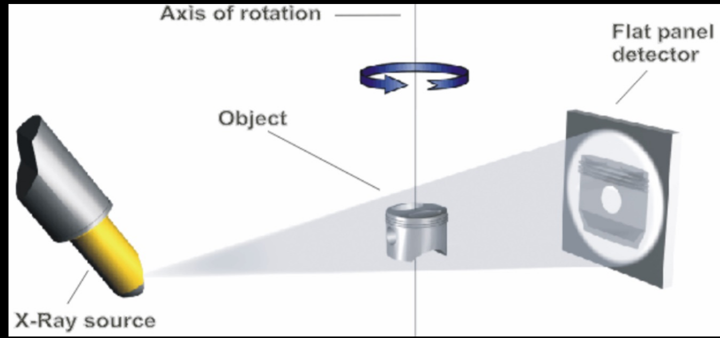
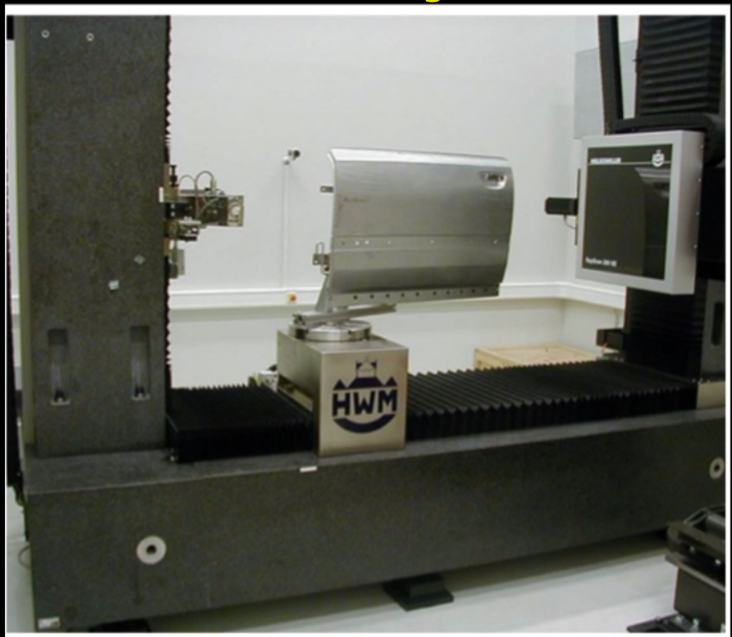


Image volumes

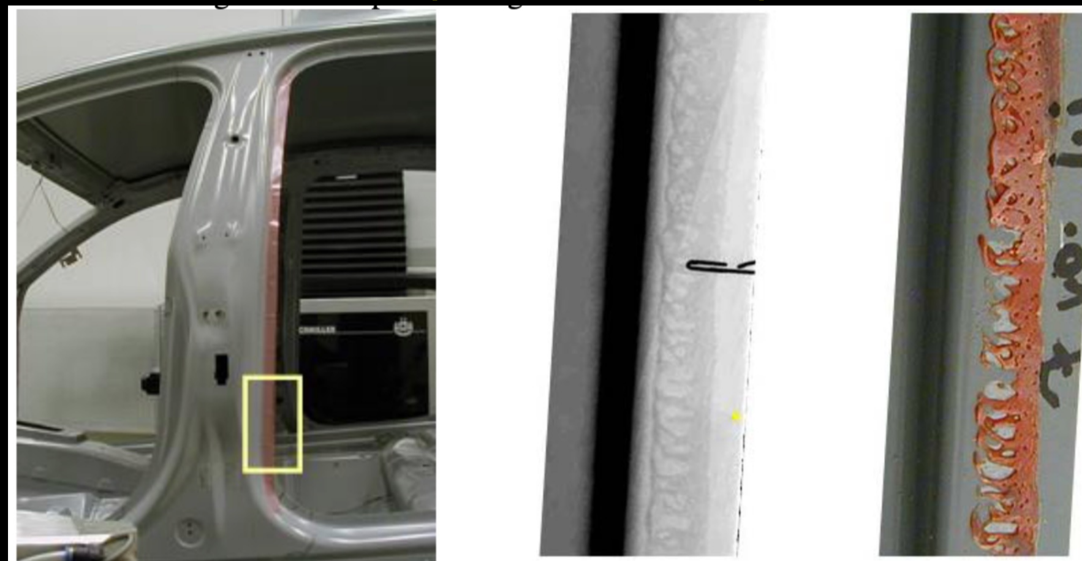
- Intact sample
  - No sample cutting
- Registration challenges:
  - Multi image resolution: Fit Region-of-interest image to whole object image

CT scanning



Car door AUDI A8, size: 1150 mm

Region of interest (ROI) | CT of ROI (non-destructive) | Microscope (destructive)

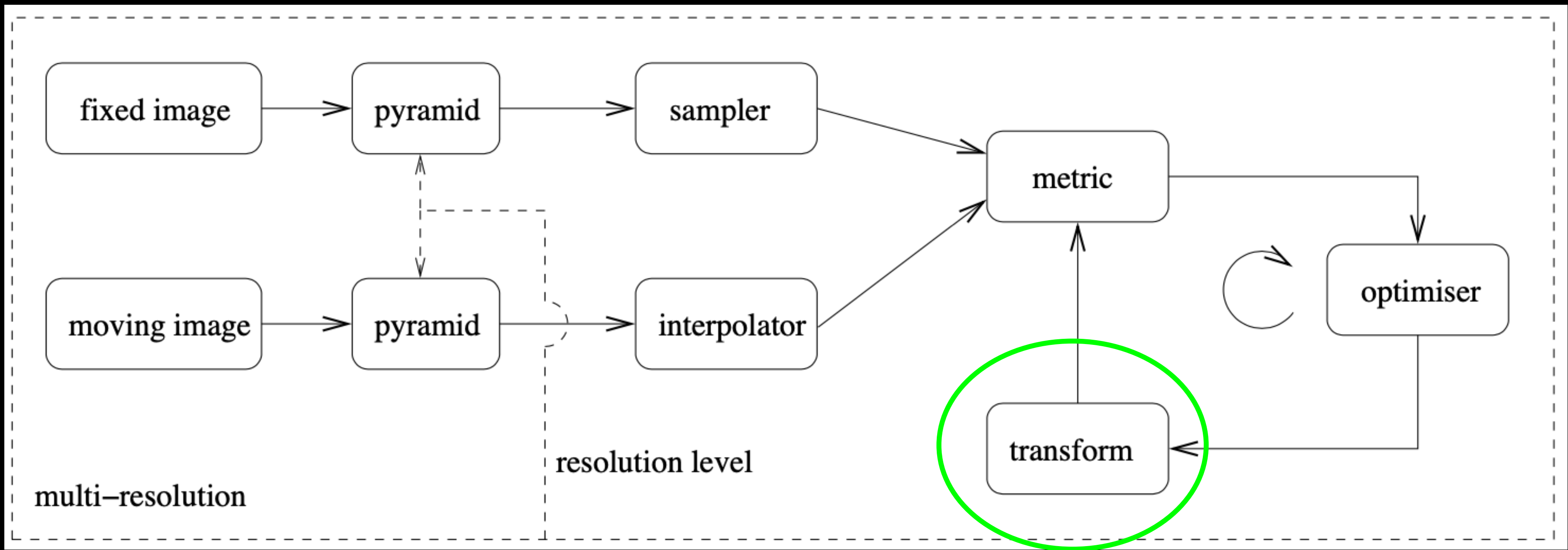


The inspection of a glued joint of a car body

Simon et al, 2006 (ECNDT)

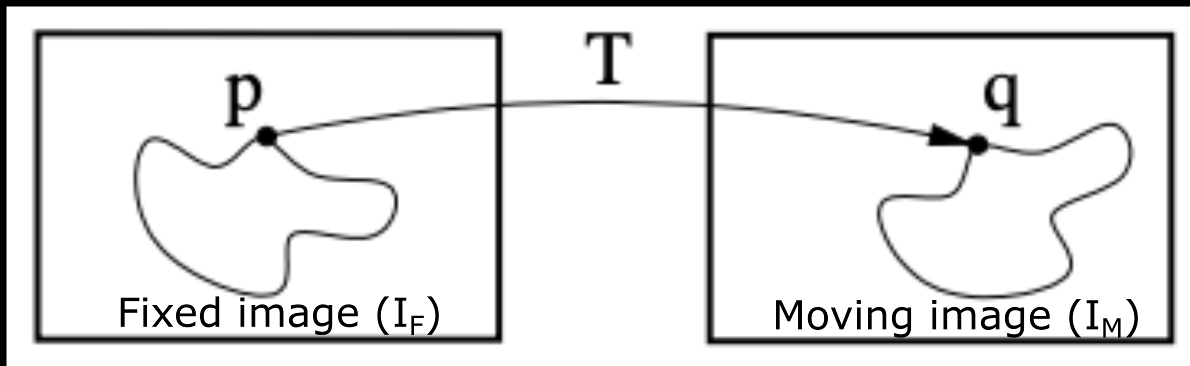
Image Registration pipeline

- Geometrical transformations



Geometric transformations

- Translation
- Rotation
- Scaling
- Shearing



(Reference image) (Template image)

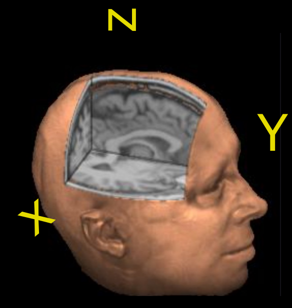
$$\hat{T} = \arg\min_{T} C(T; I_F, I_M)$$

Translation 2D vs 3D

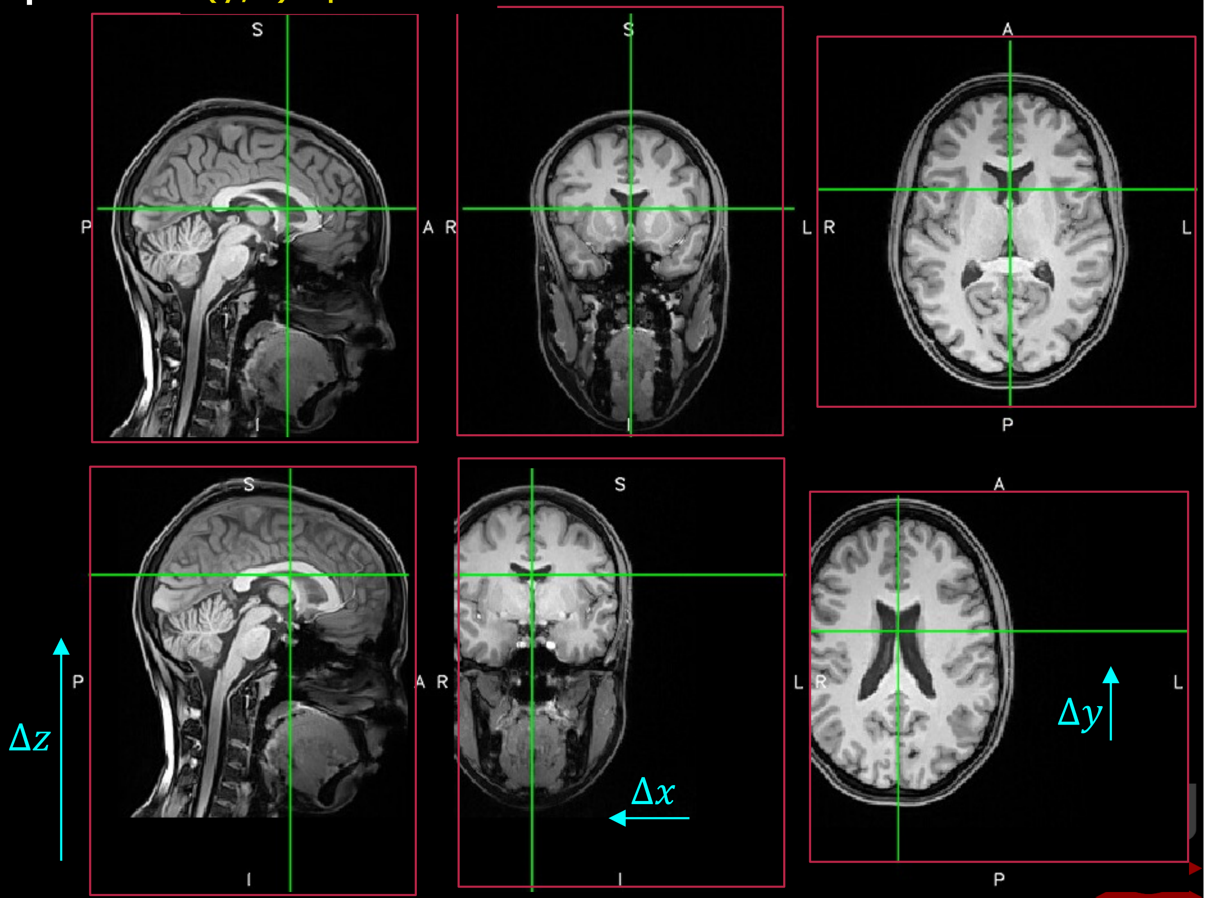
- The image is shifted
  - 2D: Inspect one slice plan
  - 3D:Inspect three slice plans

3D: (x,y,z)-plans

$$\begin{bmatrix}\Delta x\\ \Delta y\\ \Delta z\end{bmatrix} = -\begin{bmatrix}60\\ 20\\ 15\end{bmatrix}$$

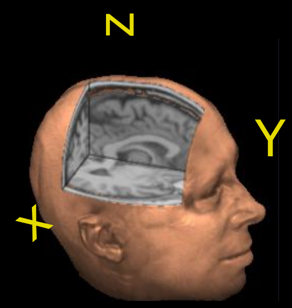


(y,z) -plan (x,z)-plan (x,y)-plan



2D: (x,y)-plan

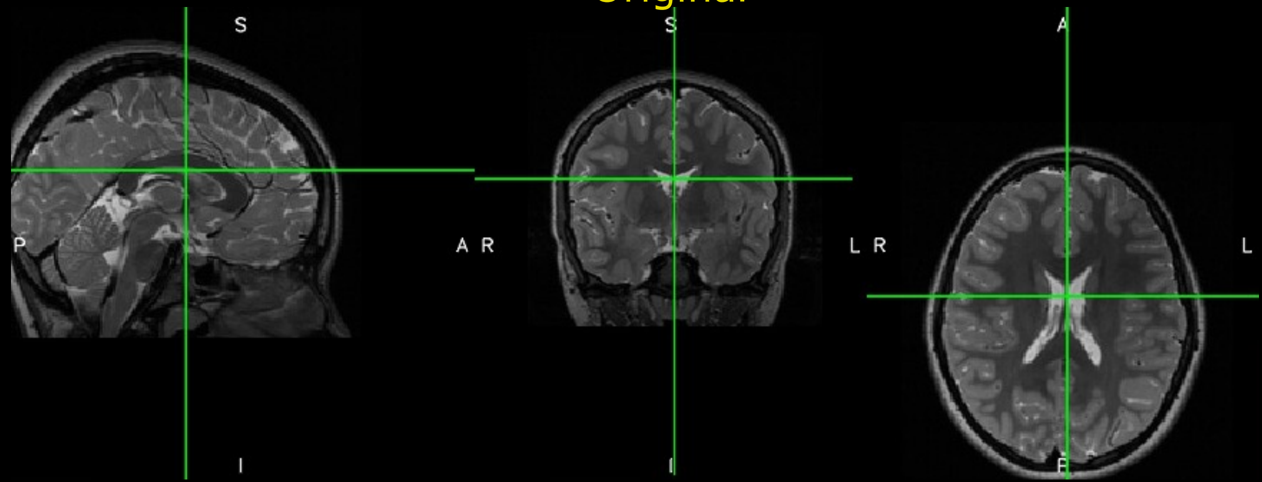
$$\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix} = \begin{bmatrix}60\\ 20\end{bmatrix}$$



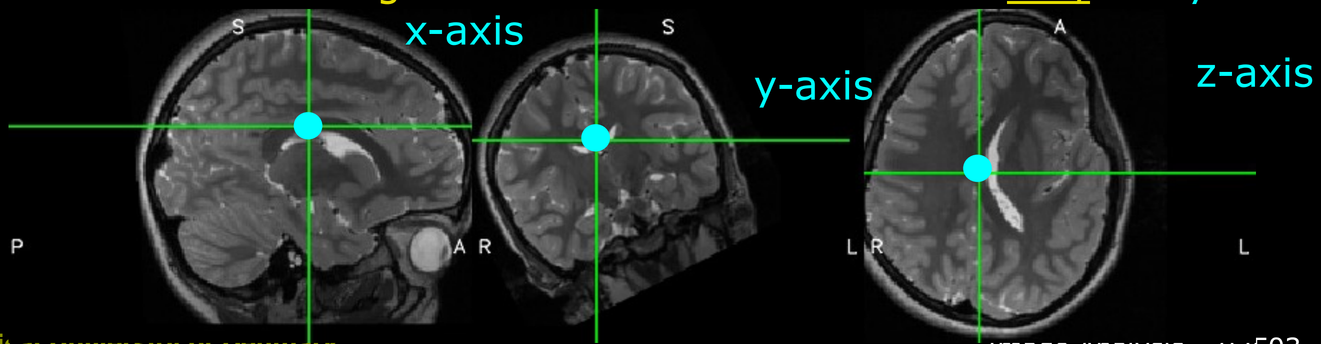
Rotation 3D

- The image is rotated around an origin (e.g. the centre-of-mass)
- Rotate the object around three axis hence three angles.
  - Inspect all three views to identify a rotation

Original



Rotated: 27 degree counter-clockwise around <u>only</u> the y-axis

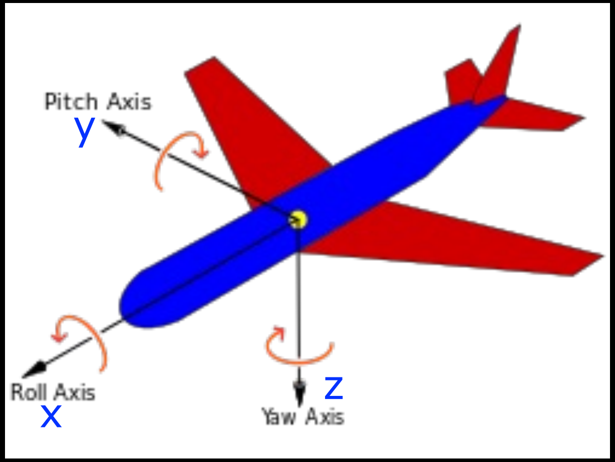


3D Rotation coordinate system

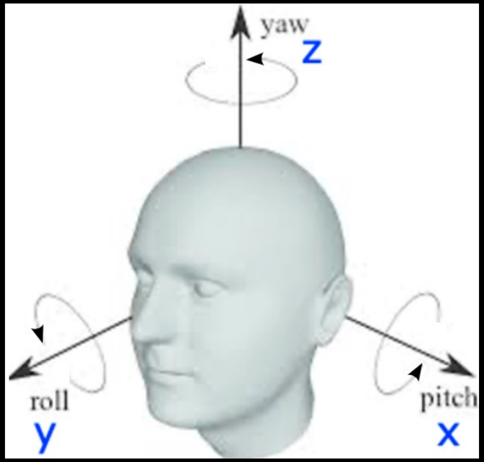
- Three element rotations round the axes of the coordinate system
- Pitch, Yaw and Roll
  - Note: Definition of the coordinate system is object specific

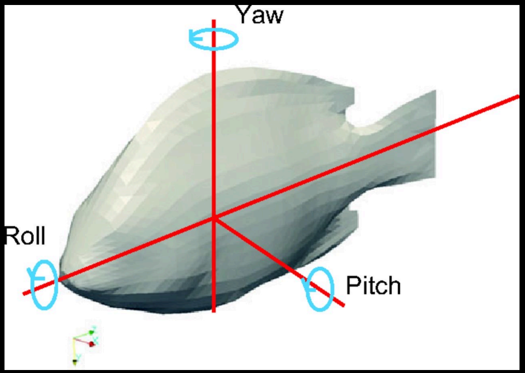
Rotation rules

- Counter clock-wise rotations: Right-hand rule (as in figures) ← We use here
- Clock-wise rotations: Left-hand rule



The principal axes of an aircraft according to the air norm DIN 9300

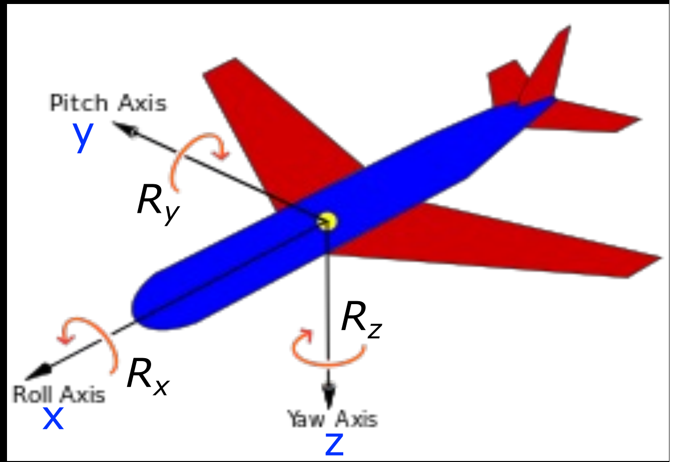




3D Rotation coordinate system

- Axis-Angle representation
- Three composed element rotations
  - Angles: α, β, γ
  - Counter clock-wise rotations (Right-hand rule)
- The order matters
  - Several Euler-angle conventions exist
- Remember: Know your origin!

