IMAGE ANALYSIS

Rasmus R. Paulsen. Tim B. Dyrby. DTU compute

Class I (31/08)

Image Analysis → Automatic extraction of information from images
A sub-topic within

- Pattern recognition
- Machine learning
- Deep learning

Image processing → Changing the information in images, but not necessarily getting any knowledge.

- Photoshopping
- Changing the visual appearance of photos
- Cropping / rotating
- Filters /effects

Examples:

1. <u>Classical machine vision</u>
- Tomato sorting machine (good tomatoes vs. green/bad tomatoes)
- Combination of very fast cameras, robotics, fast classification algorithms.
2. <u>Face tracking</u> – all features including eyes
- For digital cameras/phones (automatic focus on the face + face beautification)
- Tracking and manipulation for apps (Messenger / WhatsApp / SnapChat…)
- Awareness tracking for car drivers (warning if you fall asleep)
3. <u>Self-driving cars</u>
- Modern self-driving cars rely on many sensors (lidar, GPS, accelerometers, stereo cameras or multiple cameras, sensor fusion)
4. <u>Sports tracking – human body tracking</u>
- Huge commercial impact
- Lots of research in human body tracking
- Personal trainers

Medical Image Analysis

Extraction of information from digital images

- Reproduce expert diagnosis: more accurate, variation between doctors' opinions removed
- Computed aided diagnostics (but doctors have the last word)
- Can enhance the signs of diseases (tumours, bleeding)

<u>Examples:</u>

1. Recognize and track the heart
2. Image based surgery planning
3. Cochlear implant planning
4. Shape changes in brain structures

<u>Relevance:</u>

Images are an important tool in

1. Diagnosis
2. Treatment
3. Follow-up

IMAGE TYPES

1. Digital Images

<u>Learning objectives:</u>

- Describe the fundamental properties of a digital images
- Read and show an image in MATLAB
- Describe the commonly used image coordinate systems
- Describe the binary, the label, the multispectral and the 16-bit image

23	216	120	55
4	89	158	130
65	76	189	34
19	234	7	45

It consists of pixels (picture elements)
Each pixel has a value between **0 and 255**.

Bit→ Tiny little switch that can be either 0 or 1 – the "memory of a computer" consists of insanely many bits.
Byte →8 bits together. It is the "basic" unit in a computer.
2^8 = 256 values

00000001 = 1
00000010 = 2
00000100 = 4

00001010 = 10
00001111 = 15
11111111 = 255

How many bytes do our image take up in the computer memory? 16 (16x16=256)

Grayscale digital images

0 is black and 255 is white. The values in between are shown as shapes of gray.
<u>Typical Example:</u>

- Traditional film X-ray
- Scanned on a flatbed scanner
- Bone is white and air is black (the more the radiation, the darker)
- Used for fractures, arthritis, osteoporosis

Image Resolution

It determines how much the image fills in the memory and on the hard disk

- **Spatial resolution** → related to the position, area and size of things.
  The number of pixels used to represent the image:
  256x256, 128x128, 64x64, 32x32, 16x16, 8x8
  How many pixels are there in the images from your camera/phone? 12MPixel

Width	Height	Pixels	Mega-pixels	Camera
320	240	10.000	0.01	Prototype 1975
1600	1200	1.920.000	2	Nikon Coolpix 950
4032	3024	12.192.768	12	Samsung Galaxy S7 edge
6240	4160	26.000.000	26	Canon EOS 6D M2
8984	6732	60.480.288	60.5	Phase One P65+

- **Gray level resolution** → the number of gray level in the image (256, 64, 16, 8, 4, 2)

Image as a matrix

$$\begin{array}{c|cccc} r \backslash c & 1 & 2 & 3 & 4 \\ \hline 1 & 23 & 216 & 120 & 55 \\ 2 & 4 & 89 & 158 & 130 \\ 3 & 65 & 76 & 189 & 34 \\ 4 & 19 & 234 & 7 & 45 \end{array}$$

An image is stored in the computer memory as a 2-dimensional matrix
4 rows and 4 columns (M rows and N columns)
As a discrete function f(r,c)= (row, column) system.
Origin is in the upper left corner, 1-based

MATLAB image I – what is I(2,3) ? = 158
What is the coordinates of the pixel with value 34? I(3,4)

In MATLAB, a pixel is stored as an **UINT8**
UINT8 = Unsigned 8-bit
Integer = 1 byte

$$\begin{array}{c|cccc} y \backslash x & 0 & 1 & 2 & 3 \\ \hline 0 & 23 & 216 & 120 & 55 \\ 1 & 4 & 89 & 158 & 130 \\ 2 & 65 & 76 & 189 & 34 \\ 3 & 19 & 234 & 7 & 45 \end{array}
\qquad
\begin{array}{c|cccc} y & & & & \\ 3 & 23 & 216 & 120 & 55 \\ 2 & 4 & 89 & 158 & 130 \\ 1 & 65 & 76 & 189 & 34 \\ 0 & 19 & 234 & 7 & 45 \\ \hline & 0 & 1 & 2 & 3 \; x \end{array}$$

However, in many graphics' programs (photoshop, etc.) the origin is upper left corner, 0-based. (X,Y) system.
When using plotting (MATLAB plots, mathematics), origin in lower left corner, 0-based. (X,Y) system.

Maximum value		
MATLAB system	1-based, origin: upper left corner	(r,c) = (4,2) = 234
(X,Y) system	0-based, origin: lower left corner	(x,y) = (1,0) = 234

General Conversion: | x = c-1 | y = M-r | M= # rows

Image Histogram

It normally contains the same number of "bins" as the possible pixel values
A bin stores the number of pixel with that value
256 gray levels in the image = 256 bins in the histogram
The shape of the histogram tells us something about the image

It can be seen as a function **h(v),** where v is the pixel value
Total number of pixels is the sum of all h

<u>Example</u>: Pick a random pixel in the image. What is the probability of it having value 3? P(v=3)?

h(3) = 10   # pixels = 36

P(v=3) = 10/36 · 100% = 28%

Normalized histogram

It is made by dividing each bin count with the total number of pixels
H(v) is the normalized histogram function, H(v) is the probability that a random pixel has value v
Equal to a probability density function

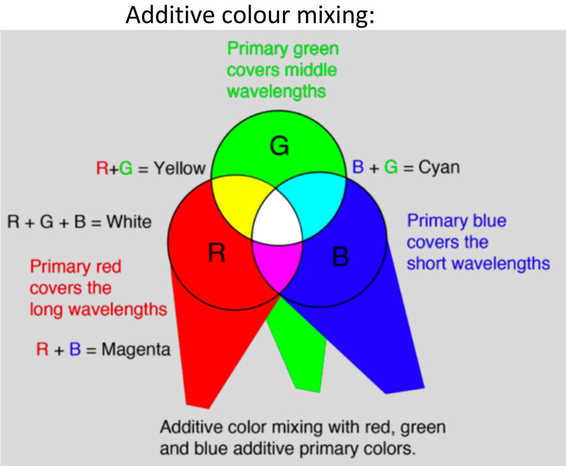
2. Colour Images

RGB → Red, Green and Blue
Television, computers, digital cameras used the "RGB colour space"
Additive colours: Final colour is made by mixing red, green and blue.
Typically the values of R,G and B lie between 0 and 255 = total 3 bytes.

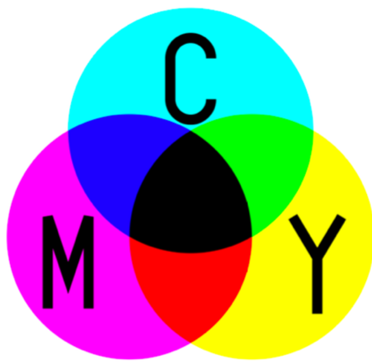
Examples:

RBG	colour
(0,0,0)	black
(255,255,255)	white
(255,0,0)	red
(0,255,0)	green
(0,0,255)	blue

Additive colour mixing:



Subtractive colour mixing:



Processing RGB images

- Each pixel in a colour image contains 3 values
- Equal to a "vector function" in mathematics
- More complicated to analyse
- Medical images are typically grayscale. Why? They just measure physical phenomena.
- Often images are converted from colours to grayscale before the analysis.

$$\boxed{V = 0.2989 \cdot R + 05870 \cdot G + 0.1140 \cdot B}$$

Is it then possible to convert a grayscale image back to a colour image?
Not able to go back as there are too many variables mathematically. However, by using AI, one might be able to predict and train the algorithm to convert grayscale images back to colour ones.

3. Binary Images

Binary means **ON or OFF** → Two colours' images.

Background	0 = black
Foreground	1 = white

Obtained by using **threshold** or classification of classes.

Examples:
- Separating objects from background
- Count the number of objects
- Measure the size and shape of objects
- Advanced 3D visualizations
- From CT scanning to “Bone Image”.

4. Label Images

The pixel value tells the **type** of the pixel:

Example of 4 different pixel value, where colours are made using a ***look-up-table:***

(0) Gray	Background
(1) Blue	Soft tissue
(2) Green	Hard bone
(3) Red	Spongy bone

Used in label classification, in deep learning:
- Medical images: Tumour (volume/percent)
- Bone density
- General anatomy recognition: blood vessels, calcifications.

5. Multispectral Images

- There is more visual information that what can be seen with the human eye.
- Standard cameras capture the red, green and blue colours.
- Capture systems that capture more bands and other frequencies exist.
- Creates multispectral images → Each pixel contains perhaps 20 values from different spectral bands.