Axis-Angle representation

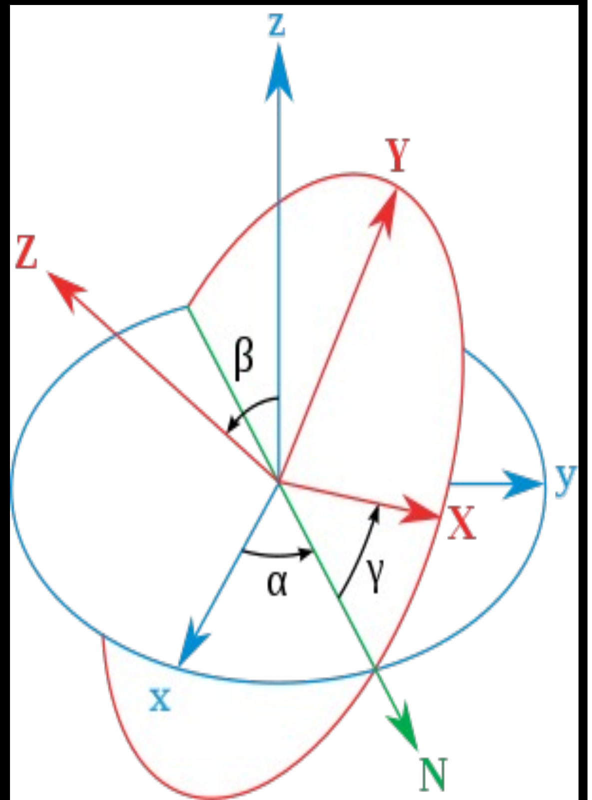


$$R_X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \quad R_Y = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \quad R_Z = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Roll Pitch Yaw

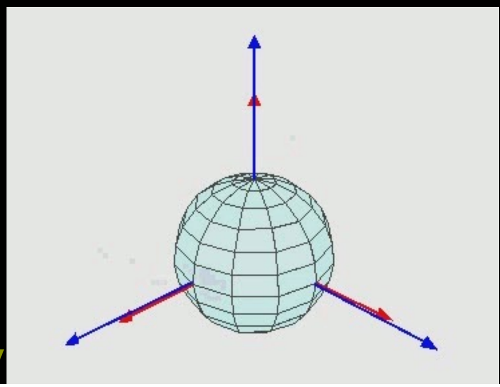
Euler convention - example

- The intrinsic ZXZ-Euler angle convention (uses the right-hand rule):
  - α: Around the z-axis. Defines the line of nodes (N)
  - β: Around the new X-axis defined by N
  - γ: Around the new Z-axis from N
- The order of coordinate system rotations:
  - Rotation order around the:
  - z-axis: Initial: Original frame (x,y,z): α
  - *New X-axis: First coordinate system rotation (X,Y,Z):* β
  - *New Z-axis: Second coordinate system rotation (X,Y,Z):* γ



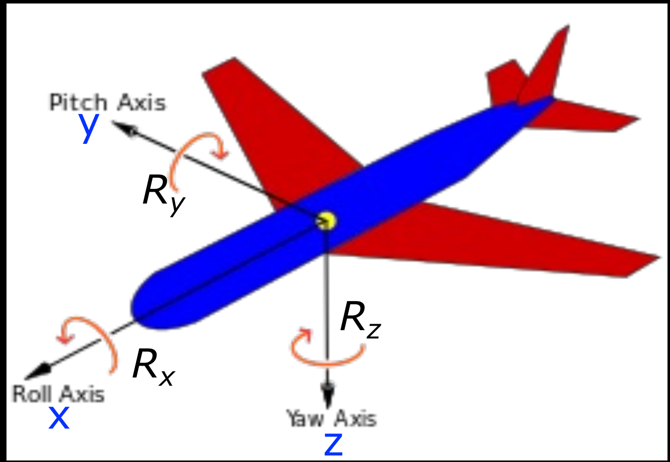
wikipedia.org/wiki/Euler_angles

$$A_R = R_Z(\gamma) * R_x(\beta) * R_Z(\alpha)$$



Euler convention – example for a flight

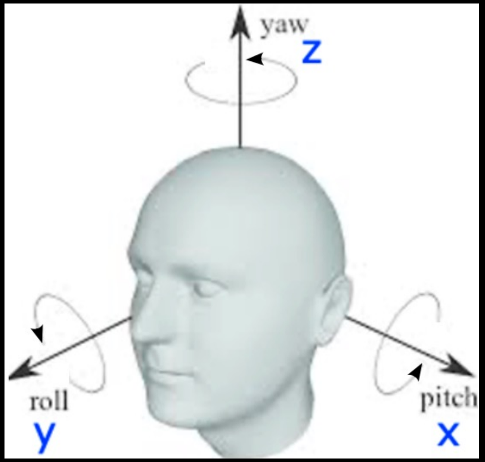
- The Yaw-Pitch-Roll Euler angle convention (use the right-hand rule)
- Use defined coordinate system for the object
- Rotation order of a flight:
  - *Yaw: rotation around the Z-axis*
  - *Pitch: Rotation around the Y-axis*
  - *Roll: Rotation around the X-axis*



$$A_R = R_z(\gamma) * R_y(\beta) * R_x(\alpha)$$

Euler convention – example for a head

- The Yaw-Pitch-Roll Euler angle convention (use the right-hand rule)
- Use defined coordinate system for the object
- Rotation order of a human head:
  - *Yaw: rotation around the Z-axis*
  - *Pitch: Rotation around the X-axis*
  - *Roll: Rotation around the Y-axis*



$$A_R = R_z(\gamma) * R_x(\beta) * R_y(\alpha)$$

Quiz 1: Affine 3D transformation

How many parameters?

A) 6
B) 5
C) 16
D) 12
E) 3

SOLUTION:

Translation: P=3

Rotation: p=3

Scaling: p=3

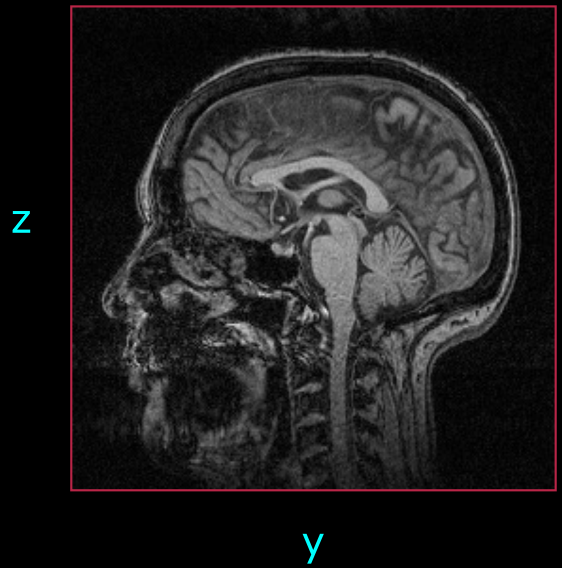
Shearing: p=3

Scaling in 3D

- The size of the image is changed
- Three parameters:
  - X-scale factor, S_x
  - Y-scale factor, S_y
  - Z-scale factor, S_z

$$A = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & Sz \end{bmatrix}$$

- Isotropic scaling:



$$A = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{bmatrix}$$

Shearing in 3D

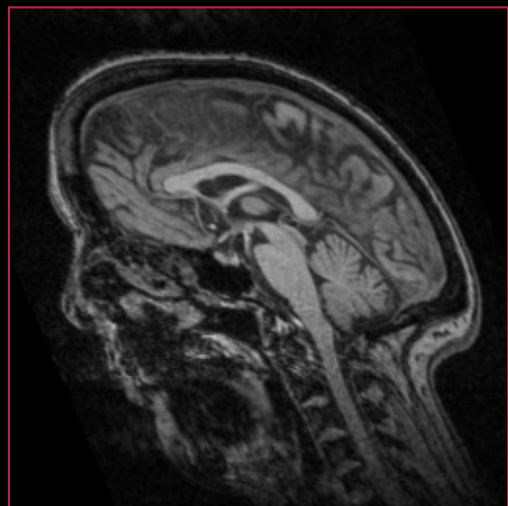
- Pixel shifted horizontally or/and vertically
- Three parameters

$$A = \begin{bmatrix} 1 & Syx & Szx \\ Sxy & 1 & Syz \\ Sxz & Syz & 1 \end{bmatrix}$$

z

y

Shearing (z,y)-plan



Combining transformations

Translation:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} + \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Rotations, Scaling, Shear:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = A \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

- Translation is a *summation* i.e. P'=A+P
- Rotation, Scale, Shear are *multiplications* i.e. P'=A*P
- Wish: To combine transformations via multiplications:

$$A = A_T * AR * A_{shear} * A_s$$

- Not possible with A_T

Homogeneous coordinates

Cartesian coordinates:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = A \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Homogeneous coordinates:

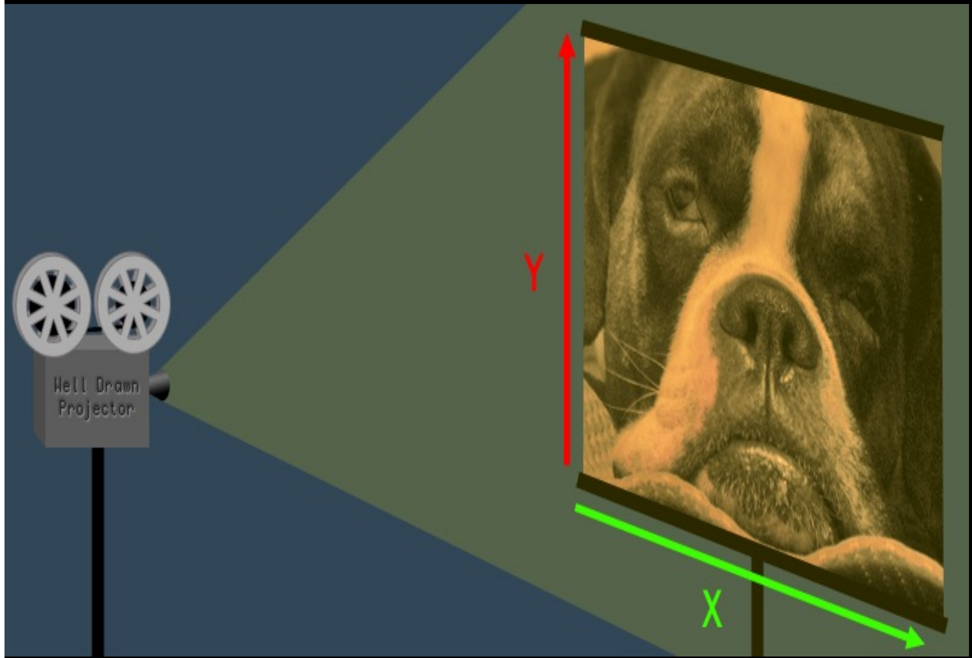
$$\begin{bmatrix} x' \\ y' \\ z' \\ w \end{bmatrix} = A \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

- Projective geometry
  - Used in computer vision
- Adds an extra dimension to vector, *W*:

$$[x, y, z, w]$$

- *W* scales the x, y and z dimensions
- *x,y,z* are “correct” when *W*=1
- How does it work?

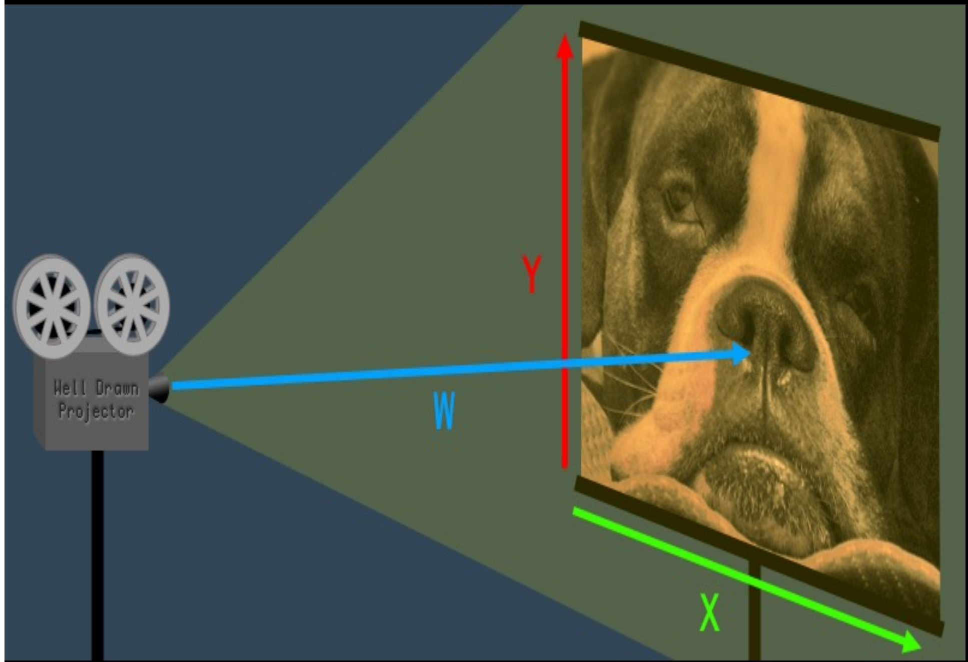
Homogeneous coordinates



- Euclidean geometry:
  - A point is (x,y)
  - A 2D image
  - Cartesian coordinates

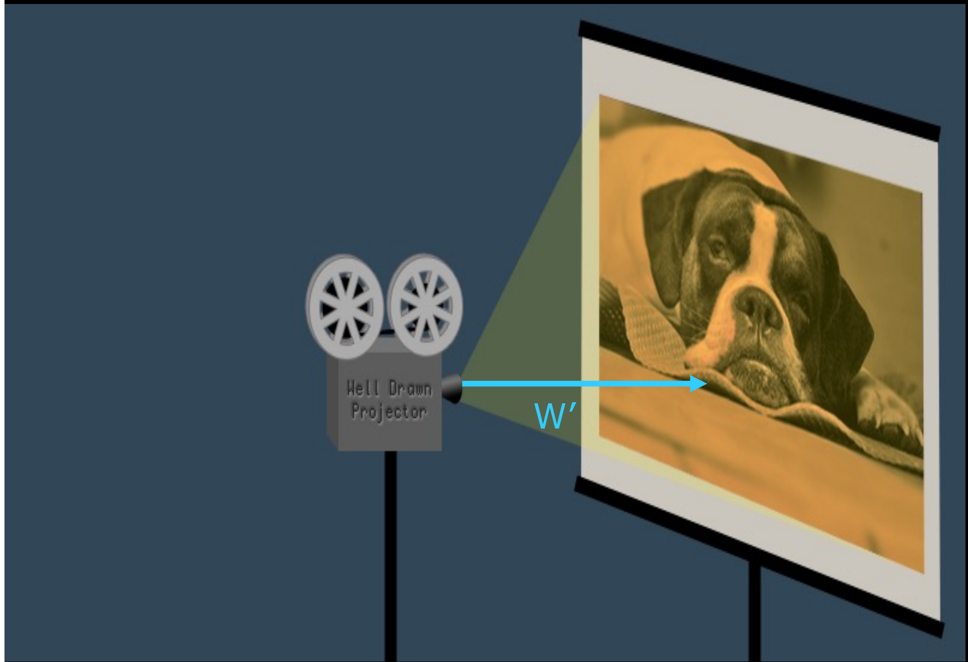
www.tomdalling.com/blog/modern-opengl/explaining-homogenous-coordinates-and-projective-geometry/

Homogeneous coordinates



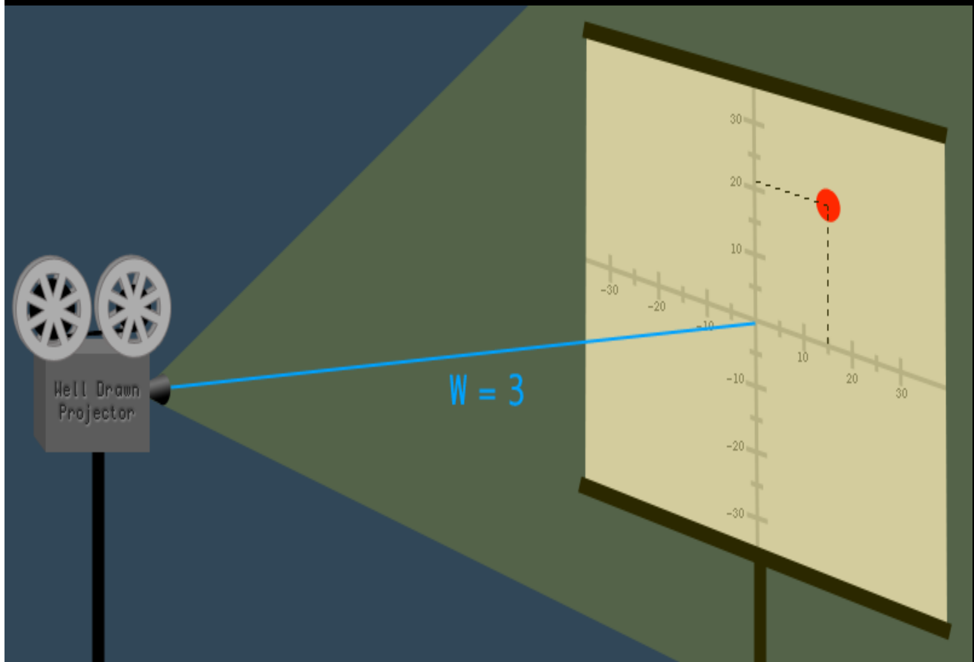
- Euclidean geometry:
  - A point is (x,y)
  - A 2D image
  - Cartesian coordinates
- Projective geometry:
  - A point is (x,y,W)
  - "Projective space" adds an extra projective dimension, W
  - Changing W scale factor:
    - No change to the point in projective space
    - Changing perspective/depth

Homogeneous coordinates



- A point in projective space is (*x*,*y*,*W*)
  - Its corresponding Euclidean point is (x/W,y/W)
- Increasing *W* *(the same x and y)*
  - The projected point appear closer to the origin
  - The object appear smaller (further away)
- Scaling to a new depth W'
  - Adjusting the point using a scale factor is *W'/W* i.e., new distance/old distance:
    (x*(W'/W), y*(W'/W), W'))
- When *W or W' = 1*
  - a projective coordinate (x,y,1) corresponds directly to Euclidean point (x,y)

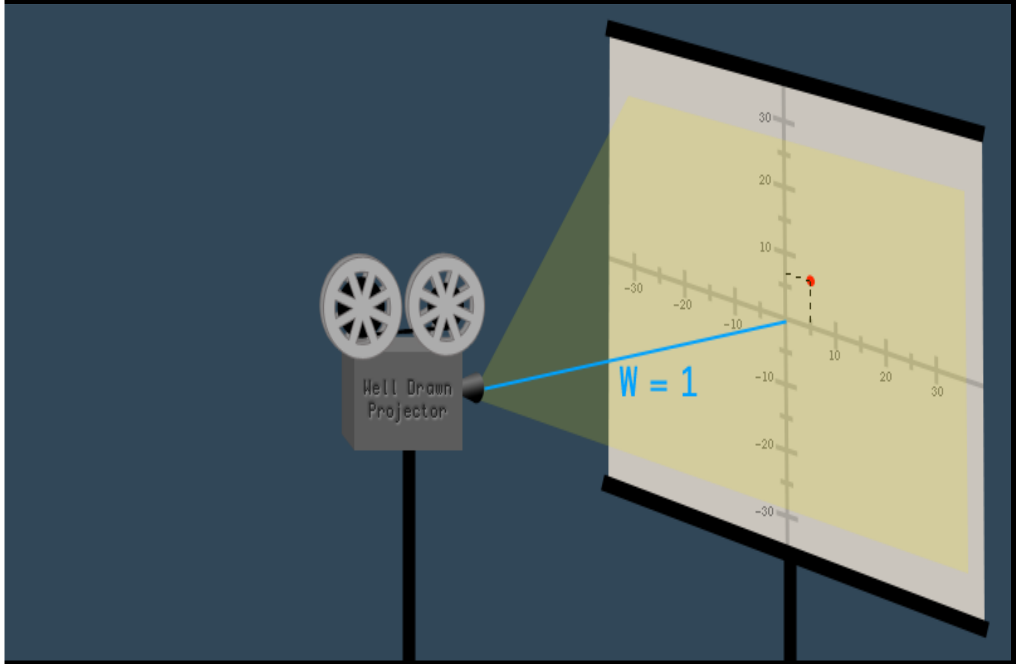
Homogeneous coordinates



Example:

- Camara:
  - 3 m away from the image, W=3
  - The dot on the image is at (15,21)
- The *projective coordinate point* is said to be
  - (15, 21, *3*)

Quiz 2: Homogeneous coordinates



A camara is placed at distance of 3 meter away from the image and the dot has the projective coordinate of (15,21,3).

Now we move the camara closer to the image i.e., 1 m away. What is the new projective coordinate?

A) (5,7,1)
B) (15,21,3)
C) (45,63,1)
D) (5,7,0.33)
E) (0,0,0)

SOLUTION:

We move closer to the image i.e. $W' = 1$ which scales with factor (1/3) the projective point at W=3 accordingly:

(15*(1/3),21*(1/3),1)=(5,7,1)

Translation transformation as a matrix

In Euclidian space

Translation: $\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix}$

In Projective space

$$\begin{bmatrix} x' \\ y' \\ z' \\ W \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ W \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ W \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} x' \\ y' \\ z' \\ W \end{bmatrix} = A_T \begin{bmatrix} x \\ y \\ z \\ W \end{bmatrix} \quad where\ A_T = \begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Geometrical transformations
  - Use Homogeneous coordinates
  - Set W=1 we 'covert' 3D → 4D space
  - Translation transformation expressed as a matrix A_T

Transformations in Projective space

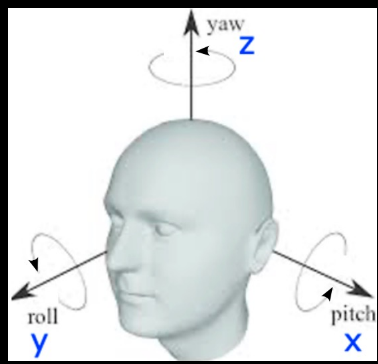
Translation: $A_T = \begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Rotations (right-hand rule):
- x=pitch
- y=roll
- z=yaw

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_y = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_z = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scaling: $A_s = \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Yaw-Pitch-Roll Euler convention



Shear: $A_z = \begin{bmatrix} 1 & Sxy & Sxz & 0 \\ Sxy & 1 & Syz & 0 \\ Sxz & Syz & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Affine transformation: $A = \underbrace{A_T * (R_z * R_x * R_y)}_{\text{Rigid}} * A_z * A_s$

Combining transformations – step by step

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix}$$

⬇

$$\begin{bmatrix} x' \\ y' \\ z' \\ W \end{bmatrix} = A_T \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ W \end{bmatrix}$$

Remember:

- Typical calculated in *radians*
- *Same procedure for 2D and 3D images*

- Step 1:Covert 3D to 4D projective space, set W=1. Make translation into a matrix

$$A = A_T * (R_x * R_y * R_z) * A_z * A_s$$

- Step 2:Multiply all 4D metrices

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = A \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- Step 3:Apply the transformation to a point

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = A \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

- Step 4:Convert back to 3D Cartesian coordinates by ignoring the *W* dimension

Different transformations

- Linear: Affine transformation
- Non-linear: Piece-wise affine or B-spline
  - Remember: First to apply the linear transformations!

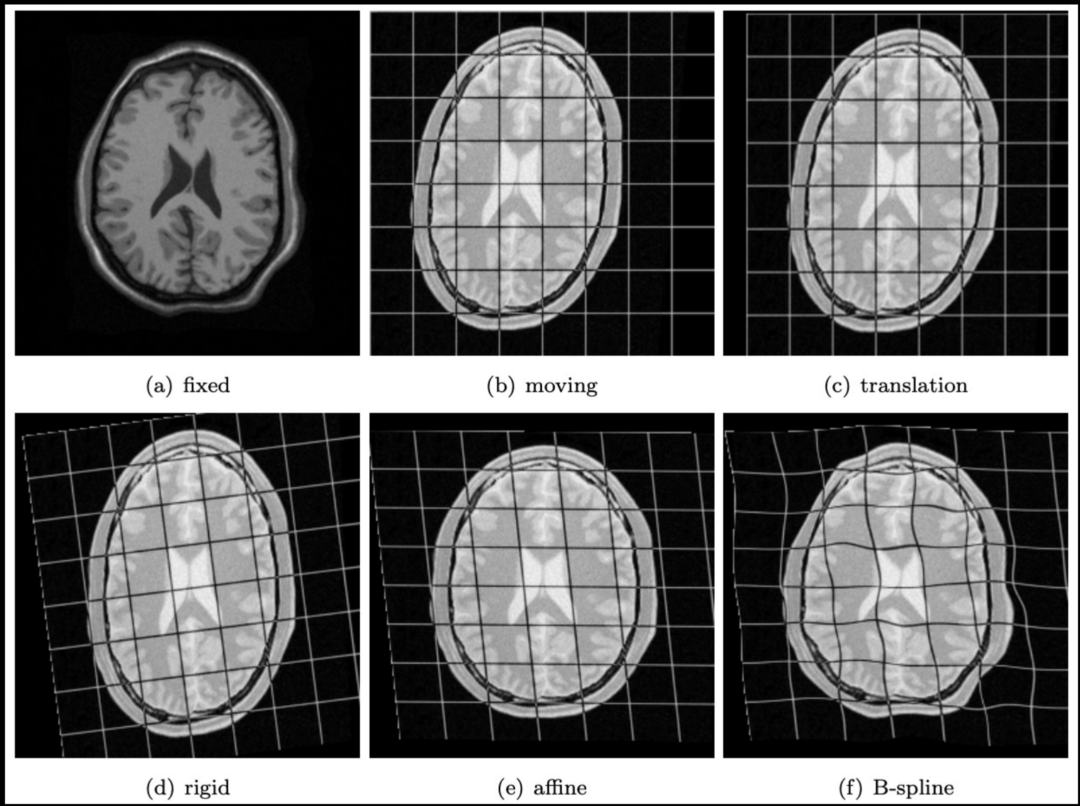
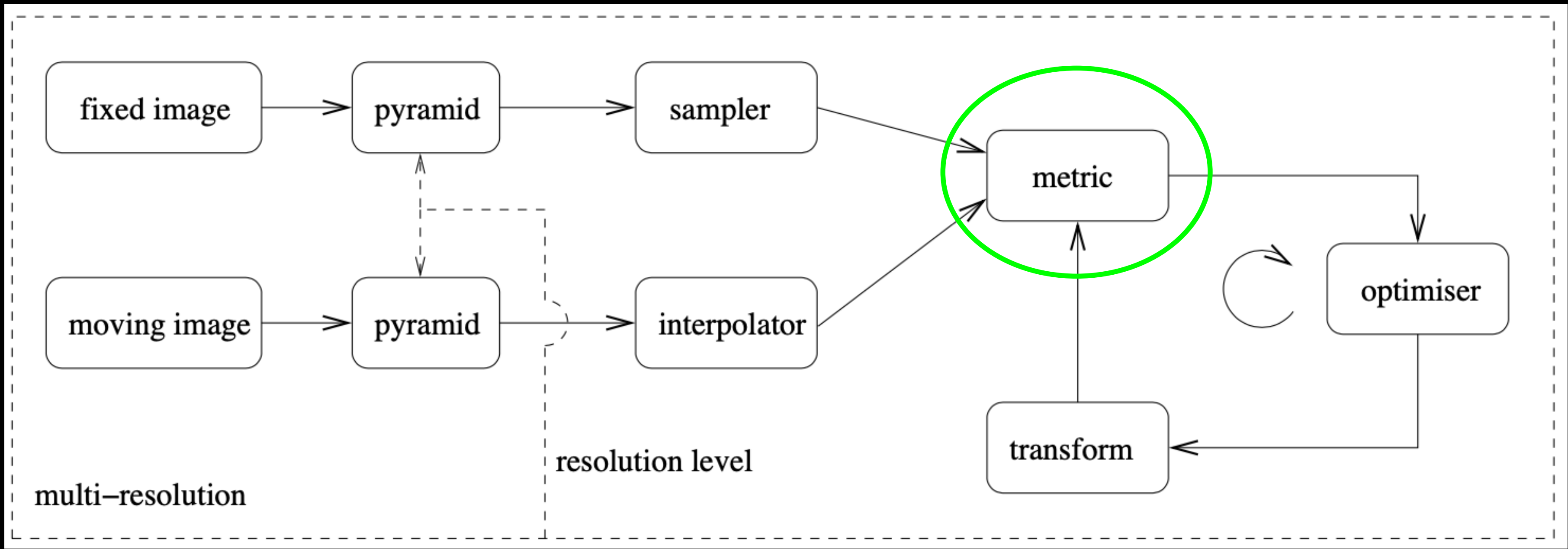


Image Registration pipeline

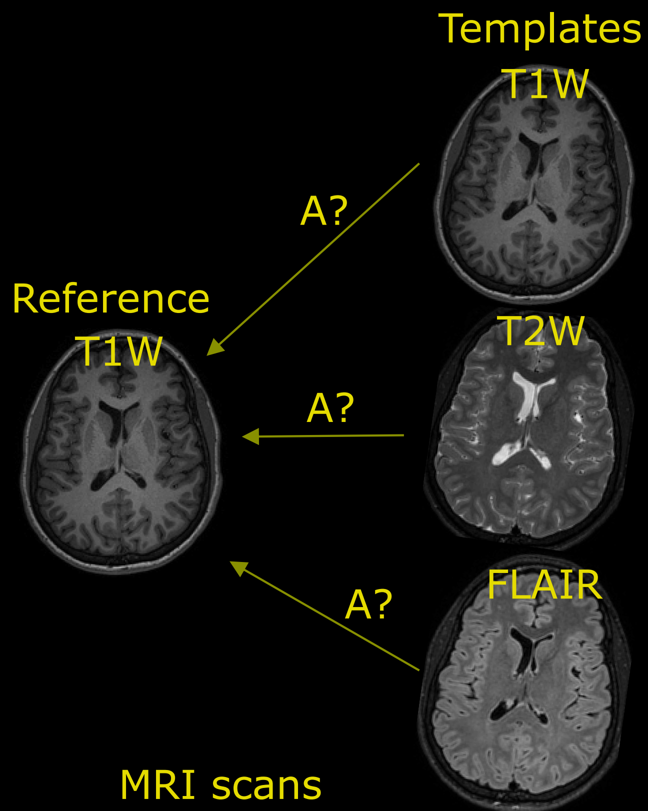
- Similarity measures



Similarity measures

Anatomical Landmarks

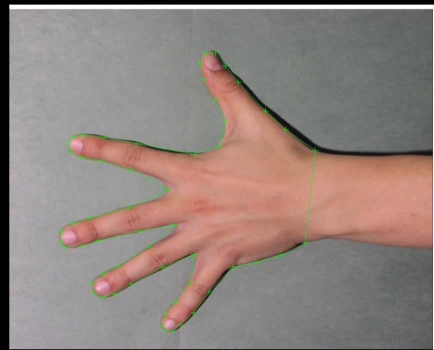
- time consuming to obtain positions manually
- Alternative: Joint intensity histogram



- Same subject
- Same intensity histogram

- Same subject
- Different intensity histogram

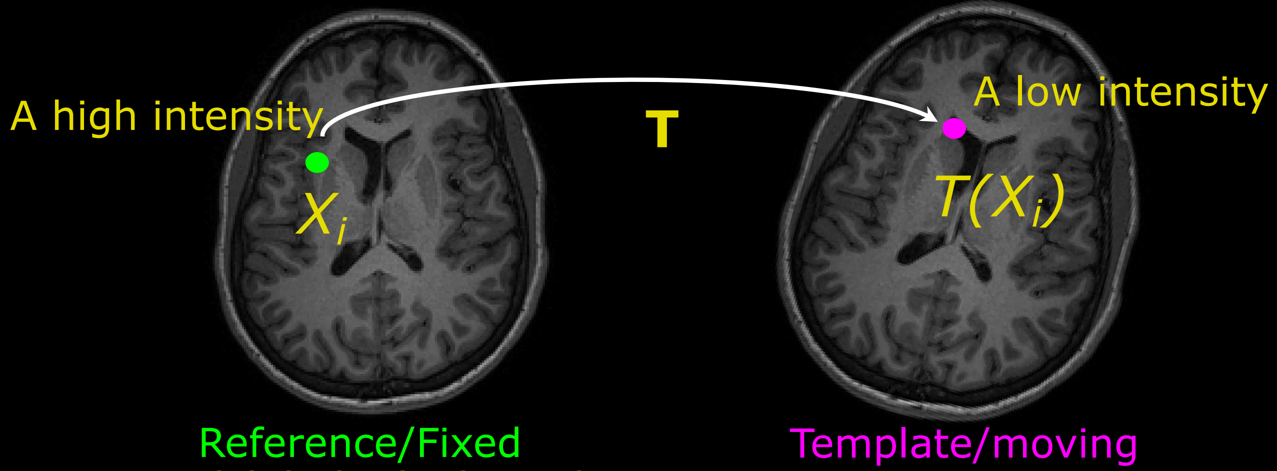
- Same subject
- Different intensity histogram



Similarity measure: Mean squared difference (MSD)

- Compare difference in intensities.
  - Same similarity measure we used for anatomical landmarks (positions) in a previous lecture
  - Fast to estimate
- Many local minima’s (sub optimal solutions)
  - Intensities are not optimal for this similarity metric

$$\mathrm{MSD}(\boldsymbol{\mu}; I_F, I_M) = \frac{1}{|\Omega_F|} \sum_{\boldsymbol{x}_i \in \Omega_F} \left(I_F(\boldsymbol{x}_i) - I_M(\boldsymbol{T}_{\boldsymbol{\mu}}(\boldsymbol{x}_i))\right)^2,$$



Is T optimal?

NO!

- Big intensity difference
- Large MSD error

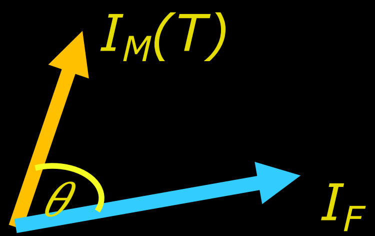
Similarity measure: Normalised Cross-correlation

- Normalised Cross-correlation of intensities in two images
  - Fast to estimate
- Risk of local minima's (sub optimal solutions)
  - Less robust if image modalities have different intensity histograms
  - Normalise: Reduce the impact of outlier regions

$$\mathrm{NCC}(\boldsymbol{\mu}; I_F, I_M) = \frac{\sum_{\boldsymbol{x}_i \in \Omega_F} \left(I_F(\boldsymbol{x}_i) - \overline{I_F}\right)\left(I_M(\boldsymbol{T}_{\boldsymbol{\mu}}(\boldsymbol{x}_i)) - \overline{I_M}\right)}{\sqrt{\sum_{\boldsymbol{x}_i \in \Omega_F} \left(I_F(\boldsymbol{x}_i) - \overline{I_F}\right)^2 \sum_{\boldsymbol{x}_i \in \Omega_F} \left(I_M(\boldsymbol{T}_{\boldsymbol{\mu}}(\boldsymbol{x}_i)) - \overline{I_M}\right)^2}},$$

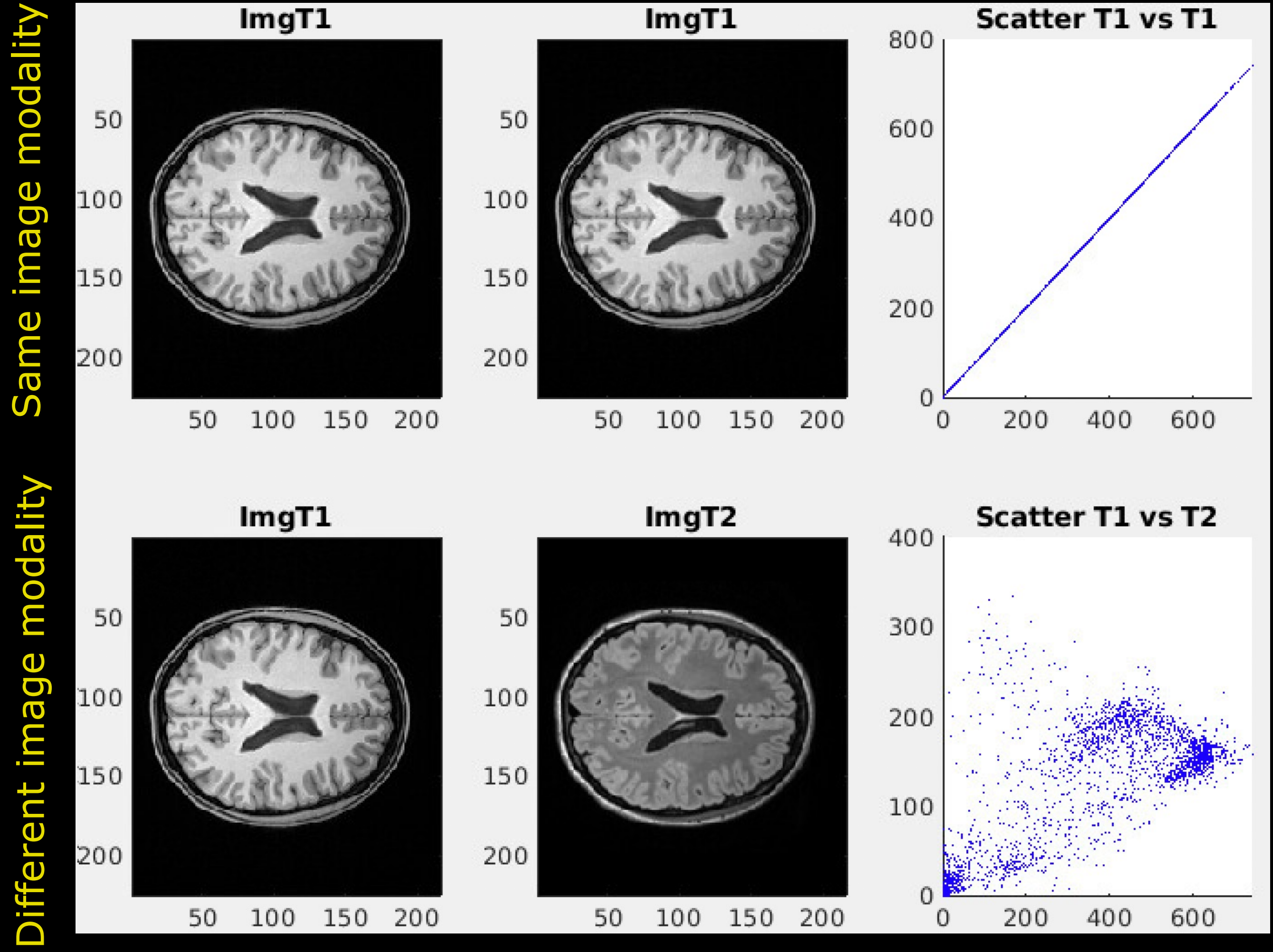
with the average grey-values $\overline{I_F} = \frac{1}{|\Omega_F|} \sum I_F(\boldsymbol{x}_i)$ and $\overline{I_M} = \frac{1}{|\Omega_F|} \sum I_M(\boldsymbol{T}_{\boldsymbol{\mu}}(\boldsymbol{x}_i))$.