Example: Multispectral System – VideometerLab
- Integrating sphere
- Light emitting diodes with different wavelengths
- High resolution camera
- Water in bread
- Classification of fungi
- Skin diseases

6. 16-bit Images

- 256 values fine for the human eye
- Pixel values not only for display (Physical meaning)
- Computed Tomography (X-ray attenuation)
- Hounsfield units

0	Water
1000	Air
120	Fat
400+	Bone

PCA Analysis

<u>Learning objectives:</u>

- Describe the concept of principal component analysis
- Explain why principal component analysis can be beneficial when there is high data redundancy
- Arrange a set of multivariate measurements into a matrix that is suitable for PCA analysis
- Compute the covariance of two sets of measurements
- Compute the covariance matrix from a set of multivariate measurements
- Compute the principal components of a data set using Eigenvector decomposition
- Describe how much of the total variation in the data set that is explained by each principal component

PCA Example: Iris Data Matrix

- One column is one flower
- One row is all measurements of one type (4 different measurements of the flower)

$$\begin{bmatrix} Sepal\ length_1 & \dots & Sepal\ length_{50} \\ Sepal\ width_1 & \dots & Sepal\ width_{50} \\ Petal\ length_1 & \dots & Petal\ length_{50} \\ Petal\ width_1 & \dots & Petal\ width_{50} \end{bmatrix}$$

- The measurements can be used to:
    - Recognize a species of flowers
    - Classify flowers into groups
    - Describe the characteristics of the flower
    - Quantify growth rates
    - ...

- Some measurements are redundant, can be combined or boiled down.

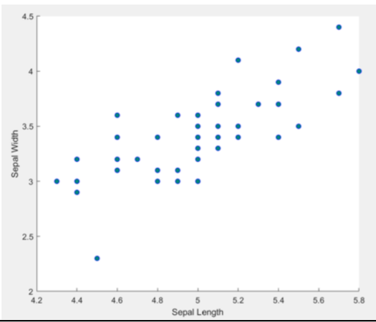
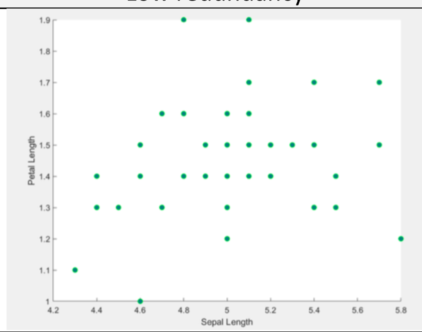
Variance

$$\sigma^2 = \frac{\sum (x - \overline{x})^2}{n}$$

σ^2_{SL}	0.1242
σ^2_{SW}	0.1437
σ^2_{PL}	0.0302
σ^2_{PW}	0.0111

Redundancy

It is measured by the **covariance** = relationship between measurements

<table>
<tr><th>High redundancy</th><th>Low redundancy</th></tr>
<tr><td>
</td><td>
</td></tr>
<tr><td>We can explain quite a lot of the sepal width if we know the sepal lengths.
Just knowing the length, we can predict the width.
Correlated measurements → high covariance</td><td>We can NOT explain the petal length if we know the sepal length.
Uncorrelated measurements → low covariance</td></tr>
</table>

Example:
a_i = {5.1,4.9…,5}
b_i = {3.5, 3…,3.3}

$\sigma^2_{a,b} = \frac{1}{n} \sum_i a_i\, b_i$ = 17.25

Vector notation:

- Vector notation for covariance: $\boxed{\sigma^2_{a,b} = \frac{1}{n}\, a\, b^T}$

- Matrix notation for covariance: m x n matrix (m=4 and n=50)

$$\begin{bmatrix} Sepal\ length_1 & \dots & Sepal\ length_{50} \\ Sepal\ width_1 & \dots & Sepal\ width_{50} \\ Petal\ length_1 & \dots & Petal\ length_{50} \\ Petal\ width_1 & \dots & Petal\ width_{50} \end{bmatrix}$$

Covariance matrix autopsy

$$\boxed{C_X = \frac{1}{n}\, X\, X^T}$$

m x m square matrix (m=4) → Observation data matrix (symmetric)
The diagonal elements of C_X are the variances.
The off-diagonal elements are the covariances.
The Principal Components of X are the eigenvectors of C_X.

Goals

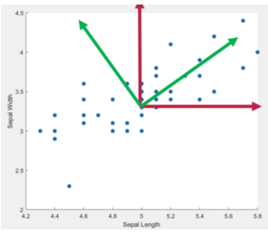
- Minimize redundancy → covariances should be small
- Maximize signal → variances should be large
- Signal to noise ratio (SNR) → all noise as low as possible, thus SNR should be large

$$\boxed{SNR = \frac{\sigma^2_{signal}}{\sigma^2_{noise}}}$$

Steps

1. Changing basis

- Subtract the mean → Centering data → New coordinate system
  The new coordinate system (green) follows the **covariance** in the data



Green system
x: signal
y: noise

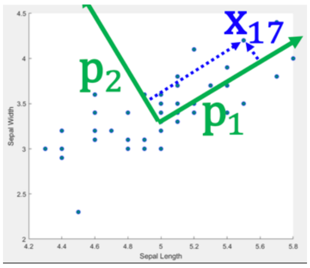
2. Find the measurement values in the new basis. We need to project the points to form the other coordinate system. Transform our data (rotating and scaling the basis)

- The **dot product** projects a point down to a new axis

$$X_{17,new} = x_{17} \cdot p_1$$

$$\boxed{\boldsymbol{Y = PX}}$$

where p_1 and p_2 are the rows of P and **Y** contains the new coordinates/measurements per sample
P are the principal components

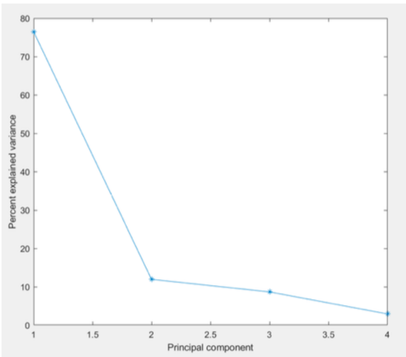


3. The new covariance matrix:
   It should be as diagonal as possible.

$$\boxed{C_Y = \frac{1}{n} Y Y^T}$$

   The i'th diagonal value of C_Y is the variance along principal component number i.

4. Explained Variance
   If one component explains 75% of the total variation, we can have one number that explains most of all measurements.



It can be used in classification problems (based on one number instead of 4)

Class II (07/09)

Image acquisition, compression and storage

1. Mapping brain network and microstructure using Magnetic Resonance Imaging (MRI)

Function units (Gray matter) → parts of the brain that communicates though brain connections (axons)
Thanks to MRI scans (Human 3T MRI), we can build a model of the **brain network.**

The challenges:

- Weakly correlating with clinical tests
- Clinical MRI is very sensitive to anatomical changes but often lacks specificity.

MRI Acquisition: *(Human 3T MRI scanner)*

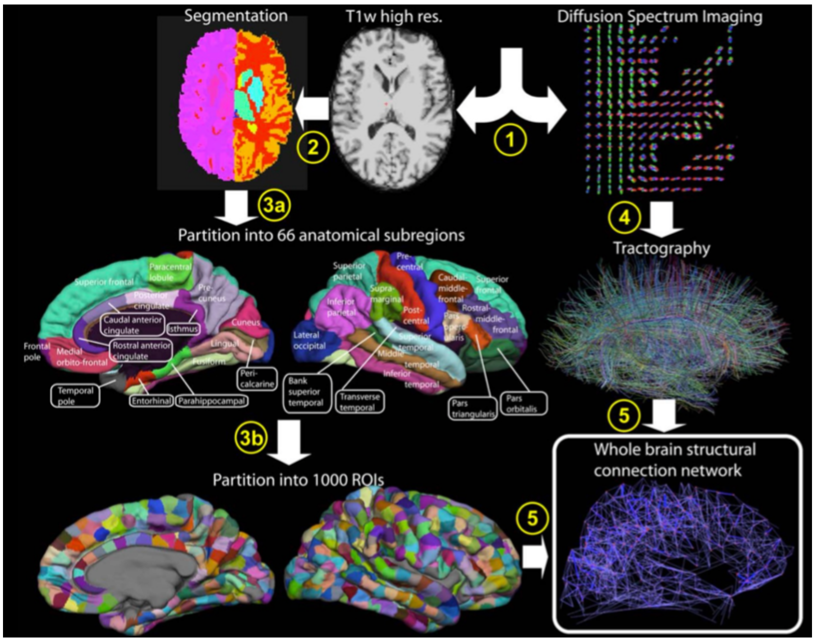


Image Analysis:

- Image acquisition
- Data storage
- Point wise operations (histograms)
- Segment anatomical regions
- PCA analysis
- Design biophysical mathematical models of the brain network

2. Image acquisition

Light

- Light can be seen as a source of **electromagnetic waves** with specific **frequencies f** (measured in Hertz = Hz) or as a photon *(from Greek phōtos, "light"). (Mass less fundamental particle).*
- It has a **wavelength** $\boldsymbol{\lambda}$ (measured in meters = m)
- It has **speed c** = 299.792.458 (m/s)