- Multiplication is a dot product
  - $I_F \cdot I_M(T) = \|I_F\| \, \|I_M(T)\| \cos\theta$



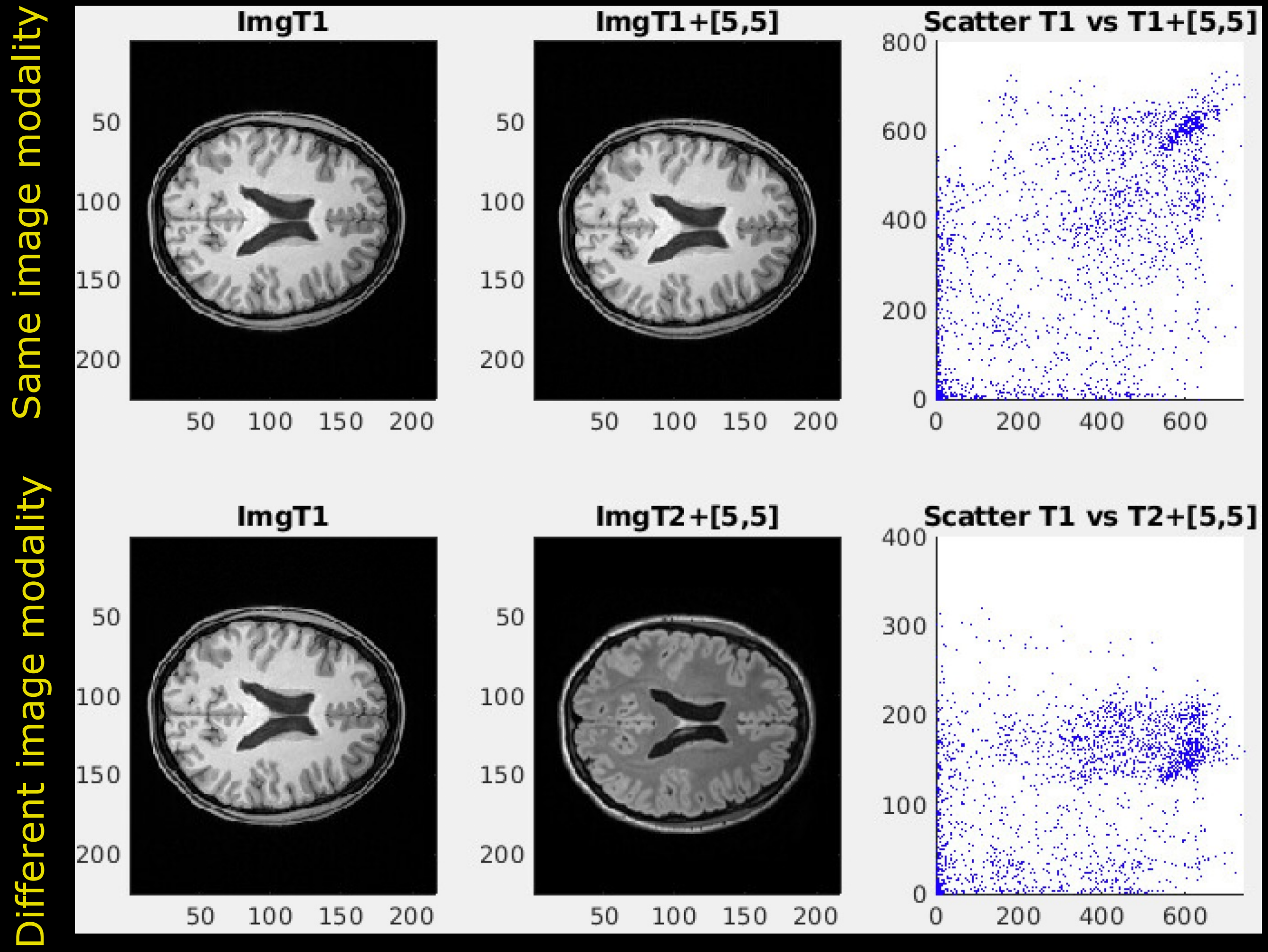
Joint intensity histograms

- Perfect registered: Optimal joint intensity agreement



Joint intensity histograms

- Small translation difference: Lower joint intensity agreement

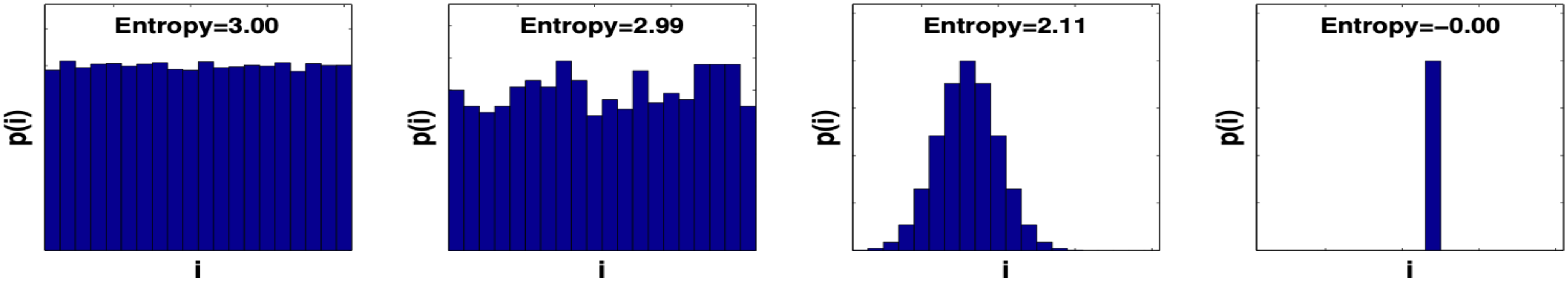


Similarity measure - Entropy

- Comes from information theory.
  - The higher the entropy the more the information content.
- Entropy (Shannon-Weiner):

$$H = -\sum_i p_i \log_b p_i$$

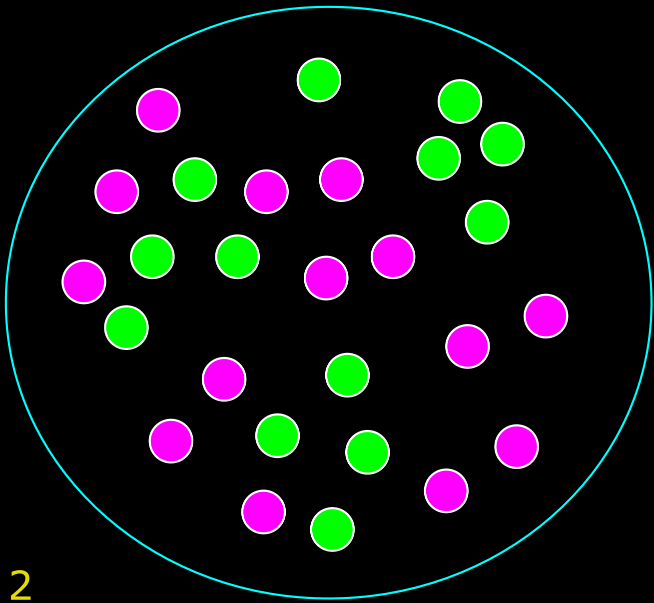
Where *b*: the base of the logarithm
- Bits: *b*=2 and bans: *b*=10
- Entropy is typically in bits i.e. typical used in digital information



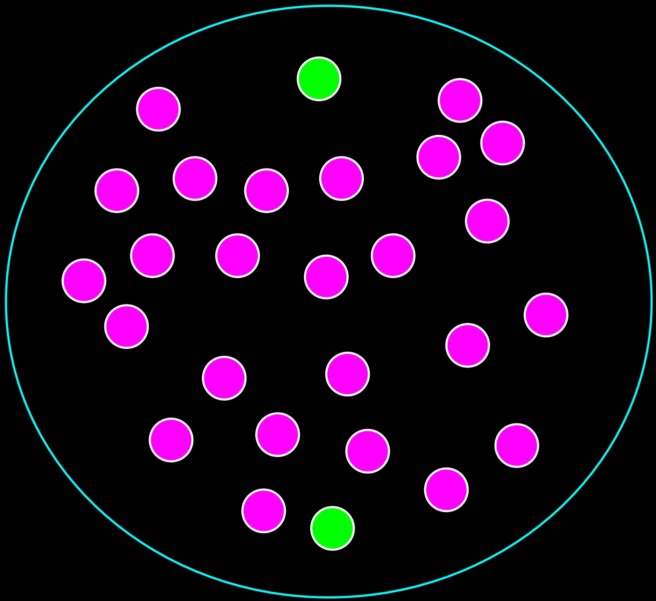
Quiz 3: Highest entropy?

I went to the candy shop and wanted to select the cady mixture that has the highest entropy. Each candy mixture include in total 27 pieces. Which one should I select?

Candy mix 1



Candy mix 2



A) Mix 1
B) Make a new choice
C) Contain no liquorice
D) Mix 2
E) It is not healthy

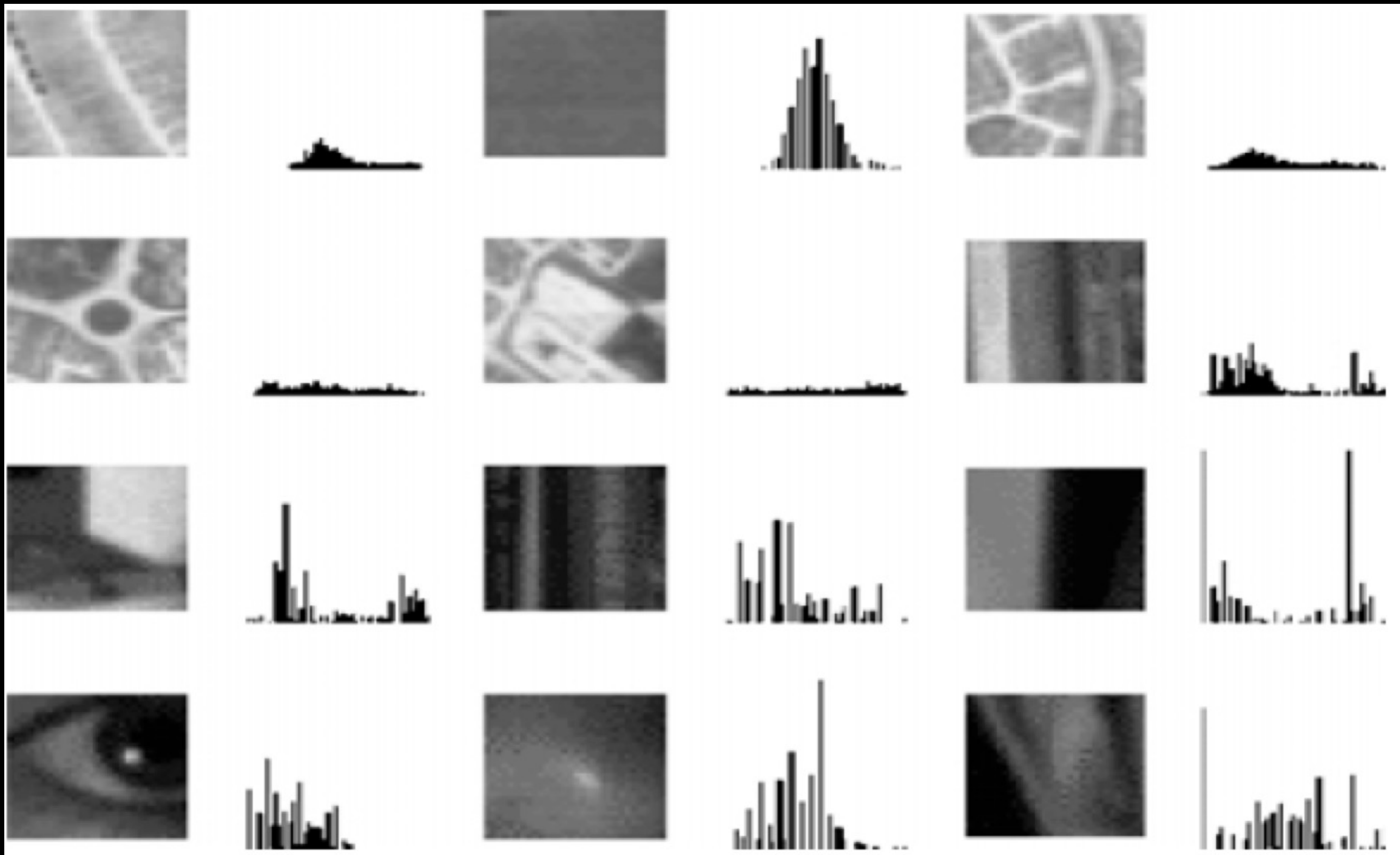
Quiz 4: What is the entropy of the candy mix 1?

A) 0.38
B) 0.99
C) 0.45
D) 0.23
E) 0.00

SOLUTION:
Green=13
Pink=14
Total=27

pG=13/27
pP=14/27
Entropy= $-pG*\log_2(pG)-pP*\log_2(pP)=0.99$

Histograms of images

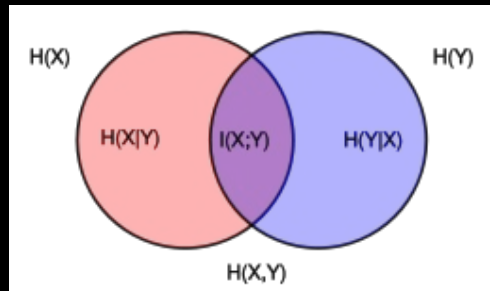


Joint entropy - Mutual information

- Joint entropy $H(X,Y) = -\sum_{X,Y} p_{X,Y} \log p_{X,Y}$

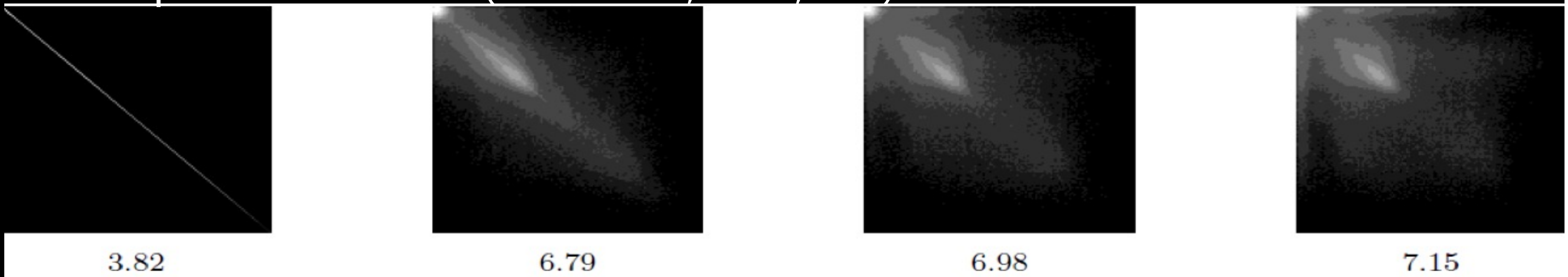
- Similarity measure: The more similar the distributions, the lower the joint entropy compared to the sum of the individual entropies i.e., total area is less spread out

$$H(X,Y) \leq H(X) + H(Y)$$



en.wikipedia.org/wiki/Mutual_information

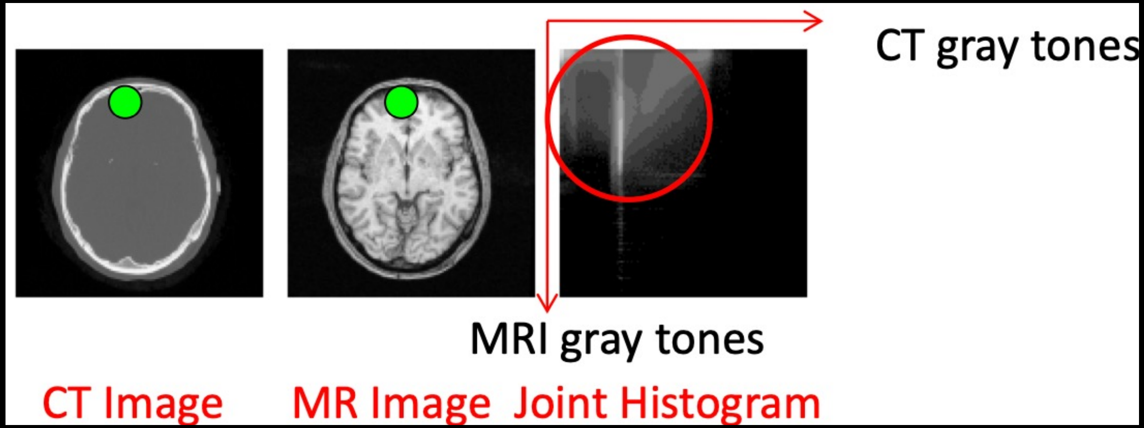
- Example of rotation (Pluim et al., 2003, TMI)



0 degrees 2 degrees 5 degrees 10 degrees

Contrast in joint histograms

- The histogram of the two images must reflect contrast to similar structures for image registration to be successful