$\lambda = \frac{c}{f}$	High frequency → short waves
	Low frequency → long waves

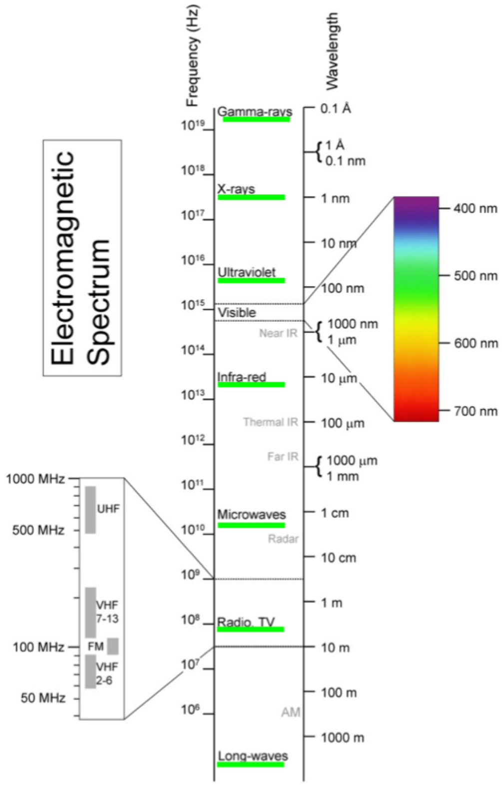
- Light has energy, using the Planck's constant

$E = h \cdot f$	High frequency → high energy
	Low frequency → low energy

Electromagnetic spectrum
(range of all frequencies, divided into 7 regions)

Wavelengths:
$1\mu m = 0.001 mm$
$1nm = 0.000001 mm$

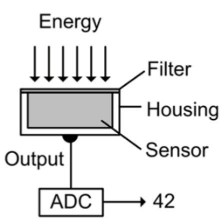
The one with most energy:
→ *X-rays*

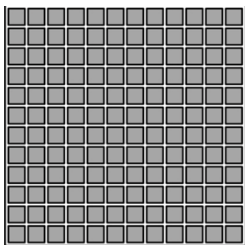


CCD

How do light become a digital image?

Single cell





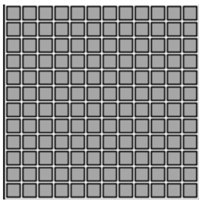
Charged coupled device (CCD-chip)
The digital film

- The CCD cell can be seen as a well that collects energy for a limited time (to be charged)
  - Exposure time
  - Integration time
  - Shutter
  -
- Energy is transformed to a digital number thanks to Analog-To-Digital converter **(ADC).** It takes an "analog signal" and converts it to a digital signal.
  Ex: $(01001001)_2 = (0 \times 2^7) + (1 \times 2^6) + (0 \times 2^5) + (0 \times 2^4) + (1 \times 2^3) + (0 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) = (73)_{10}$

- 1 CDD = 1 pixel (only for grayscale images, more complex for RGB images)
  Ex: 10 MPixel camera → 10 million analog to digital conversion for one image

Quiz: What is the size of a single CCD cell?

5.3mm



7.2 mm

2048 x 1536 pixels

.

7.2/2048 = 3.5μm
5.3/1536 = 3.5μm

- To collect energy:
  The shutter opens and the CCD is hit by light → CCD integrates light → The shutter closes
  The collected energies are transferred and converted to digital values by the ADC and stored in the memory of the camera.

- What happens if we integrate over…

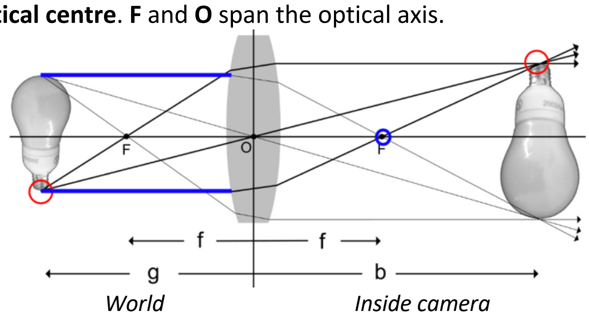
Long integration time	Short integration time
- Motion blur: It causes blurring of the moving object - Over-exposure (the well is overrunning) - Blooming	- Noise - Lack of contrast

- However, the camera is more than a CCD. The CCD is just the sensor.
  There is also "an optical system".

  Optical System:
  How do we get an image on the CCD?
  - Light follows a straight line
  - Light that hit one spot reflects in many directions. Thus, the same point hit by rays from all over the object.

  Pinhole camera:
  The optical system has a barrier with tiny hole. The light is coming through the tiny hole. The problem is that we have very little light.
  How do we get more light inside the camera, while keeping the focus? By using **lens.**

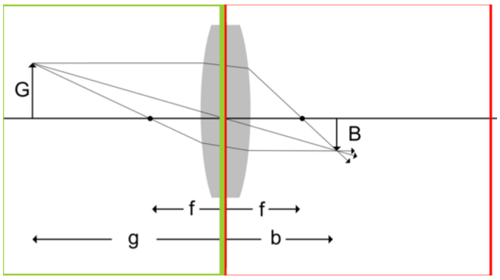
  Lens:
  - A lens focuses a bundle of rays to one point.
  - Parallel rays pass through a ***focal point F*** at a **distance f** beyond the plane of the lens (**f** is the **focal length**).
  - **O** is the **optical centre**. **F** and **O** span the optical axis.



Focal point – focal length

Light coming from “really far away” can be seen as parallel rays.
Then the rays intersect at the focal point.
Focal length f = distance from optical centre O to focal point F

Where do non-parallel rays meet?



World *Camera*

Thin lens equation or Gauss’ lens equation

$$\frac{1}{g}+\frac{1}{b}=\frac{1}{f}$$

g - distance to object
b - distance to intersection

<u>*Quiz:*</u> *Where do the rays meet in the camera?*
Camera with focal length of 5mm
Rasmus standing 3 meters away (3000mm)

$$\frac{1}{b}=\frac{1}{f}-\frac{1}{g} \quad \rightarrow \quad b=\frac{f\cdot g}{(g-f)} \ = \frac{5\cdot 3000}{(3000-5)} \sim 5mm$$

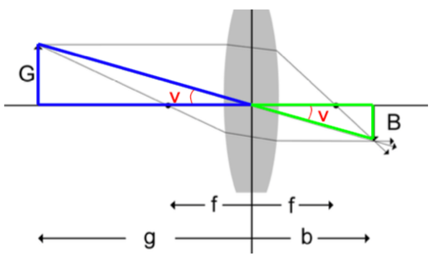
How do we make focused images?
Placing the CCD right → CCD should be placed at **b.**

<u>Focusing:</u>

- We move the camera
- Distance to object (**g**) **changes**, but **f** is **fixed.**
- **B changes** → Moving CCD to b → ***FOCUSING***

<u>Object size:</u>
What is the size of an object on the CCD?

Important relation



- Two triangles
- One side length g and one with b
- B and G are related

g - distance to object
b - distance to intersection
G - Object height
B – object height on CCD

$$\frac{b}{B}=\frac{g}{G} \rightarrow \ B = b\cdot\frac{g}{G}$$

Zoom:

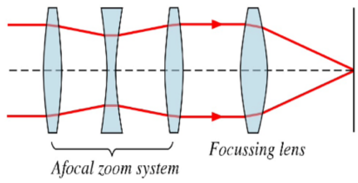
*How do we **Zoom** then?* → *By making **B larger**. (Changing B)*

!!! By changing B → We change the focal length.

$$B = \mathbf{b} \cdot \frac{g}{G} \quad \rightarrow \underbrace{\frac{\mathbf{1}}{\mathbf{g}}}_{\textit{constant}} - \frac{1}{b} = \frac{\mathbf{1}}{\boldsymbol{f}}$$

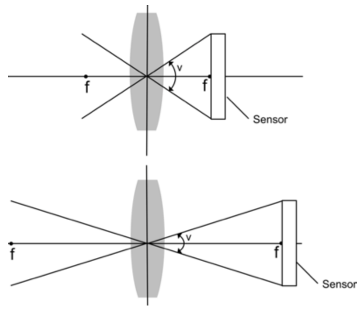
But…changing the focal length:

- Not possible on a simple lens
- Need a “zoom lens”
- Several lenses together



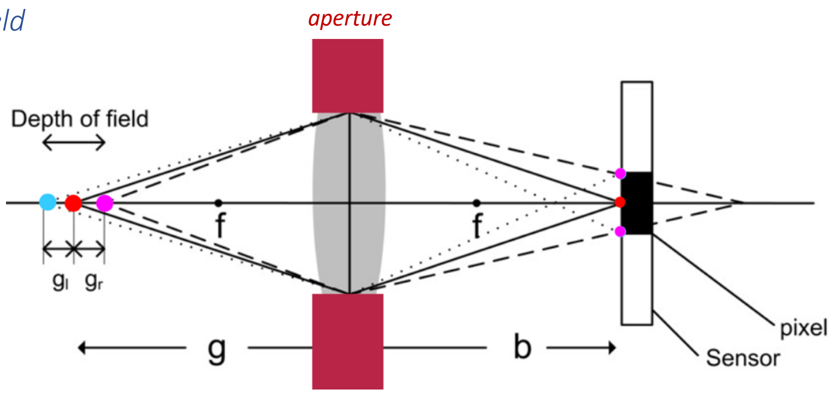
Field of view (FOV)

- It is described by an angle: large angle → the larger FOV
- It depends on CCD size and focal length
- Fisheye lens, with small focal length and large field of views
- CCD chip is a rectangle:
    - Horizontal field of view
    - Vertical field of view
- Zoom changes fields of view
    - Optical Zoom
    - Digital Zoom



(*two cameras with different focal length)*

Depth of field



- Look at one pixel in the middle.
- The object is place at distance g

→ **How much can we move the object?**

Light must hit the same pixel.