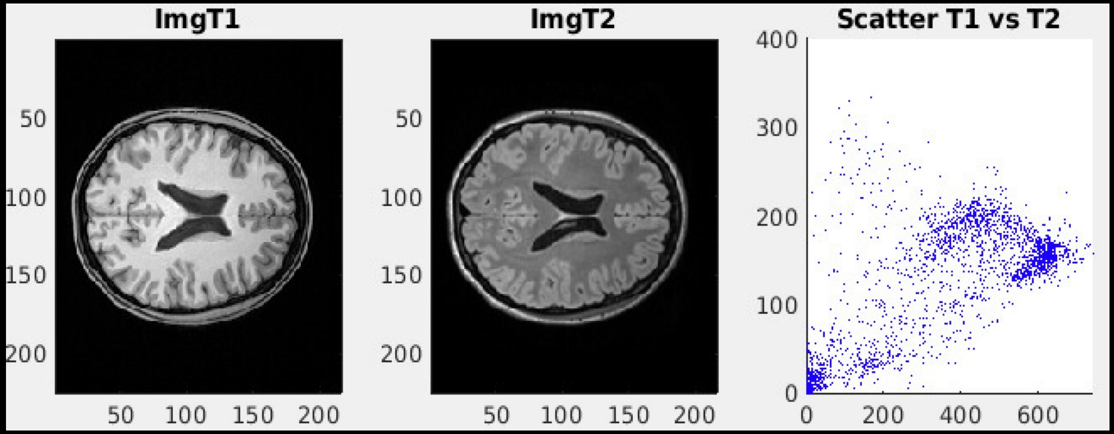
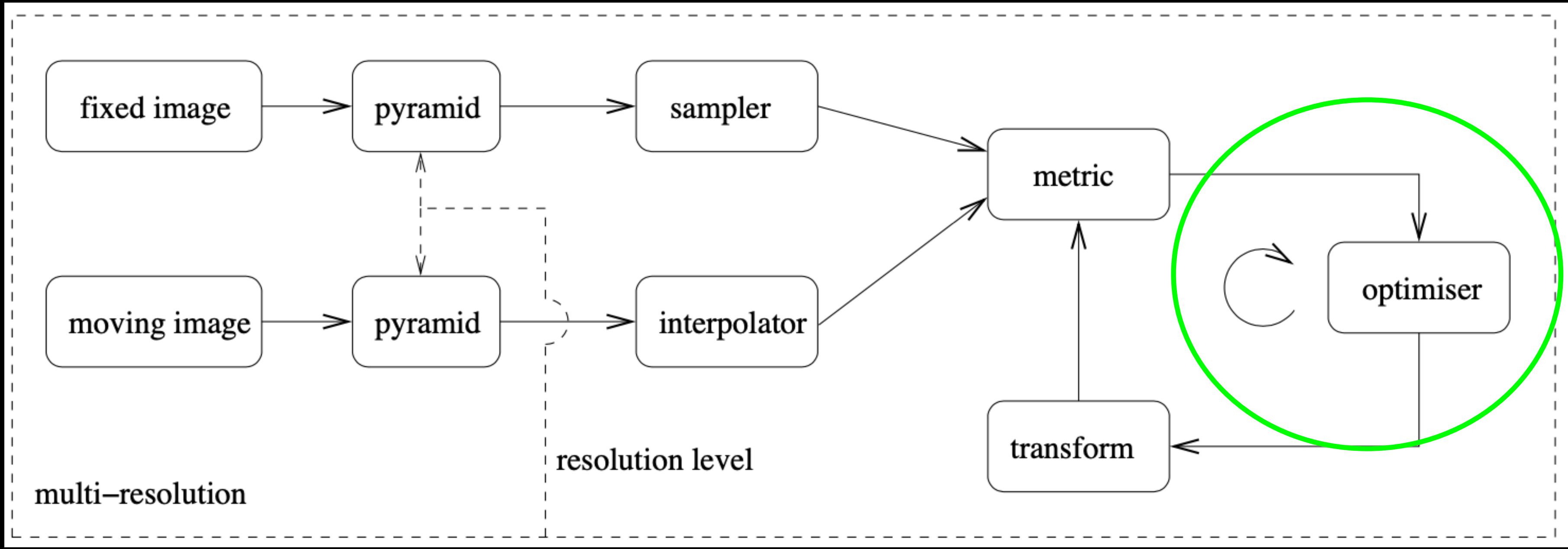


Image Registration pipeline

- The optimiser
  - How to find the transformation parameters?



The optimizer

- We have an objective function describing:
  - A cost function (C) based on a similarity metric
    - Quantifying how well a geometrical transformation ($T(w)$) maps an image (moving, I_M) into another (fixed, I_F)
- Hence, a good match is a minimum difference:

$$\hat{T}_w = \arg\min_{T_w} C\,(T_w; I_F, I_M)$$

The parameters

$$w \in \mathcal{R}^p$$

- The parameters is a vector with p elements
- The type of transformation and the dimension of the dataset set the number of parameters
  - Translation p = 2 or 3 (3D)
  - Rotation p = 1 or 3 (3D)
  - Scaling p = 1

Optimization by minimization

- Find the parameter set that minimizes the objective function
- How to find the solution?
  - Analytical: Works fine for translation (previous lecture)
  - Numerical: Iterative approaches to search for affine transformations

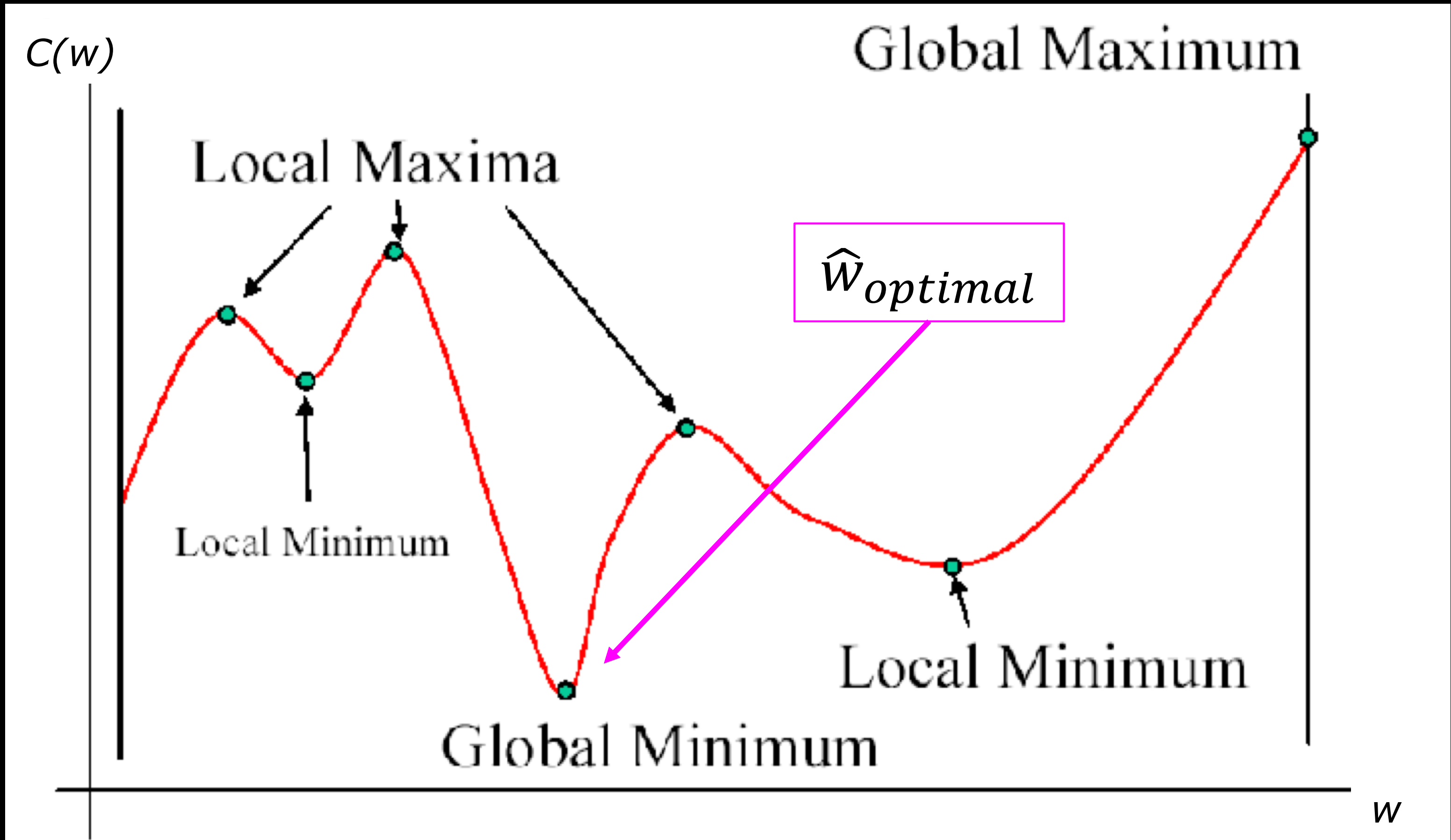
To find: $\hat{w} = \arg\min_{w} C$

We simply differentiate w.r.t. *w*:

$$\frac{\partial C}{\partial w} = 0$$

The challenge

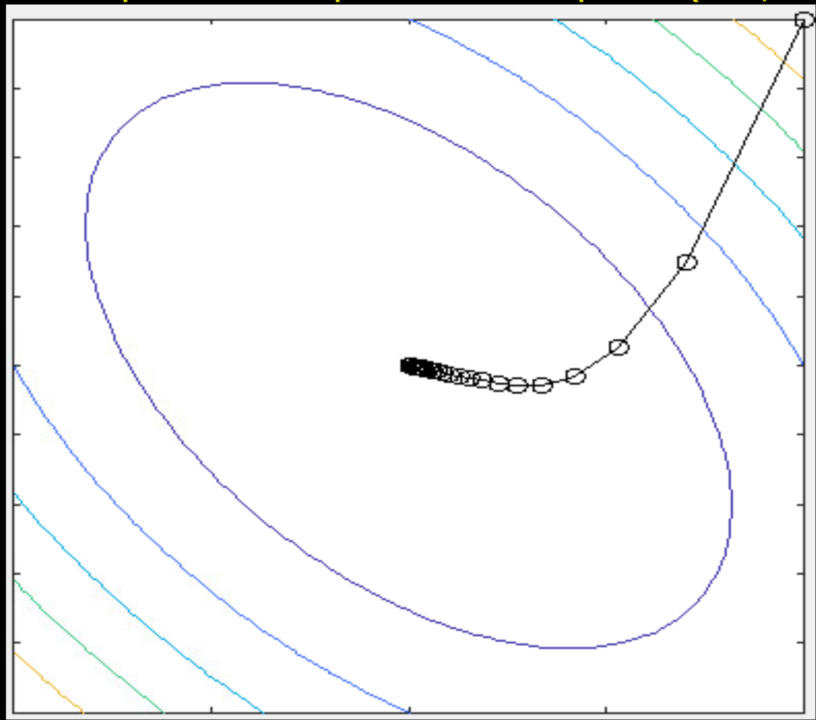
- ***w*** span a p-dimensional space $\mathbf{w}=[w_1,w_2,\ldots,w_p]^T$
- Complex parameter space with many data points
  - Finding the lowest place in mountains



Iterative optimisation

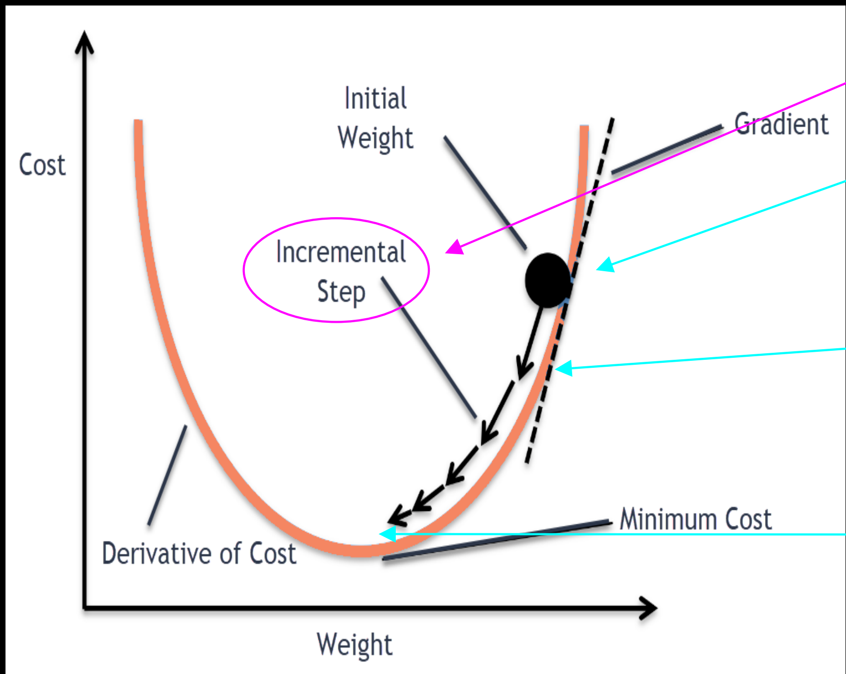
- Aim: Find in parameter space w: $\frac{\partial C}{\partial w} = 0$ i.e. a global minima
  - Search all possible combinations of w? (not a good idea)
  - Systematically search the parameter space = Good idea
- Iterative optimisation strategies
  - Step-wise searching the parameter space
- Many methods exist
  - Gradient based
  - Genetic evolution
  - ...

Contour plot of 2D parameter space (w1,w2)



Gradient descent

- Definition: C(***w***) is differentiable in neighbourhood of a point w_n
- C(***w***) decreases in the *negative* gradient direction of w_n.
- $w_{n+1} = w_n - \gamma \nabla C(w_n)$
  - $\nabla C(w_n)$: Gradient direction at point w_n
  - γ: Step length --> If small enough: $C(\boldsymbol{w}_n) \geq C(\boldsymbol{w}_{n+1})$



Procedure:

γ 0) Define a step length

$\nabla C(w_0)$ 1) Start guess of a position

2) Find gradient

3) Take a step

$\nabla C(w_1)$ 4) Repeat 2)+3)

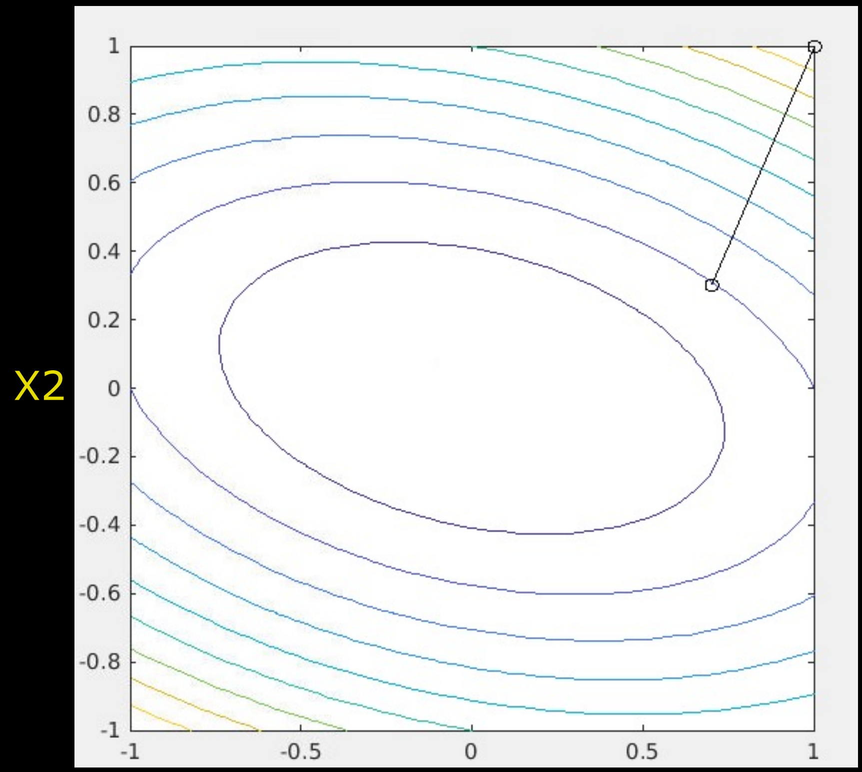
5) Solution: Global minima

$$\nabla C(w_{n+1}) = \frac{\partial C}{\partial w} \approx 0$$

Gradient descent

- Cost function: $C(x) = x_1^2 + x_1 x_2 + 3x_2^2$
- Gradient at point x_n: $-\nabla C(x_n) = -\begin{bmatrix} 2x_1 + x_2 \\ x_1 + 6x_2 \end{bmatrix}$
- Step length: γ=0.1;
- Max steps: 1000
- Start position: $x_0=[1,1]^T$

Iteration:1

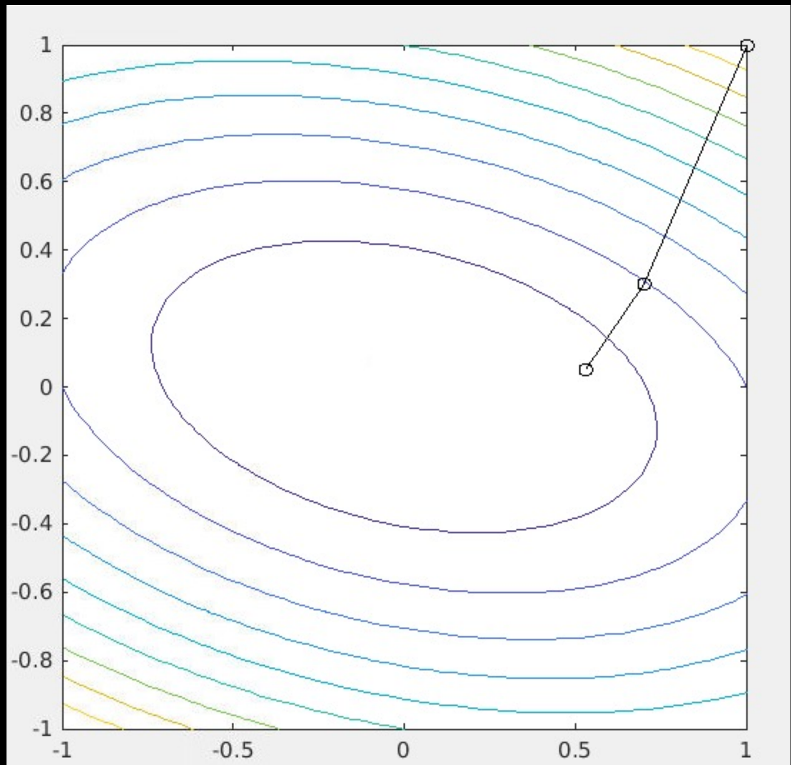


From a Matlab function: *grad_descent.m*
By James T. Allison

Gradient descent

- Cost function: $C(x) = x_1^2 + x_1 x_2 + 3x_2^2$
- Gradient at point x_n: $-\nabla C(x_n) = -\begin{bmatrix} 2x_1 + x_2 \\ x_1 + 6x_2 \end{bmatrix}$
- Step length: γ=0.1;
- Max steps: 1000
- Start position: $x_0=[1,1]^T$