- Move it to the left (g_l)
- Move it to the right (g_r)– still hit the same pixel, but twice.

The *aperture* controls the amount of light.

With a small aperture: We have a large depth of field → less light → longer exposure

Summary: To acquire a good image. It matters:
Distance to object, motion of the object, Zoom, focus, depth-of-fields, focal length, shutter, field-of-view, aperture, sensor (size and type).

3. Image storage

How? Though hard disks, memory cards, CDs etc.

- Storage for bytes. (With 500 GB → 500.000.000.000 bytes)
- Stores data as lists of bytes
- A file on a hard disk: It has a length (in bytes, MB, GB) and it contains numbers (bytes).
- Images as data. Each pixel is a byte → A byte is made of 8 bits.
  So, we need the size of the image and the data.
    - Width as 2 bytes (0-65535)
    - Height as 2 bytes (0-65535)

Simple image format: Windows Bitmap Format (BMP)
It stores the image as

- A header with information about size
- Data with no compression

Compression: Look for patterns to represent the data more "compact"
→ The count and the value = **Run length encoding**

- Simple but useful data compression
- General, not only for images
- It also used by the Windows Bitmap Format (BMP)

Quiz: Run Length coding of images

1	5	5	5	3
3	2	3	3	201
201	19	19	19	147
147	130	130	130	130
147	147	147	88	88

1 1 3 5 2 3 1 2 2 3 2 201 3 19 2 147 4 130 3 147 2 88

Compression ratio: It gives a measure of how much data/image is compressed.
Ex: From 16 to 12 → 16:12 = 4:3 → Ratio: 1.33

$$Compression\ ratio = \frac{uncompressed\ size}{compressed\ size}$$

Image formats

Lossless image formats	Lossy image formats
-Do not throw away information -Good for storing medical images, as we do not want to destroy any information - Not very effective for photos. Why?	-Removes "unimportant" information -Removes the "high frequencies" -Similar to the MP3 sound format *Ex: JPEG (JPG)*

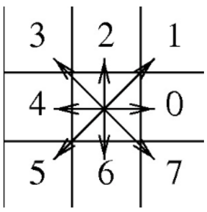
Too many changes in the images that leads to noisy images. *Ex: PNG (portable networks graphics)*	

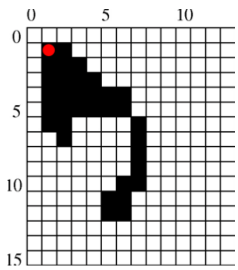
<u>Compression artefacts:</u>

- Lossy compression changes the image
- Normally not a problem for photos
- Big problem for medical images
    - Mammogram: As we are looking for tiny bright spots, it would be changed by lossy compression.
    - Thus: Use JPEG (JPG) for photos only.

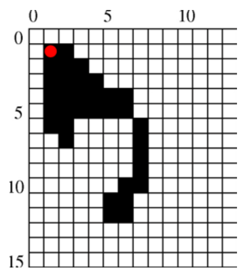
Binary Images

- Binary (means on or off): two colors:
    - Background (0 = black)
    - Foreground (1= white)

<table>
<tr><th colspan="2">Representation</th></tr>
<tr><th>Chain coding</th><th>Run length coding</th></tr>
<tr>
<td>
- Sufficient to describe the foreground.
- Background given by the foreground.
- The coordinates of the starting pixel is stored.
- Secondly the sequence of step directions is stored.


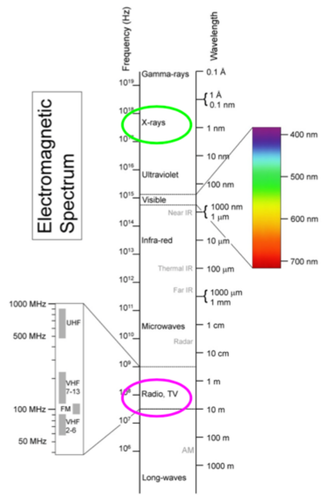
<u>Ex:</u>
(1;1) (0777067666656421122223444456322222)


</td>
<td>
- The foreground is described
- Each line of the image is described
- For each “run” the row number is stored
- The start column and end column are stored.

<u>Ex:</u> [row; (from column; to column)];
[1; (1;2)]; [2; (1;3)]; [3; (1;4)]; [4; (1;6)]; [5; (1;6)]; [6; (1;2) (7;7)]; [7; (2;2) (7;7)]; [8; (7;7)]; [9; (7;7)]; [10; (6;7)]; [11; (5;6)]; [12; (5;6)];


</td>
</tr>
</table>

Beyond reflective light



- X-Ray Imaging
    - High-energy light
    - Including:
    - Computed Tomography (CT)
      (Medical scanner for hard tissue)
    - Synchrotron light
      (High brilliance x-Ray, nano-scope for soft and hard tissue)
- Magnetic Resonance Imaging
    - Radio frequency
    - Medical imaging (soft tissues)

X-Ray Images

It is one of the most used forms of medical imaging, with a high-energy light. Simple, cheap and fast, but involving radiation.

It was invented by *Wilhelm Conrad Röntgen,* a German physics professor experimented with a *Crookes tube.* He discovered that an unknown ray could be captured on photographic plates, and named them X-rays (others, call them: Röntgen-rays). First, he had no idea they were dangerous and made an X-Ray of his wife’s hand. (The first medical X-Ray on 22 December 1895)

X-Ray became popular extremely fast: shoe fitting, wedding pictures, examination of your bones in coin machines… There were x-ray clinics in small normal apartments.

Then, people started to realise that exposure to X-rays could be dangerous.

<u>Electromagnetic spectrum:</u>

- Wavelength: 10 pm < λ < 10 nm
- Small wavelength → High energy

How to generate light

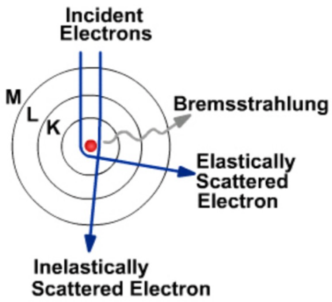
The light is generated by emitting photons.

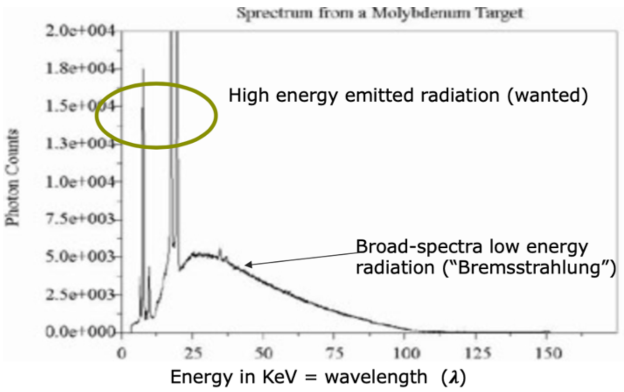
A) Make a beam of accelerated electrons:
- Close to the speed of light
- High energies: kilo electron volts (keV)

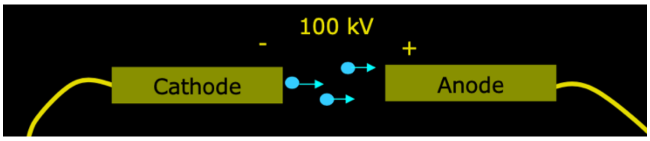
B) Emitting photons
   1) Incoming electrons excite the atom. The electron jumps to higher energy level. Fall back to original energy level – release energy → emit a photon.
   2) De-acceleration of electron → emit a photon.

<u>Production of X-Rays:</u>

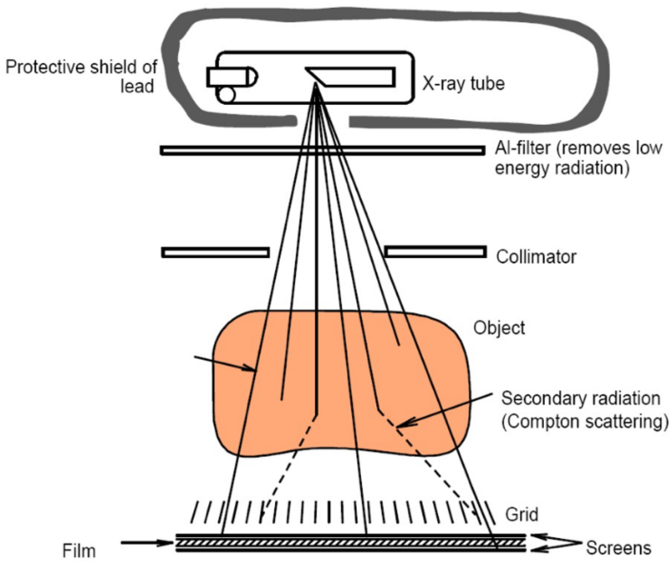
- The electrons are accelerated using a cathode. The heating up a filament releases more electrons.
- Some hit the anode (the heavy metal target)
- The electrons slow down in the anode material:
    - Generating heat
    - A small part of the energy is transformed to X-rays
- The electron comes very close to the nucleus
    - The electromagnetic interaction causes a deviation of the trajectory.
    - The electron loses energy, and an X-ray photon is emitted.