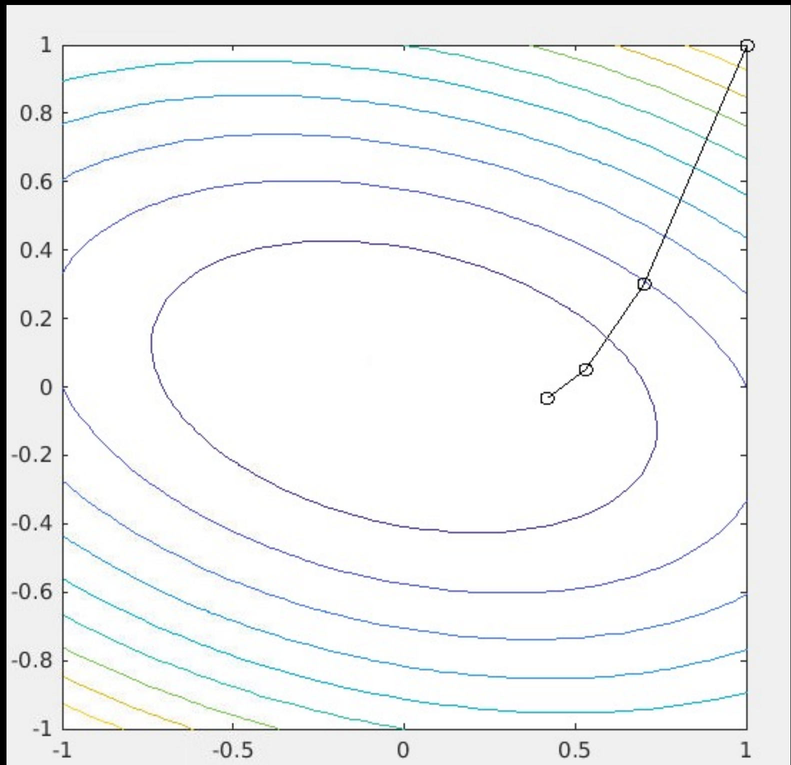
Iteration: 2



Gradient descent

- Cost function: $C(x) = x_1^2 + x_1x_2 + 3x_2^2$
- Gradient at point x_n: $-\nabla C(x_n) = -\begin{bmatrix} 2x_1 + x_2 \\ x_1 + 6x_2 \end{bmatrix}$
- Step length: γ=0.1;
- Max steps: 1000
- Start position: $x_0=[1,1]^T$

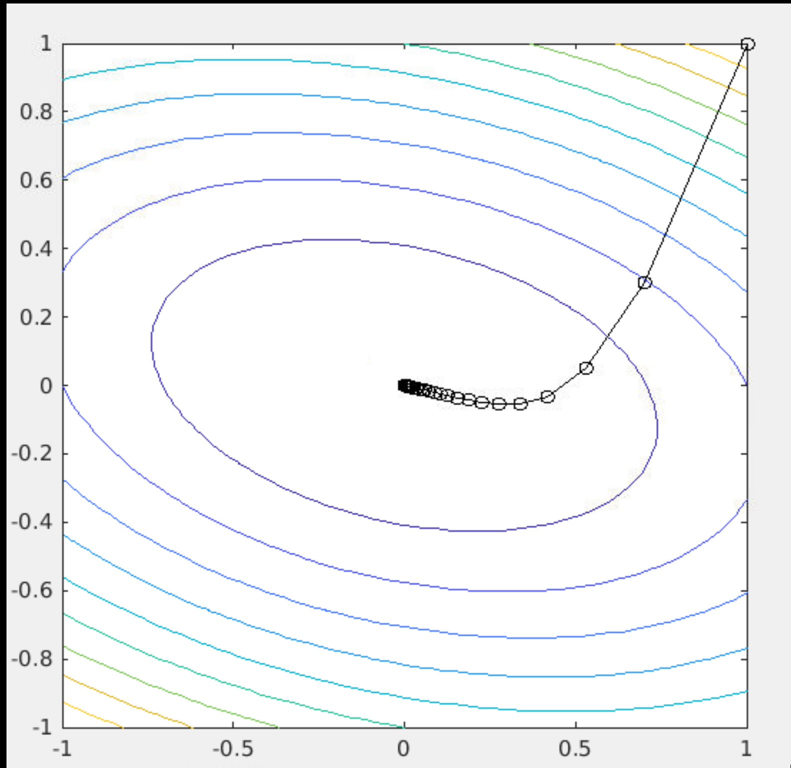
Iteration:3



Gradient descent

- Cost function: $C(x) = x_1^2 + x_1 x_2 + 3x_2^2$
- Gradient at point x_n: $-\nabla C(x_n) = -\begin{bmatrix} 2x_1 + x_2 \\ x_1 + 6x_2 \end{bmatrix}$
- Step length: γ=0.1;
- Max steps: 1000
- Start position: $x_0=[1,1]^T$

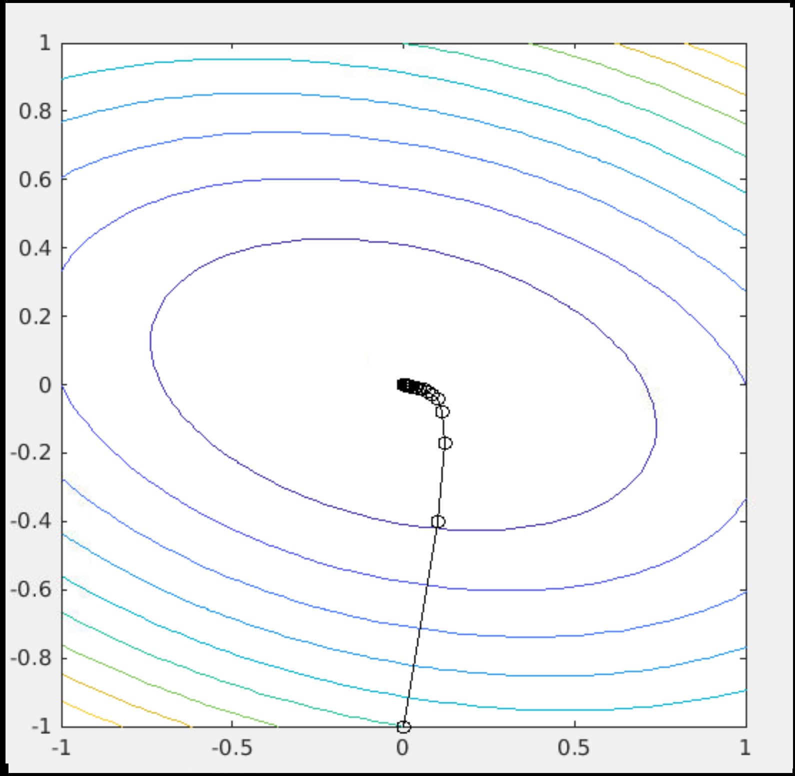
Iteration:37 (final)



Gradient descent

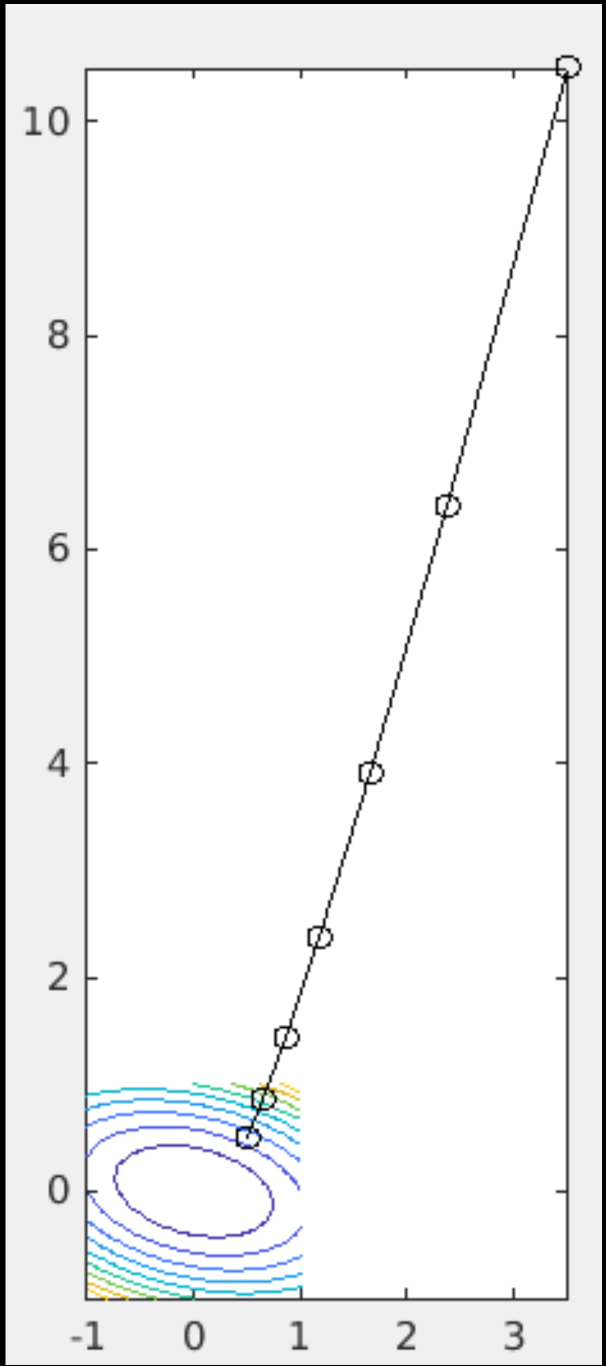
- Cost function: $C(x) = x_1^2 + x_1x_2 + 3x_2^2$
- Gradient at point x_n: $-\nabla C(x_n) = -\begin{bmatrix} 2x_1 + x_2 \\ x_1 + 6x_2 \end{bmatrix}$
- Step length: γ=0.1;
- Max steps: 1000
- Start position: $x_0=[0,-1]^T$
- Can find solution from any place
- No local minima's nearby

Iteration:31 (final)



Gradient descent

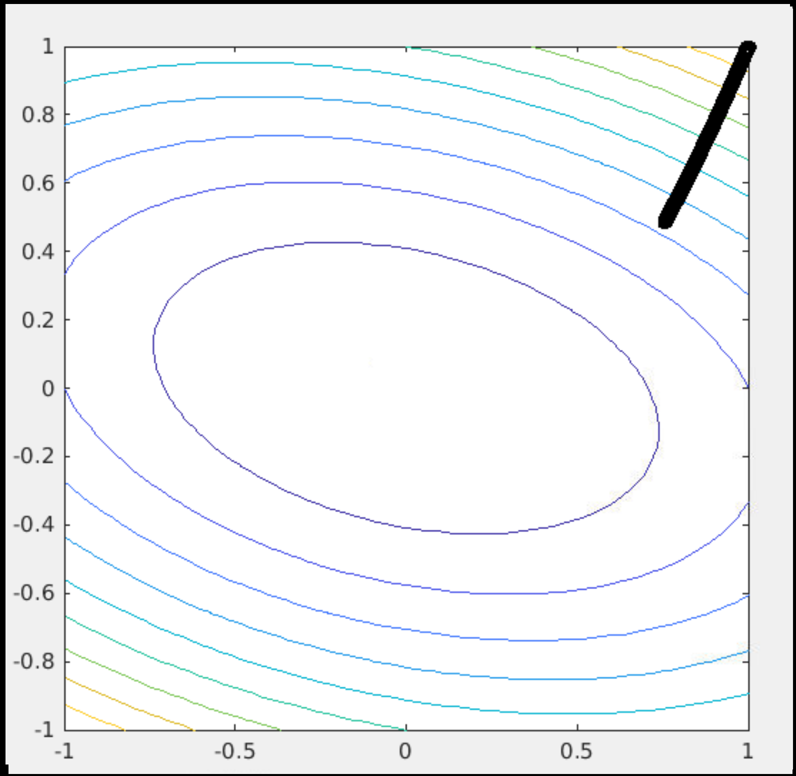
- Cost function: $C(x) = x_1^2 + x_1 x_2 + 3x_2^2$
- Gradient at point x_n: $+\nabla C(x_n) = + \begin{bmatrix} 2x_1 + x_2 \\ x_1 + 6x_2 \end{bmatrix}$
- Step length: γ=0.1;
- Max steps: 1000
- Start position: $x_0=[0.5,0.5]^T$
- If use positive gradient
  - WRONG DIRECTION!



Gradient descent

- Cost function: $C(x) = x_1^2 + x_1 x_2 + 3x_2^2$
- Gradient at point x_n: $-\nabla C(x_n) = -\begin{bmatrix} 2x_1 + x_2 \\ x_1 + 6x_2 \end{bmatrix}$
- Step length: γ=0.0001;
- Max steps: 1000
- Start position: $x_0=[1,1]^T$
- Too small step size –many steps
- Do not find a solution

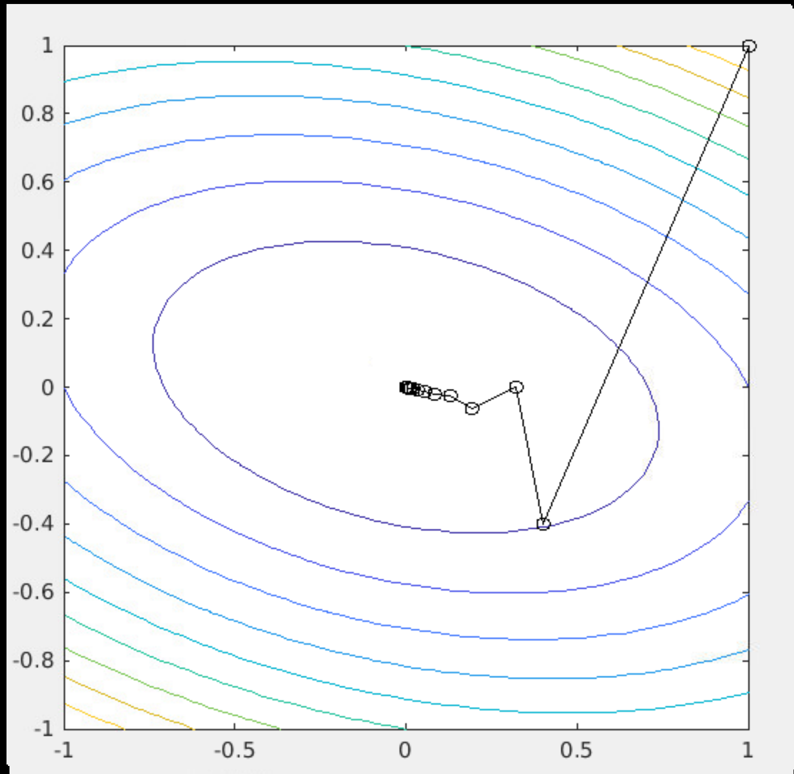
Iteration:1000 (final)



Gradient descent

- Cost function: $C(x) = x_1^2 + x_1 x_2 + 3x_2^2$
- Gradient at point x_n: $-\nabla C(x_n) = -\begin{bmatrix} 2x_1 + x_2 \\ x_1 + 6x_2 \end{bmatrix}$
- Step length: γ=0.2 (optimal)
- Max steps: 1000
- Start position: $x_0=[1,1]^T$
- Few steps: Optimal step size

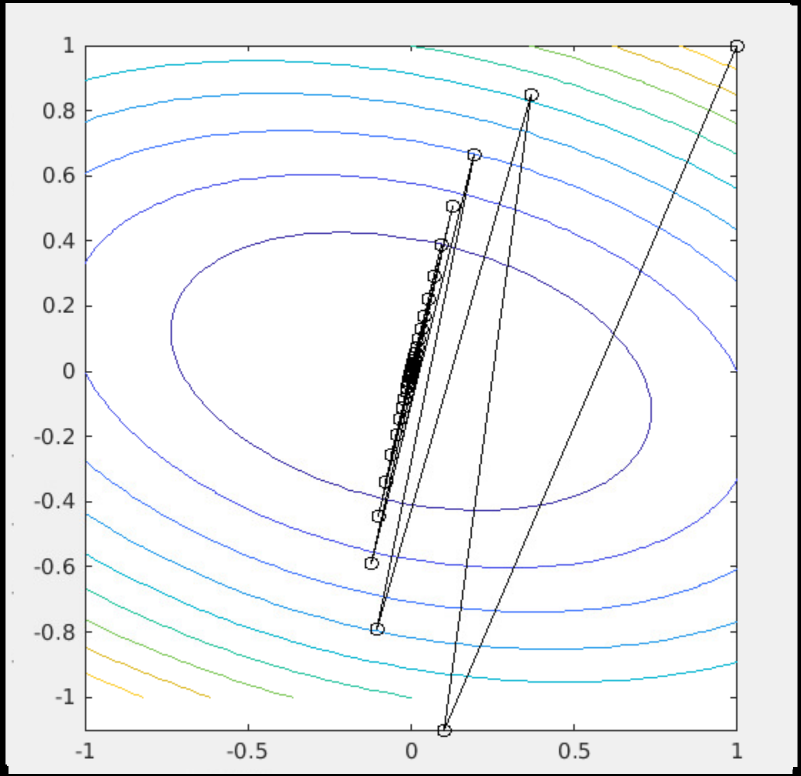
Iteration:17 (final)



Gradient descent

- Cost function: $C(x) = x_1^2 + x_1 x_2 + 3x_2^2$
- Gradient at point x_n: $-\nabla C(x_n) = -\begin{bmatrix} 2x_1 + x_2 \\ x_1 + 6x_2 \end{bmatrix}$
- Step length: γ=0.3
- Max steps: 1000
- Start position: $x_0=[1,1]^T$
- Too large step size – unstable
- Sensitive to local minima's
- Solution: Dynamic step length

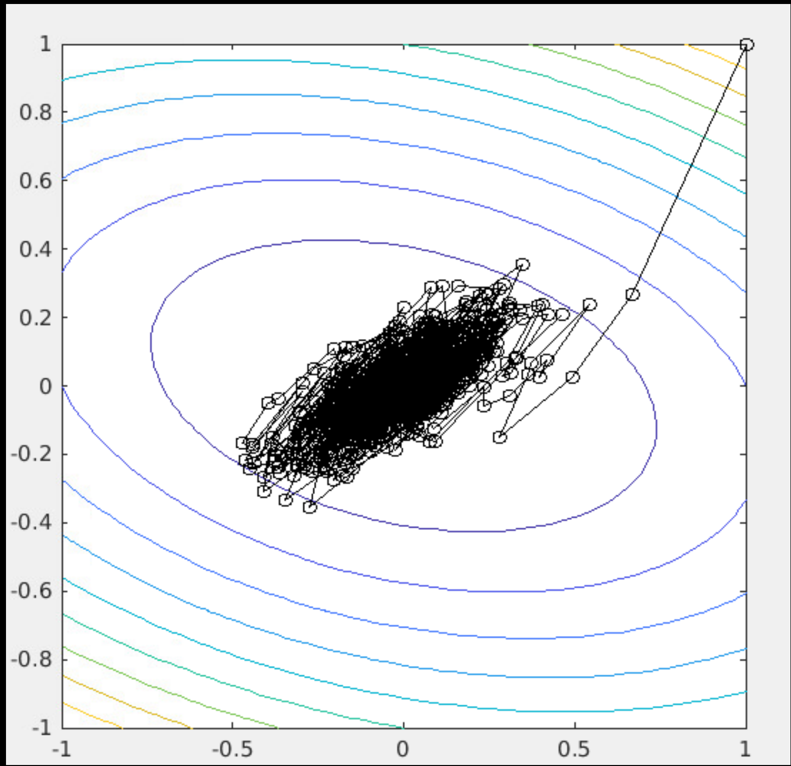
Iteration:65 (final)



Gradient descent

- Cost function: $C(x) = x_1^2 + x_1 x_2 + 3x_2^2$
- Gradient at point x_n: $-\nabla C(x_n) = -\begin{bmatrix} 2x_1 + x_2 \\ x_1 + 6x_2 \end{bmatrix}$
- Step length: γ=0.1
- Max steps: 1000
- Start position: $x_0=[1,1]^T$
- Noisy data: Cannot find optimum

Iteration:1000 (final)



Quiz 5:What is the updated position xnew?