Electron volts: → 1 eV is the energy increase that an electron experiences, when accelerated over a potential difference of 1 V. In medical imaging: 20 keV < E < 150 keV 
Energy of the radiation vs. the wavelength: $E = \frac{h}{\lambda}$ [eV] , h: Planck constant

<u>Full X-Ray system:</u>



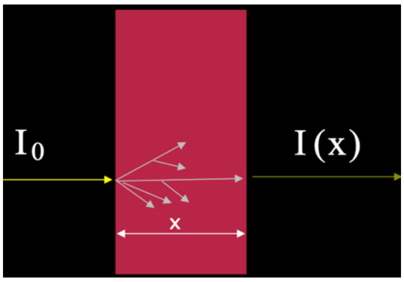
Contrast in X-ray images:

There are some materials/tissues that absorb more X-rays than others. X-Rays “got through”.

- Dark area → high radiation (air, fat, soft tissue)
- Bright area → low radiation (metals, bone)

X-ray attenuation:

X-rays hits an object and travels through it.



Io: intensity at the entrance

I(x): what is left on the other side, after a length of x.

The rest disappears in several different ways.

The attenuation is computed by using Lambert-Beer’s law:

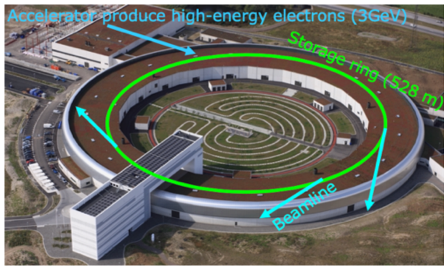
$$I(x) = I_0 \cdot \exp(-\mu x)$$

(Different materials have different attenuation coefficients)

CT scanning – imaging by sections

- 512 x 512 pixels per slice
- Many projections
- Image reconstruction
    - Enormous system of equations
    - Find each pixel attenuation coefficient
    - Hounsfield Units (HU). Calibrate units in medical imaging.
    - AIR: 100
    - Water: 0
- Not solvable by direct methods

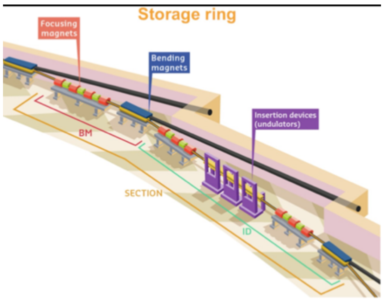
Synchrotron radiation

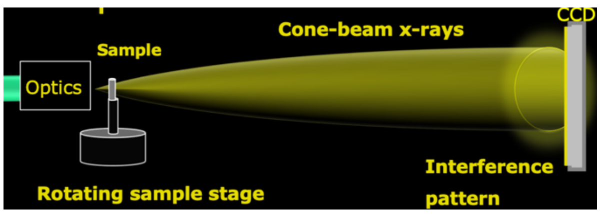


- It is a very homogenous electron beam
- Large scale research facility (MAXIV in Lund, Sweden)

The storage ring:

- Segment-wise linear
- Undulators: magnets force electrons to follow a wavy trajectory, thus, improving the brilliance of the beam
- Bending magnets: electrons deflected from their straight path emit x-ray tangentially (synchrotron light)

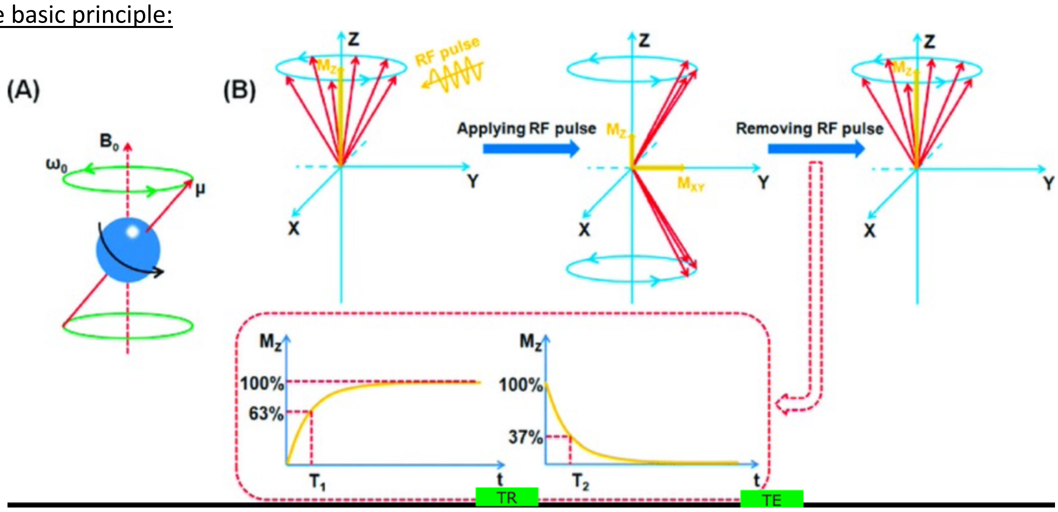


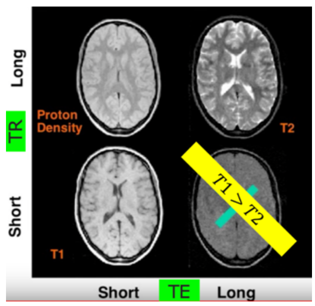


Magnetic Resonance Imaging (MRI)

- Magnetic and radio frequency in mega Hz.
- Soft tissue + brain function
- Expensive compared to CT
- 3D imaging
- No documented danger
- Volume pixel: Voxel
- Clinical voxel sizes 3 to 1 mm^3 but can detect microstructures using biophysical models.
- Application: Having a 3D structure of the whole body (10 min) → know how it it's the fat distributed to the whole body. We can see the tissues.
- Magnetic field: preclinical MRI (7T), Human (1.5T, 3T, 7T).

The basic principle:

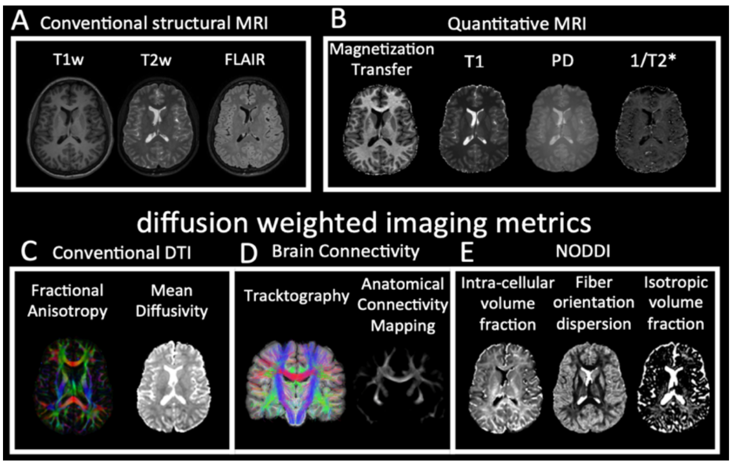




Signal contrast:

$$M_0 = \left(1 - e^{-\frac{TR}{T1}}\right) e^{-\frac{TR}{T2}}$$

Multi-modality of the same subject: → Long-scan times: High risk of motion

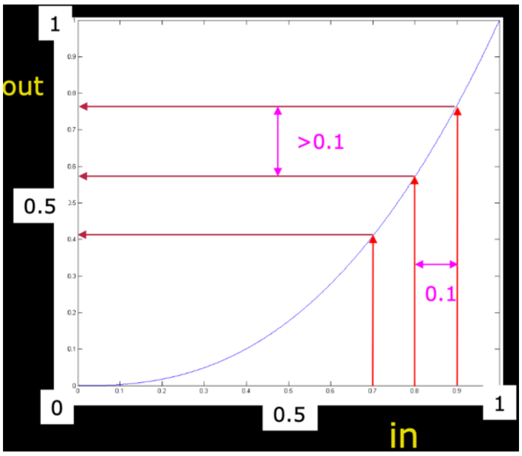


Class III (14/09)

Pixelwise operations and colour images. PCA on images.

1. Gray value mappings

Mapping = To make a correspondence between two sets of values
- Mapping function: out = f(in)



What happens with the values?
- Values with difference 0.1
- Output values "spread out"

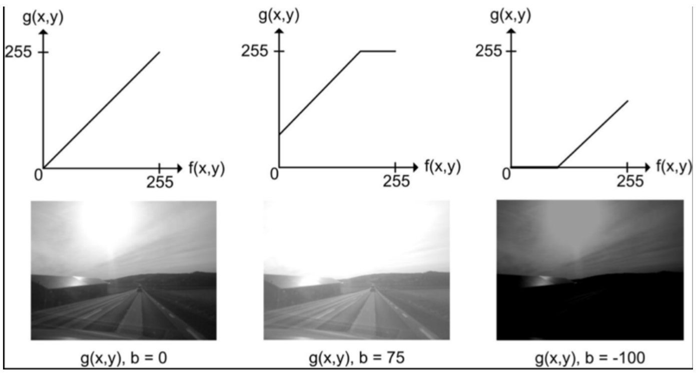
When could it be good to change the gray level values?
- Lack of contrast
- Make the image lighter
- Make the image darker

Point processing

- Grey level enhancement
- The value of the output pixel is only dependent on the value of one input pixel. It processes one pixel at a time independent of all other pixels.
- A global operation: changes all pixels
- Example: Used to correct brightness and contrast

A) <u>Brightness: (=intensity)</u>
To change brightness:
 - To each pixel is added the value *b*
 - *f(x,y)* is the input image
 - *g(x,y)* is the enhanced/output image
 - If *b* > 0: brighter image → It means saturation (removing information)
 - If *b* < 0: less bright image

$$\boxed{g(x,y) = f(x,y) + b}$$



B) <u>Contrast: (=level of details)</u>

To change contrast:

- Each pixel is multiplied by *a*
- *f(x,y)* is the input image
- *g(x,y)* is the enhanced/output image
- If *a* > 0: more contrast
- If *a* < 0: less contrast

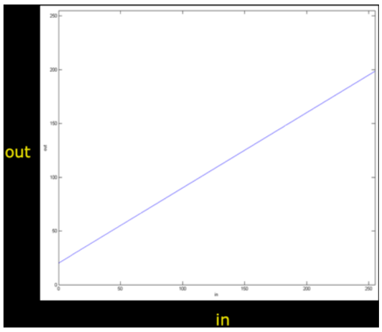
$$g(x, y) = a \cdot f(x, y)$$

C) <u>Combining brightness and contrast →Linear transformation</u>

- A straight line

$$g(x, y) = a \cdot f(x, y) + \mathrm{b}$$

- Example: *a* = 0.7 and *b* = 2
  -More bright (b>0)
  -Less contrast (a<1)



Histogram

- It normally contains the same number of “bins” as the possible pixel values.
- A bin stores the number of pixels with that value.
- The shape of the histogram tells us a lot.
  - Dark vs. Bright image
  - Low vs. High contrast

<u>Histogram stretching:</u>

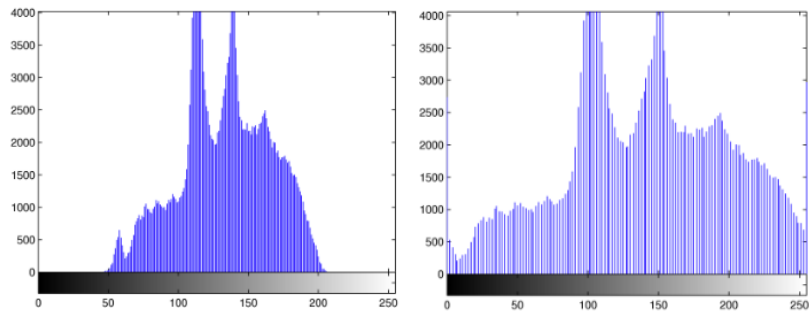
How do we optimise the image using the histogram?

→ Stretch it, so new minimum = 0 and new maximum = 255.

→ We can do it by using **brightness and contrast.**

$$g(x, y) = \frac{v_{max,d} - v_{min,d}}{v_{max} - v_{min}} \left(f(x, y) - v_{min} \right) + v_{min,d}$$

$$v_{min,d} = desired \min value = 0$$
$$v_{max,d} = desired \max value = 255$$
$$v_{min} = current \min value$$
$$v_{max} = current \max value$$



Weaknesses:

- A single pixel value of 0 or 255 ruins the histogram stretching
- Sometimes we just want:
  - To stretch only the high pixel values
  - While “compressing” the low pixel values
  - Non-linear mapping

Deep learning and color/gray scale transformations:

- Deep learning needs training data: input image + ground truth labels or classes
- When you lack data, you can augment your data:
  - Create artificial versions
  - Adding variations
  - Changing gray / color levels in the image
  - Point wise operations

2. Non-linear mappings

Not always nice to work with byte images → better to work with image with values [0,1]

Byte image → Conversion to [0,1] → Non-linear transformation → Back to bytes → Byte image

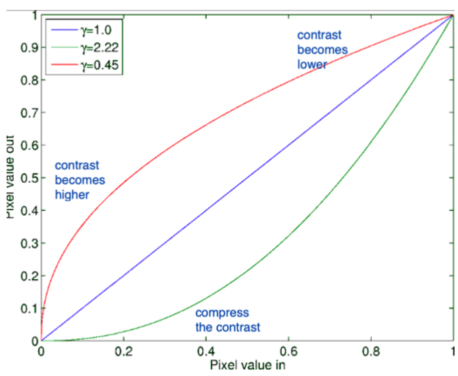
- A byte contains integer values [0, 255]. It cannot store 127.4232.
- A value of type *double* can contain "all numbers"
- Why not use doubles always? One double= 8 bytes in the memory.

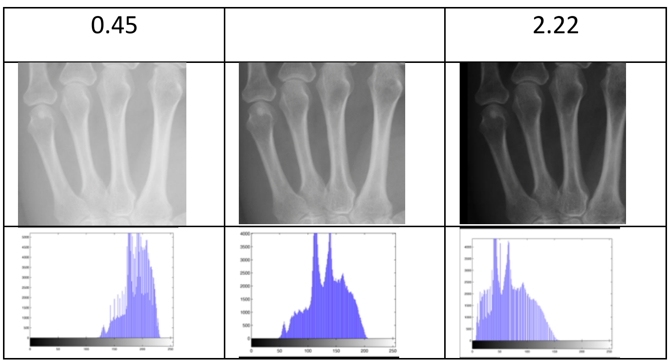
Map pixels to [0,1]	Pixels back to bytes
Conversion to [0,1] $$g(x,y) = \frac{1}{255} f(x,y)$$	Input pixels are [0,1] We want them to be [0,255] Simple linear transformation: $$g(x,y) = 255 \cdot f(x,y)$$ Back to bytes: *uint8*

Gamma mapping

Non-linear mapping/transformation. It helps to increase the contrast (dynamics) in more selected part of the histogram.

$$g(x,y) = f(x,y)^{\gamma}$$

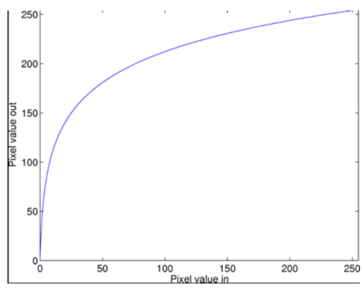




Logarithmic mapping

Non-linear mapping/transformation for images with very bright spots. Thus, low intensity pixel values are enhanced.

$$g(x,y) = c\log(1 + f(x,y)) \quad , c = \frac{255}{\log(1+v_{max})}$$



Thresholding

A threshold T is a value: *(difficult to choose for the entire image)*
The thresholding is chosen based on the histogram.

- Pixels below that value is set to 0 (background)
  - if $f(x, y) \leq T$ then $g(x, y)$ = 0
- Pixels equal or above is set to 1 (object)
  - if $f(x, y) > T$ then $g(x, y)$ = 255

<u>Automatic Thresholding</u> = Otsu's method

- Two classes: background and object
- T divides pixels into object and background
- Compute pixel value variance in each class
- Find T that minimises combined variance

<u>Histogram Shaping:</u>
With a threshold you want a histogram with two peaks: Bimodal
An ideal histogram has well separated peaks.
Obtaining a bi-modal histogram is very important in the image acquisition, but usually not possible.

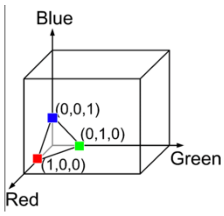
It depends on the different parameters of the camera: light, setup, camera, lens, backlight.

Object colours

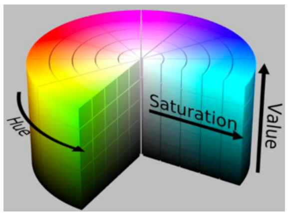
- Subtractive colours
- Additive colours: final colour is made by mixing red, green and blue. (RGB). The values of R, G and B lie between 0 and 255.
  But they can also be normalized.

$$(r, g, b) = \left(\frac{R}{R+G+B}, \frac{G}{R+G+B}, \frac{B}{R+G+B} \right)$$

$$I = \frac{R+G+B}{3}$$



- HSI Color Representation
  - Hue: The dominant wavelength in the perceived light (the pure colour)
  - Saturation: The purity of the colour
  - Intensity: The brightness of the colour (sometimes called the value)



- From RBG and HIS:

$$H = \begin{cases} \cos^{-1}\left(1/2 \cdot \frac{(R-G)+(R-B)}{\sqrt{(R-G)(R-G)+(R-B)(G-B)}}\right), & \text{if } G \geq B; \\ 360° - \cos^{-1}\left(1/2 \cdot \frac{(R-G)+(R-B)}{\sqrt{(R-G)(R-G)+(R-B)(G-B)}}\right), & \text{Otherwise.} \end{cases} \quad (8.8)$$

$$H \in [0, 360[$$

$$S = 1 - 3 \cdot \frac{\min\{R, G, B\}}{R+G+B} \qquad S \in [0, 1] \quad (8.9)$$

$$I = \frac{R+G+B}{3} \qquad I \in [0, 255] \;, \quad (8.10)$$

Melanoma segmentation

Need for an algorithm that can do pixelwise classification (Background/skin vs. Melanoma) using the colours → **Color thresholding**

<u>Color Variation:</u>

- The major variation is in the brightness. This will spread out the values in the RBG space.
- The Hue is rather constant
- HSI Space
    - HUE and saturation rather stable
    - Only variation in intensity /value

Contrast in medical images: Image acquisition

How do we optimise image acquisition when we want to look at:

1.Bones

- X-rays: It goes through soft tissue with little loss and are attenuated in bone.
  The attenuation is the gradual loss in intensity.
- CT scanners use X-rays (good approach)
- A simple threshold can often extract the bones
- Areas with only bone and soft tissue will have a bimodal histogram.