Model fitting uses a cost function: $C(x) = x_1^2 + x_1 x_2 + 3x_2^2$
and an iterative optimizer Gradient descent with a step length of 0.2

What is the new position of xnew $=[?,?]^T$ after one step from position $x=[1, 0]^T$?

A) $[0.3,2.3]^T$
B) $[-1.7,0.3]^T$
C) [1.4,0.2]T
D) $[0.6,-0.2]^T$
E) $[5.2,2.2]^T$

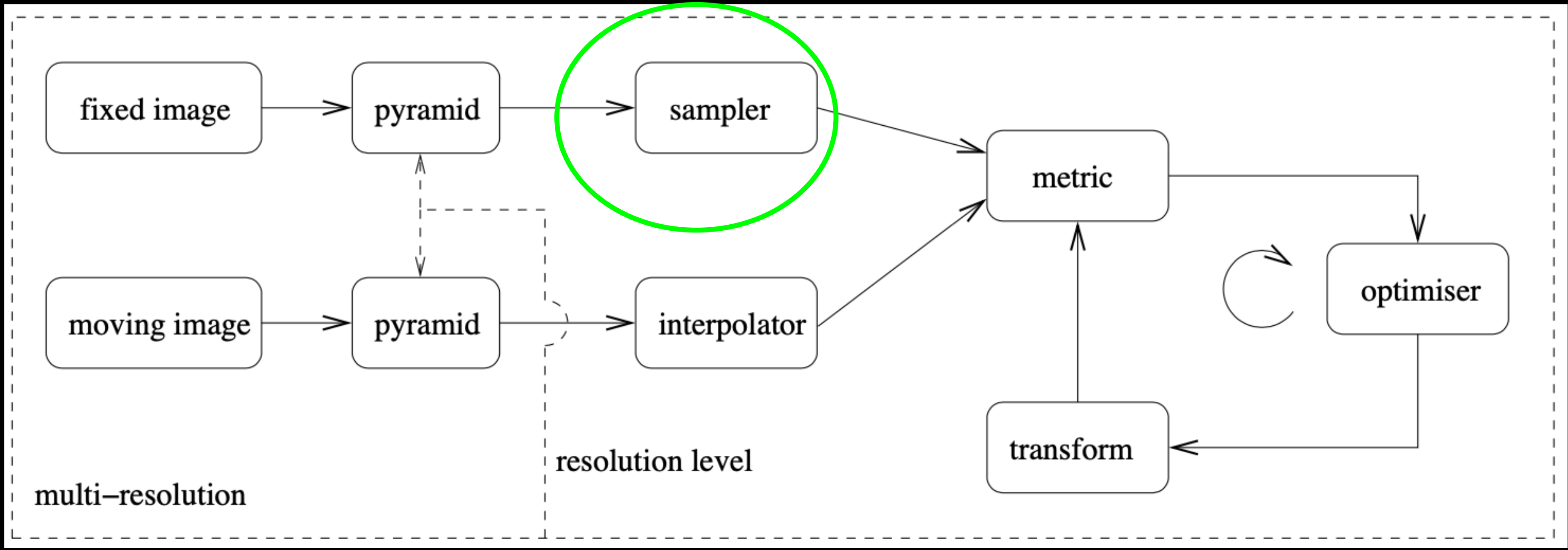
Solution:
1) Calculate the gradient for $x=[1,0]^T$
- differentiate C: $\nabla C(x) = \begin{bmatrix} 2x_1 + x_2 \\ x_1 + 6x_2 \end{bmatrix}$

$\nabla C([1,0]^T) = [2,1]^T$

2) Update the step: $x_{new} = x - \nabla C * stepLength$
- $xnew=[1,0]^T-0.2*[2,1]^T=[0.6, -0.2]^T$

Image Registration pipeline

- The sampler
  - How many data points for a robust similarity measure?

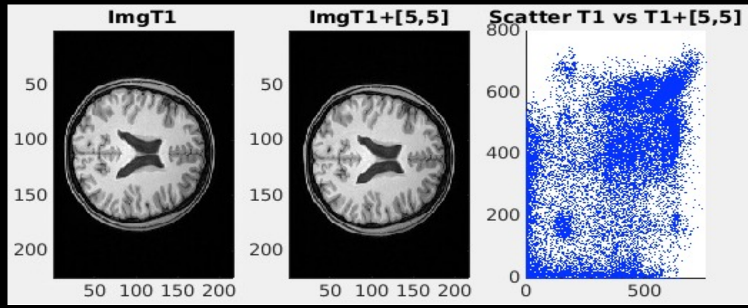


The sampler

- Calculating the similarity metrics:
  - Summing over all pixels/voxels in an image is VERY time consuming
- Selecting a sparse sampling strategy
  - Reducing CPU load and reduce memory load when
  - Efficient selection of image points

with translation

All samples



The sampler

- Sparser sampling: Similar scatter plot
  - Define a good compromise (sample the whole image)
- Ordered vs Random
  - Spatial dependency: Dependent on large homogeneous structures
  - Very sparse sampling: Risk not sampling small structures

Ordered — Random

Every 2nd — Every 10th

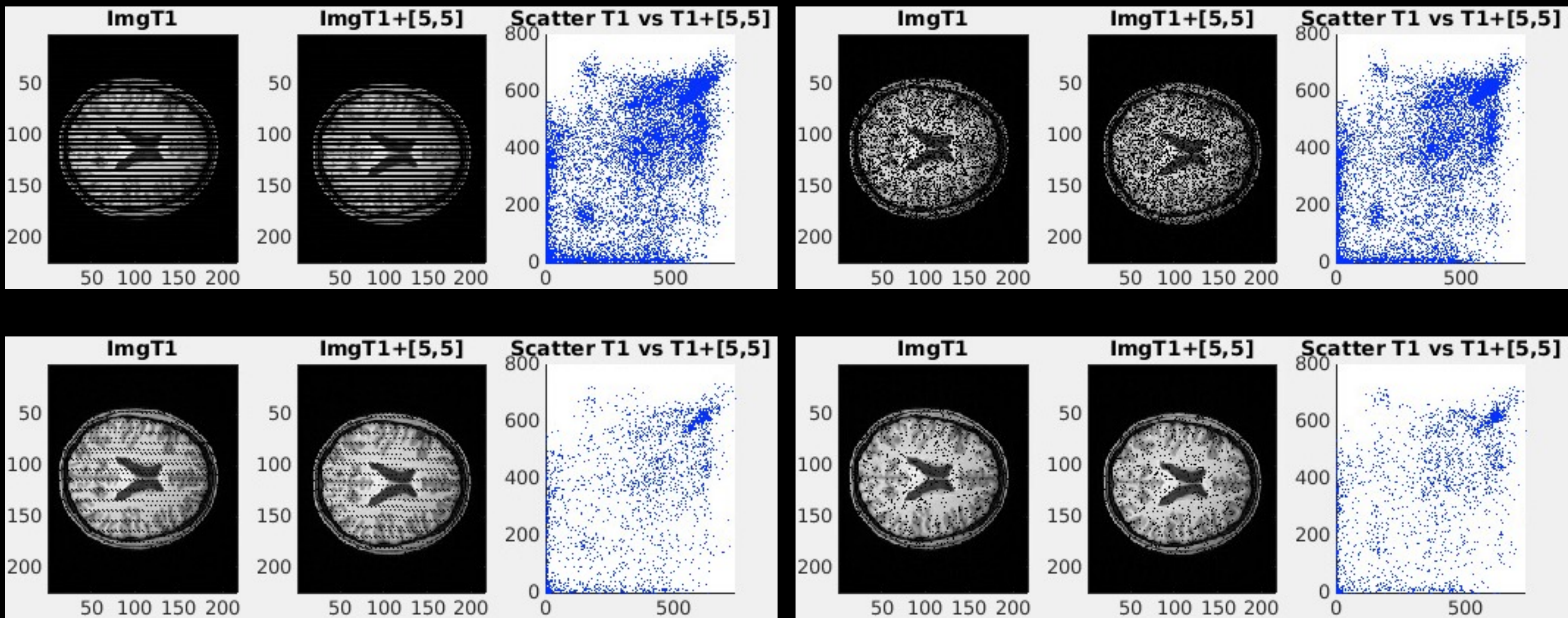
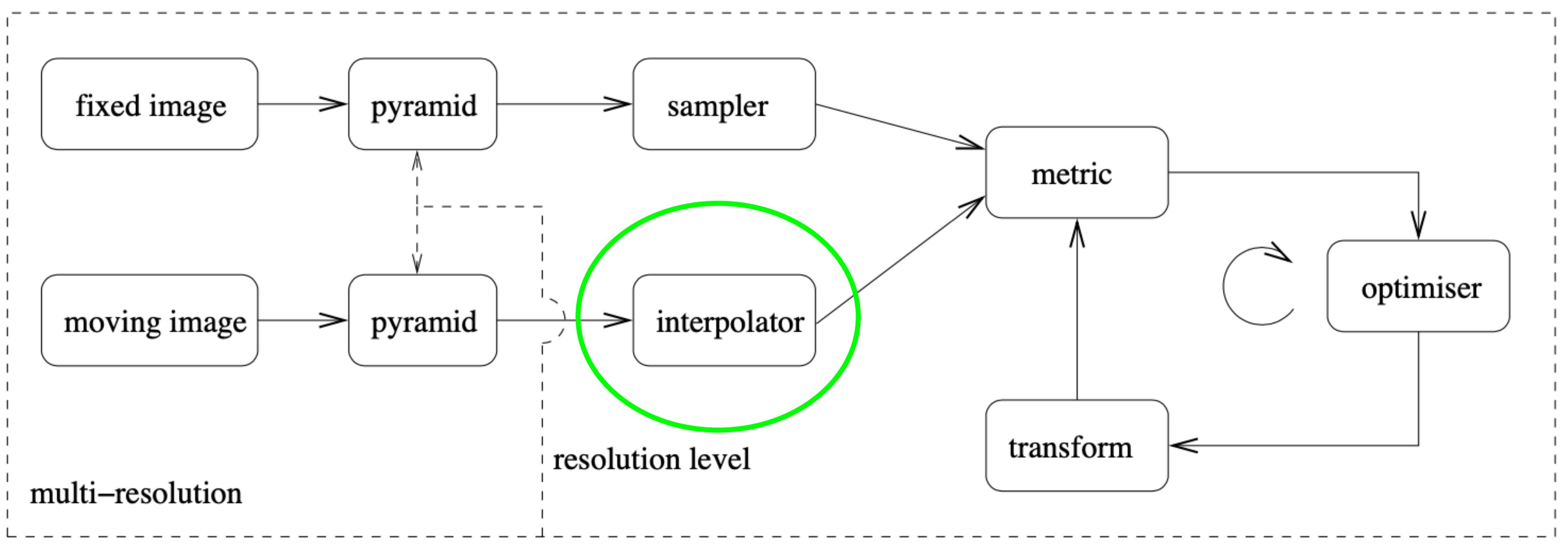


Image Registration pipeline

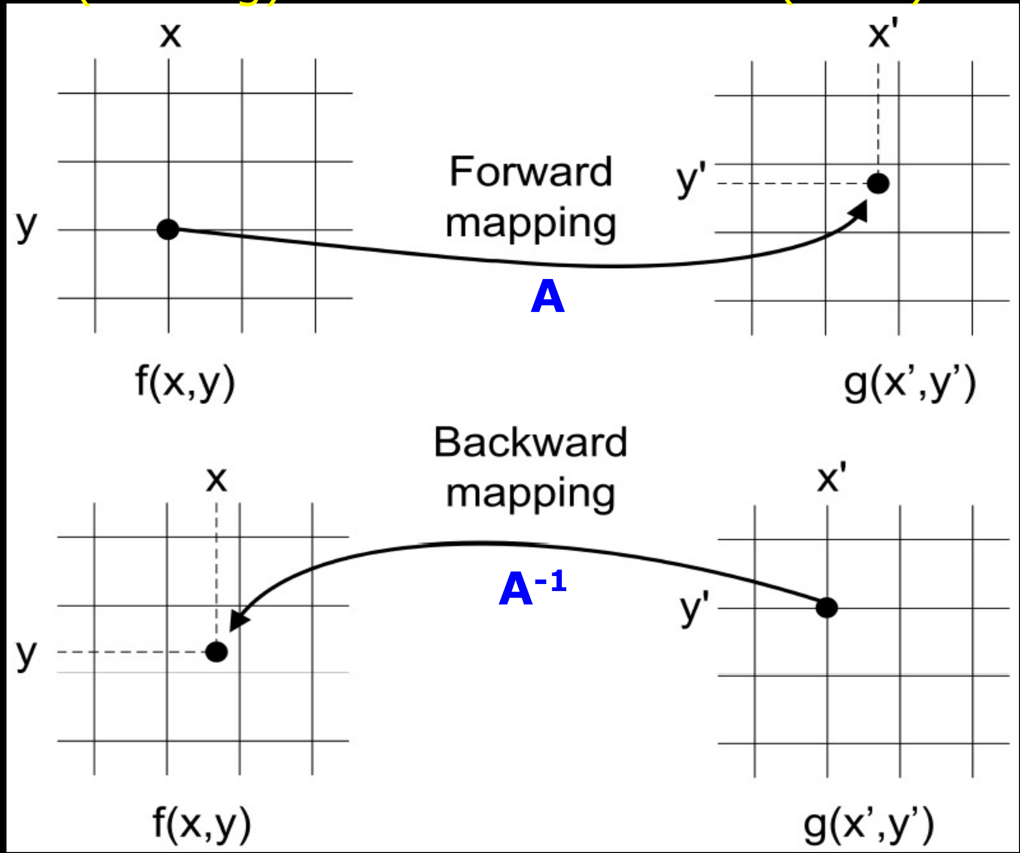
- Interpolation
  - To map the intensities from the template image to the grid of the reference image via a transformation matrix



A FLASH BACK to a previous Lecture:
Forward vs Backward mapping

- In a nutshell
  - Going backward we need to invers the transformation

Template (moving) Reference (fixed)



Interpolation methods

- Enhances structural boundaries
  - Higher-order interpolation methods: Reduce blurring
- May visually appear “sharper”
  - Do not change the image information!
  - Only if combining interpolated images w. different information of the same object – e.g. different angles of moving object e.g. car

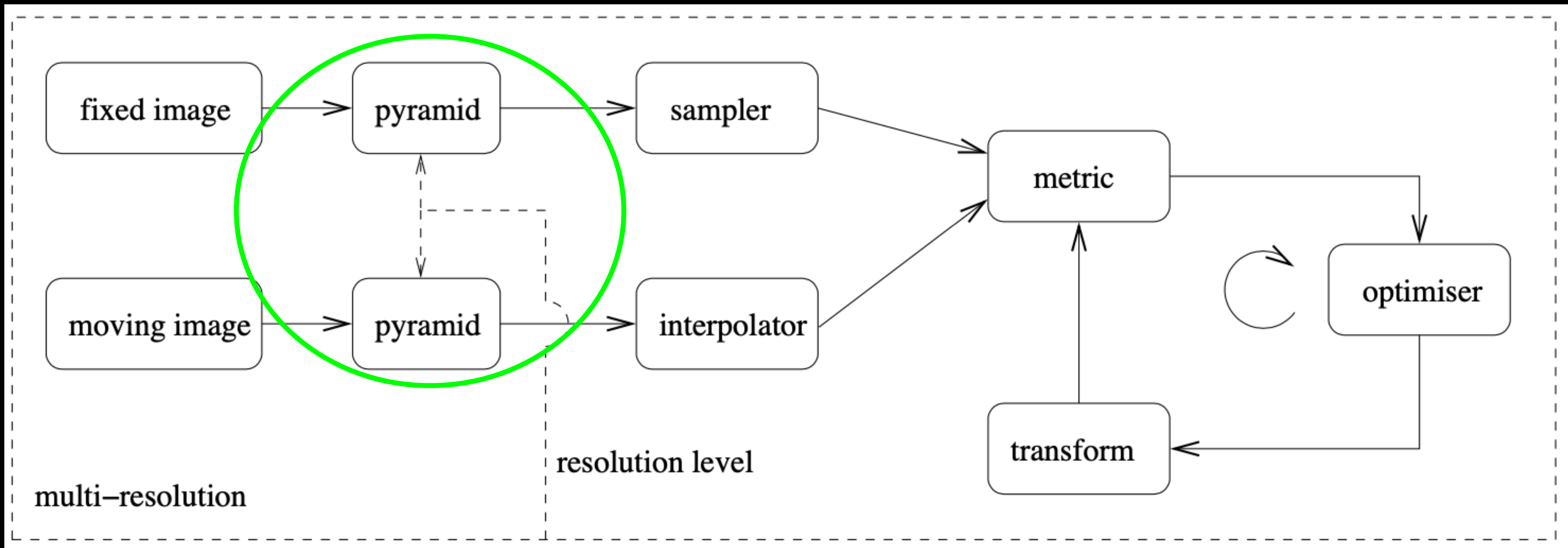
    → Super resolution (another topic)

(a) (b) (c) (d) (e)

Figure 2.4: Interpolation. (a) nearest neighbour, (b) linear, (c) B-spline $N = 2$, (d) B-spline $N = 3$, (e) B-spline $N = 5$.

Image Registration pipeline

- Pyramid



The Pyramid Principle

- To ensure robust image registration

Some stones?

Pretty close

Walking distance

From a bird

From space?

Very detailed

Good overview

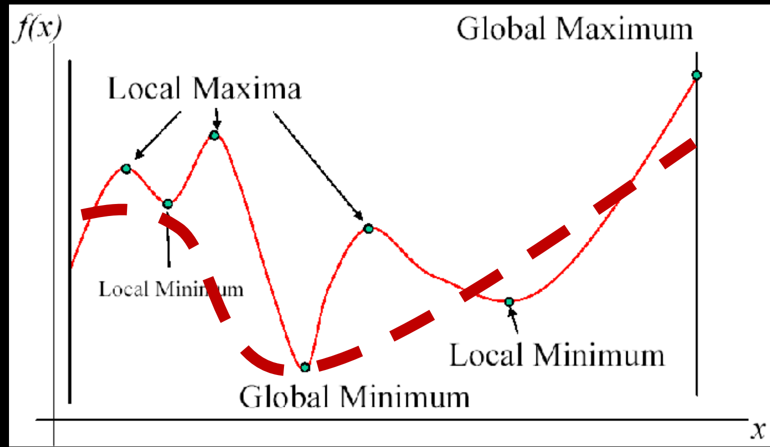
Too coarse

The Pyramid Principle

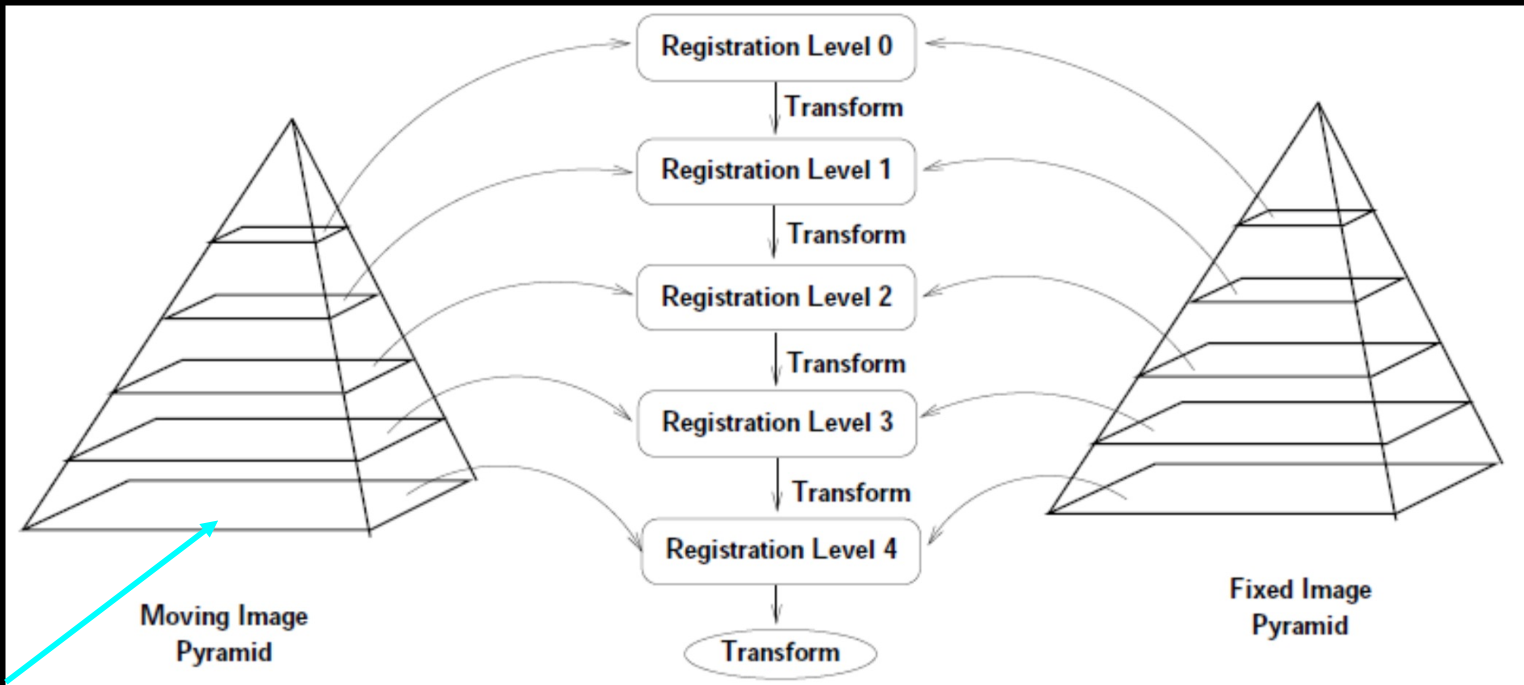
- To ensure robust image registration

The Pyramid Principle

- A Multi-resolution strategy
- To ensure robust image registration
  - To reduce local minima's
  - What is a prober image resolution level ?



Coarser structural details



Original resolution

The Pyramid Principle

- Lower image resolution
  - Down sampling (memory reduction, fewer data)
- Less structural details
  - Smoothing (Complex method settings become more general)

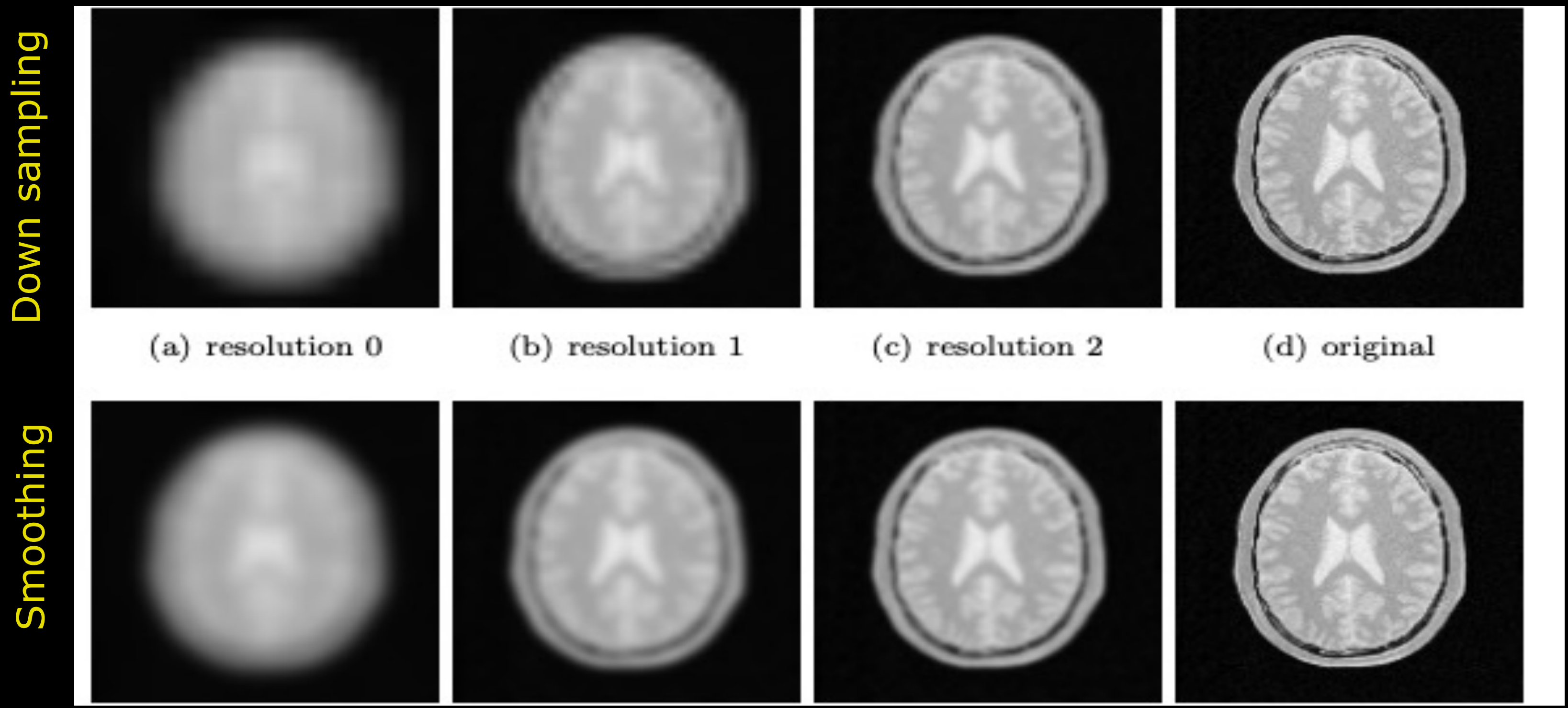
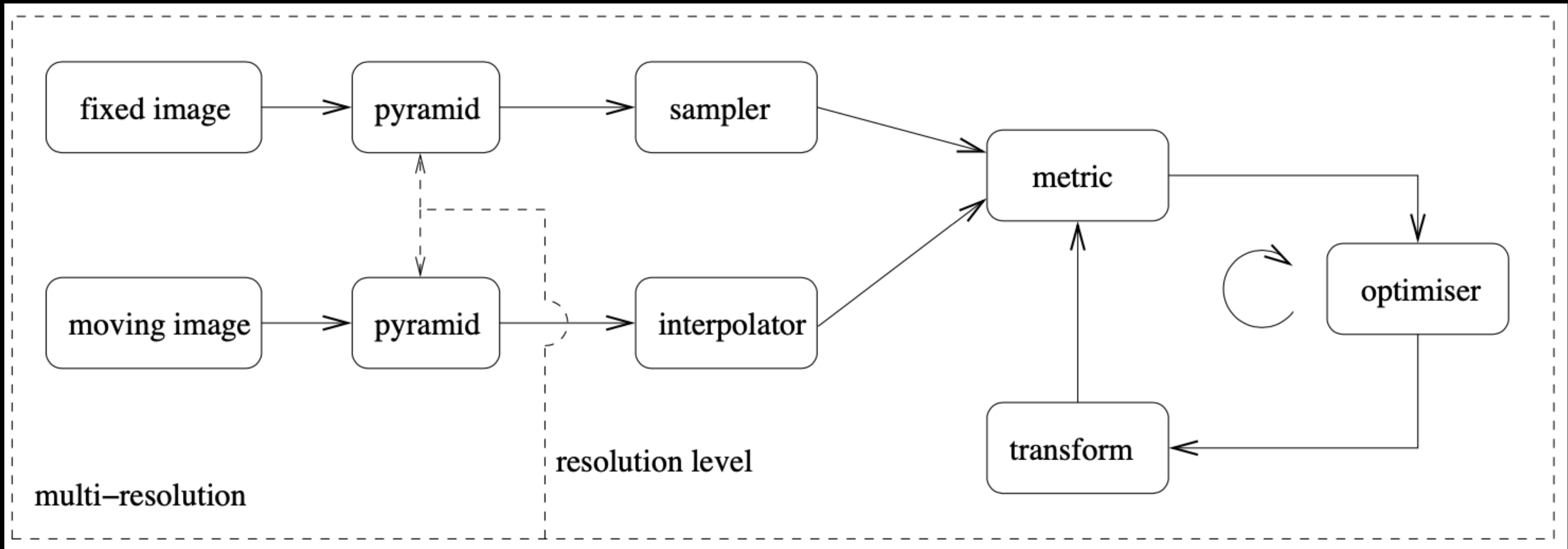


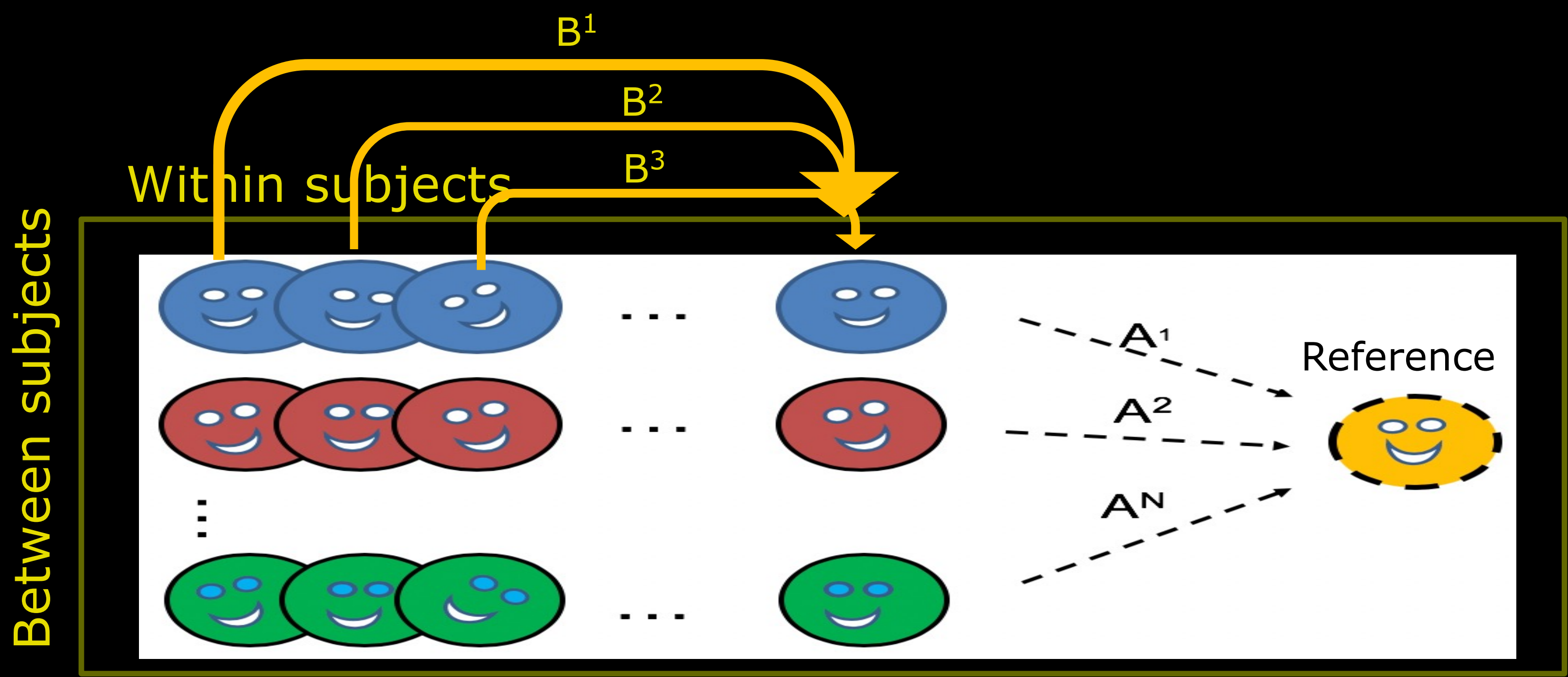
Image Registration pipeline

- At the end we just select an existing tool
- Still, we need how too select method settings ☺
  - This was the first step in the registration pipeline



Combining Image Registration pipelines

- First step : Within subjects (Same structure + temporal)
- Second step: Between subjects (different structure+ temporal)
  - Can use an iterative procedure to improve registration
- Combine subject-wise transformation metrics by multiplication
  - Apply only one interpolation at the end to minimise blurring



Quiz 6: Quality inspection - How

How to quality assurance (QA) the image registration results?

A) Use a similarity measure
B) Visual inspection
C) No need it to - just works
D) Sum of square difference
E) Search the internet for experience

Image Registration pipeline strategy

- Within subjects and between challenges
  - E.g. Histology 2D → 3D: Structural difference between slices
  - Visually inspect your results!!

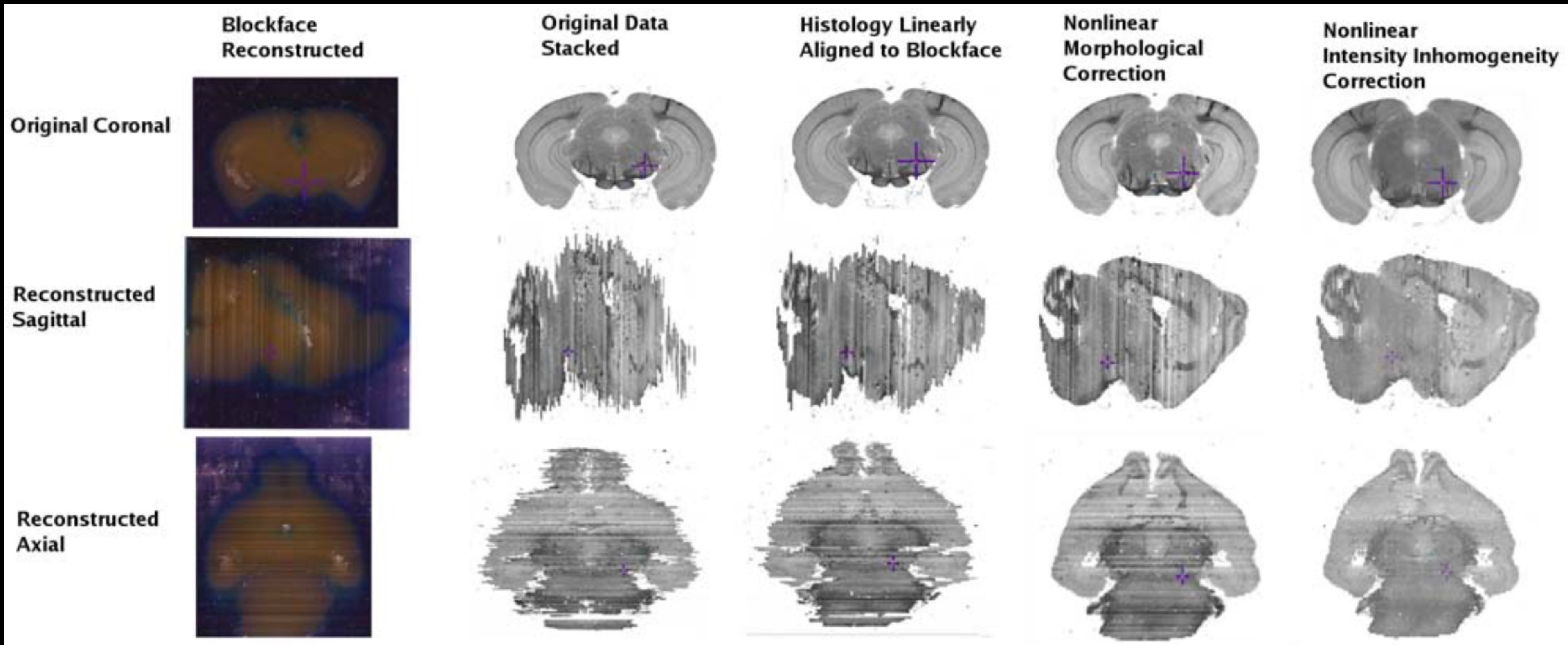
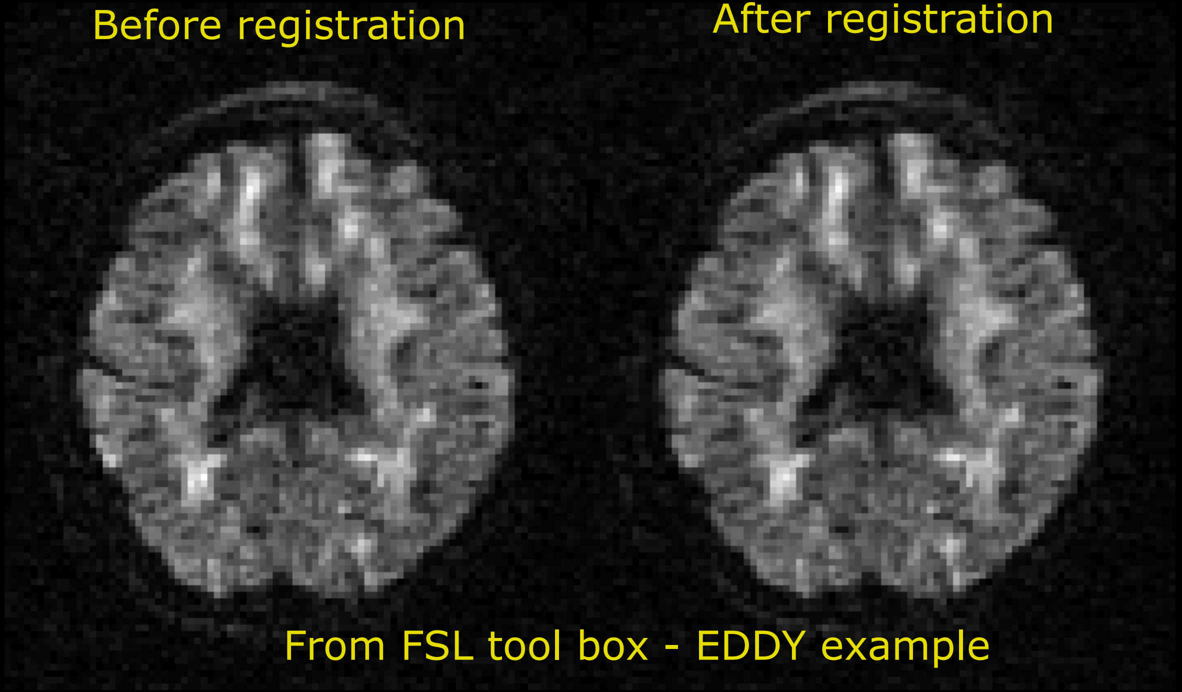


Image Registration pipeline strategy

- Within subjects across time points (temporal)
  - Remove image distortions + subjection motion
- Visually inspect your results!!



What can you do after today?

- Describe difference between a pixel and voxel
- Choose a general image-to-image registration pipeline
- Apply 3D geometrical affine transformations
- Define coordinate system of an object for 3D rotation
- Use the Homogeneous coordinate system to combine transformations
- Compute a suitable intensity-based similarity metric given the image modalities to register
- Compute the normalized correlation coefficient (NNC) between two images
- Compute Entropy
- Describe the concept of iterative optimizers
- Compute steps in the gradient descent optimization algorithm
- Apply the pyramidal principle for multi-resolution strategies
- Select a relevant registration strategy: 2D to 3D, Within- and between objects and moving images

Next week – Real-time face detection using Viola Jones method