2.Brain structures

→ Magnetic Resonance Imaging (MRI) is used

- Much more difficult to explain since it is based on powerful magnetic fields and radio waves with the need of water molecules.
- The bone is black

3.Cancer

→ CT scan

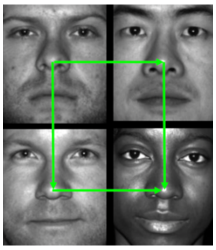
- Liver cancer is difficult to see
- As cancer cells have typically a high metabolism, some substances are easier to see on different scanners.
- Using the cancer metabolism as a *tracer.* The commonly used tracer is: *18F-FDG (18F-fluorodeoxyclucose).* Used in oncological PET.
- Areas with high glucose intake will be brighter (higher intake of radioactive molecules)
- Bimodal histograms in areas with cancer cells

→ Combining images

- CT is good for bone and PET is good for cancer → PET/CT scanner
- Combining two or more separate image using image registration
- The tumour became much more separated from the background. The solution would be:
    - Clever imaging techniques and intelligent image analysis

Principal component analysis on images.

Face data



- Face images: 168 x 192 grayscale
- Aligned: The anatomy is placed “in the same position in all images”
- Same illumination conditions on the image we use

- The main variation in face images: *Variation of appearance*
  - Not the position in the image
  - Not the light conditions
  - Not the direction of the head

<u>Steps:</u>

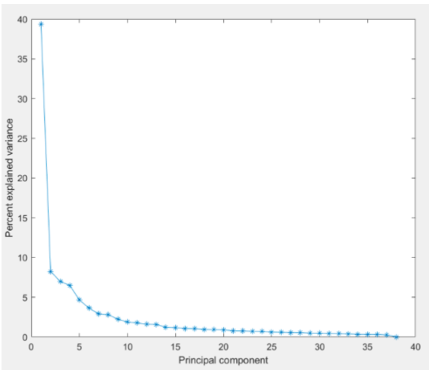
1. Putting images into matrices. The image can be made into a column matrix (stack all image columns into one column)
2. Face images in a matrix form
   - One column is one face: n=38 faces, m=168x192= 32256-pixel values per image

$$X = \begin{pmatrix} p_{1,1} & \cdots & p_{1,n} \\ \vdots & \ddots & \vdots \\ p_{m,1} & \cdots & p_{m,n} \end{pmatrix}$$

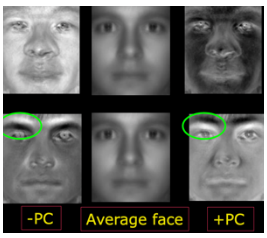
3. The average face. The average of each row → one column. Put it back into image shape.
   When blurry around the eyes, it means that it is not perfectly aligned.
4. Subtracting the mean face from all faces.

$$X' = \begin{pmatrix} p_{1,1} & \cdots & p_{1,n} \\ \vdots & \ddots & \vdots \\ p_{m,1} & \cdots & p_{m,n} \end{pmatrix} - \bar{X}$$

5. Analysing the deviation from the mean face
   We want to do the principal component analysis on the deviations from the average face
   → PCA analysis on X' matrix (32256 x 38)
   - Standard covariance matrix is: 32256 x 32256
   - Turk and Pentland found a trick: compute the PCA on the 38x38 matrix instead od the 322256x32256 matrix
6. PCA on faces. Visualization



- First eigenvector explains 40% of variation
- Second eigenvector explains 8% of variation
- A tool to see major variations: brow lifting



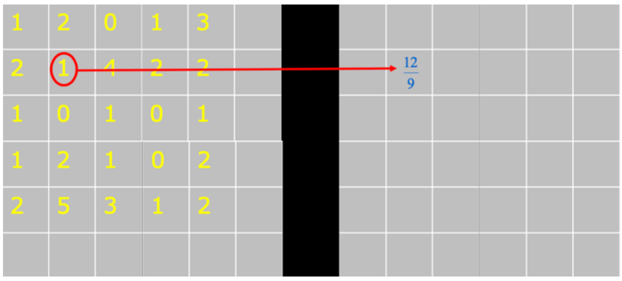
7. Synthesizing faces
   A new face can be created by combing: average face + linear combination of PCs.
   <u>*Ex:*</u> *New face= Average face + 0.05 PC1 – 0.12 PC2*

8. Decomposing faces. Using the average face and the linear combinations of PCs.
   It is found by projecting the face on the principal components.
   The weights can then be used for classification/identification.
   <u>*Ex:*</u> *Decompose face= Average face +* w_1 *PC1 +* w_2 *PC2*

Class IV (21/09)

Neighbourhood Processing

1.Point processing



- The value of the output pixel is only dependent on the value of the one input pixel.
- A global operation that changes all pixels.
- Grey level enhancement: Process one pixel at a time independent of all other pixels. Used to correct brightness and contrast.

Neighbourhood processing

Several pixels in the input have an effect on the output.

<u>Use of filtering:</u>

1. **Noise removal**

- Salt and pepper noise.
    - Pixel values that are very different from their neighbours. Very bright or very dark spots. (0- or 255-pixel values) Scratches in X-rays.
    - Removal: Set all pixels to the average of the neighbours (and the pixel itself). But there is a problem since the noise “pollutes” the good pixels too.
    - Solution: The median filter, as noise has no influence on the median.

<table>
<tr><td colspan="3">The median filter
All pixels are set to the median of its neighbourhood (including the pixel itself).
The median value is the middle number of the values when sorted from low to high.
Noise pixels do not pollute good pixels.</td></tr>
<tr><td>
Scanned X-ray with salt and pepper noise</td><td>
Average filter 3x3</td><td>
Median filter 3x3</td></tr>
</table>

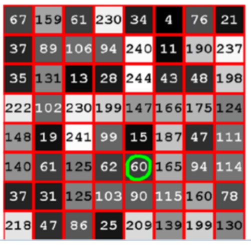
- **Image filtering:** It creates a new filtered image. The output pixel is computed based on a neighbourhood in the input image.
    - 3x3 neighbourhood (filter size 3x3, kernel size 3x3, mask size 3x3)
    - Larger filters often used (size 7x7, with 49 number of elements)

- **Rank filters**. They are based on sorting the pixel values in the neighbouring region.
    - Minimum rank filter. Darker image. Noise problems.

- Maximum rank filter. Lighter image. Noise problems.
- Difference filter. Enhances changes (edges).

<u>Quiz:</u> Rank filters on image.

The image is filtered with a 3x3 median filter (medI). The original image is also filtered with a 5x5 minimum rank filter (minI). The final image is made by subtracting minI from medI. What is the result in the marked pixel?



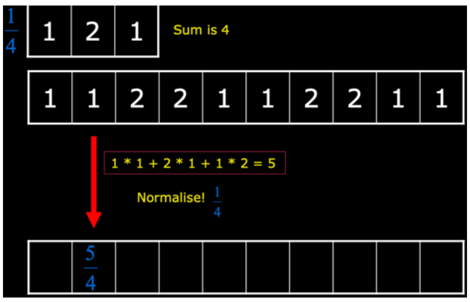
medI 3x3: f(4,5) = 99
minI 3x3: f(4,5) = 15

medI-minI = 84

- **Correlation.** Measure of similarity between measurements.
  A high correlation means that there is a relation between values.

  Image analysis is also about recognition of pattern. We need similarity. Need of something to tell us if there is a high match between our pattern and a part of the image.

<u>1D Correlation:</u> (normalisation is used)

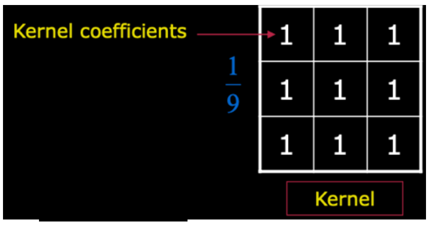


The sum of the kernel elements is used.
Keep the values in the same range as the input image.

Normalisation factor: sum of kernel coefficients

$$\sum_x h(x) = 1 + 2 + 1$$

<u>2D Correlation- Images:</u> The filter is now 2D.

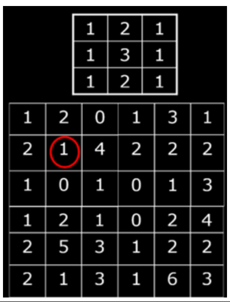


This mask is moved row by row.
Mathematics:

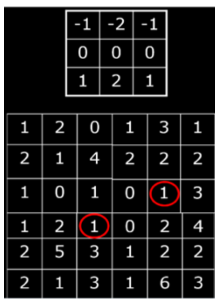
$$g(x,y) = f(x,y)\ o\ h(x,y)$$

$$g(x,y) = \sum_{j=-R}^{R} \sum_{i=-R}^{R} h(i,j) \cdot f(x+i, y+j)$$

<u>Quiz:</u> Correlation on image. No kernel normalization.



f(1,1) = 16



f(2,3)= 10 *f(4,2)=0*

Normalization factor: $\sum_x \sum_y h(x,y)$

<u>Quiz:</u> Template match on image.
A template match is done on the image to the left with the template seen to the right. To find the best match the correlation is computed. What is the correlation in the marked pixel?

227	208	90	97	145	42	58	27
245	62	212	145	120	154	233	245
140	237	149	19	3	67	39	1
35	89	140	14	86	167	211	198
38	50	234	135	41	176	137	208
66	64	73	199	203	191	254	222
214	157	193	238	79	115	20	22
65	121	192	33	135	21	113	102

66	232	37
204	46	35
110	67	222

g (3,3) = 149·66 + 19·232 + 3·37 + 140·204 + 14·46 + 86·35 + 234·110 + 135·67 + 41·222= 90454

2. Smoothing

- **Smoothing filters.**
  (=smoothing kernel, mean filter, low pass filter, blurring)

  - The simplest filter: *Spatial low pass filter.*
    It removes high frequencies.

    1/9

1	1	1
1	1	1
1	1	1

  - *Gaussian filter.* 1/16

1	2	1
2	4	2
1	2	1

Large kernels smooth more by removing high frequency information.
They work good at enhancing big structures.

<u>Quiz:</u> Mean filter on image – missing value.
A 3x3 mean filter is applied to the image. The result in the marked pixel is 86. What is the value of the pixel where the value is missing?

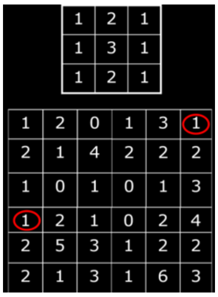
227	208	90	97	145	42	58	27
245	62	212	145	120	154	233	245
140	237	149	19	3	67	39	1
35	89	140	14	86		211	198
38	50	234	135	41	176	137	208
66	64	73	199	203	191	254	222
214	157	193	238	79	115	20	22
65	121	192	33	135	21	113	102

g (4,2) = (145 + 120 + 154 + 19 + 3 + 67 + 14 + 86 + x) 1/9 = 86
608 + x = 86· 9
608 + x = 774
x = 166

Border handling: How to manage the values at the border → By extending the input

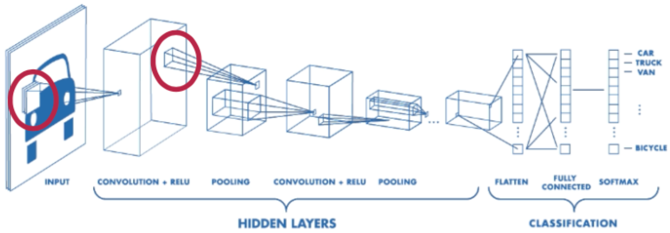
- Zero padding.
  It creates dark border around the image. As there are black borders at the corner, this is not convenient.
- Reflection.
  Better than zero padding.

Quiz: Correlation on image with zero padding



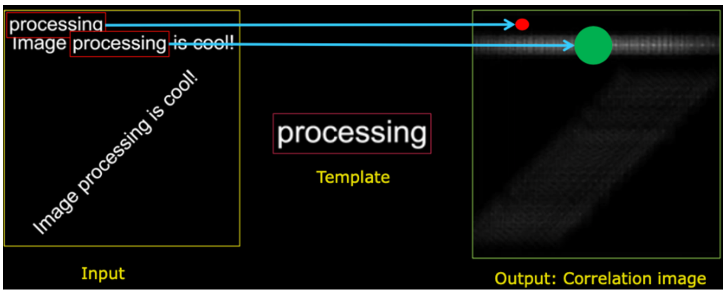
$g(5,0) = 1 \cdot 3 + 3 \cdot 1 + 1 \cdot 2 + 2 \cdot 2 = 12$
$g(0,3) = 2 \cdot 1 + 0 \cdot 1 + 1 \cdot 2 + 5 \cdot 1 + 3 \cdot 1 + 2 \cdot 2 = 16$

- **Deep learning connection → Bank of filters.**
  The part of the network that touches the image consists of a bank of filters, organised in a multi-level hierarchy.
  The weights of the filters are adapted to the problem. (backpropagation)



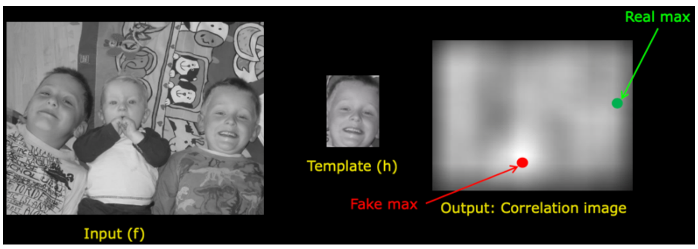
- **Template Matching**
  It locates objects in images. The correlation between the template and the input image is computed for each pixel.

  The pixel with the highest value is found in the output image. (Here is the highest correlation). This corresponds to the found pattern in the input image.



Problematic Correlation:
Correlation matching has a problem with light areas.

$$g(x,y) = \sum_{j=-R}^{R} \sum_{i=-R}^{R} h(i,j) \cdot f(x+i, y+j)$$



→ Solution: **Normalised Cross Correlation.** Another type of normalization.

$$NCC(x,y) = \frac{Correlation}{Length\ of\ image\ patch \cdot Length\ of\ template}$$

<table>
<tr>
<td>Length of template
Vector length: put all pixel values into a vector and compute the length of this vector.

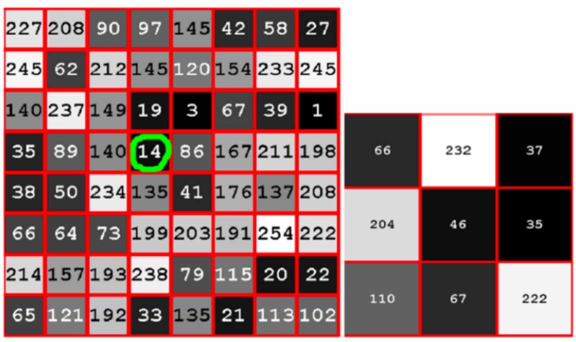
It describes the intensity of the template:
- Bright template has a large length
- Dark template has a small length

$Length\ temp = \sqrt{\sum_{j=-R}^{R} \sum_{i=-R}^{R} h(i,j) \cdot h(i,j)}$</td>
<td>Length of image patch
Vector length based on pixel values in image patch. It describes the intensity of the image patch.</td>
</tr>
<tr>
<td colspan="2">The length of the image patch and the length of template normalise the NCC.
If the image is very bright the NCC will be “pulled down”.

NCC will be between:
▪ 0: No similarity between template and image patch
▪ 1: Template and image patch are identical</td>
</tr>
</table>

<u>Quiz:</u> Normalized cross correlation on image ***(exam)***

A template match using normalized cross correlation is performed. What is the resulting value in the marked pixel?



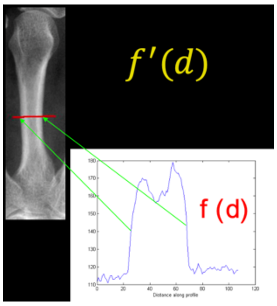
Correlation: g (3,3) =149·66 + 232·19 + 3·37 + 140·204 + 14·46 + 86·35 + 234·110 + 135·67 + 41·222 = 90454

*Length of template: sqrt(sum(sum(T.*T))) = 412.7699*
*Length of image patch: sqrt(sum(sum(Img.*Img))) = 352.7393*

Result: $\frac{correlation}{length\ of\ tempate \cdot length\ of\ image\ path} = \frac{90454}{412.7699 \cdot 352.7393}$ *= 0.6212*

3. Enhance edges

Edge: High change in grey level value. Objects are often separated from background by edges.



The profile as a function of f(d).
What value is high when there is an edge?

- The slope of f
- The slope of the tangent at d

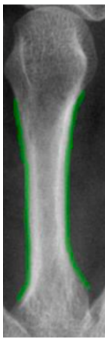
- **Finite Difference**

Definition of slope	Approximation	Simpler approximation h=1
$f'(d) = \lim_{h \to 0} \frac{f(d+h) - f(d)}{h}$	$f'(d) \approx \frac{f(d+h) - f(d)}{h}$	$f'(d) \approx f(d+1) - f(d)$

The discrete approximation of f'(d) can be interpreted as a filter.

	-1	0	1		
f(d-2)	f(d-1)	f(d)	f(d+1)	f(d+2)	f(d+3)

- **Edges in 2D.** Changes in grey level values.



- Image gradient
- Gradient is the 2D derivative of a 2D function f(x,y)
- Equal to the slope of the image
- A steep slope is equal to an edge

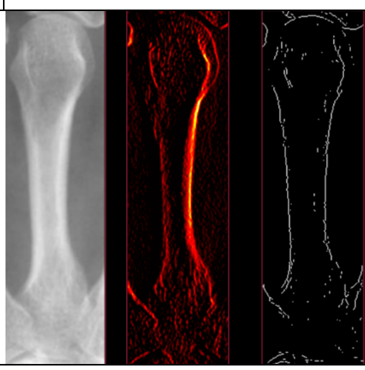
$\nabla f(x,y) = \vec{G}(g_x, g_y)$
Magnitude = $\sqrt{g_x^2\ g_y^2}$

Edge filter kernel: *Prewitt filter.* Typical edge filter.

-1	0	1
-1	0	1
-1	0	1

Output image has high values where there are edges.

Edge detection:

1.Edge filter. Prewitt filter as an example	2. Thresholding. Separate edges from non-edges.	3.Output as a binary image. Edges are white.
		

Morphology

The science of form, shape and structure

- Point wise operations
- Filtering
- Thresholding. Gives us objects that are separated by the background.
- **Morphology.** Manipulate and enhance binary objects.
    - To remove noise
    - To isolate objects
    - To customize to specific shapes

How does it work?

Grayscale image → Pre-processing (inversion) → Threshold (binary image) → Morphology

Filtering	Morphology
- Gray level images - Kernel - Moves it over the input image - Creates a new output image	- Binary images - Structuring elements (SE) - Moves the SE over the input image - Creates a new binary output image

1D Morphology

Input image:

1	0	**0**	**0**	**1**	1	1	0	1	1

Structuring Element (SE):

1	1	1

Output Image:

Hit Operation → **DILATION**

(To make wider or larger)

- If just one 1 in the SE match with the input
    - Output 1
- Else
    - Output 0

	1	0	1	1	1	1	1	1	

(The object gets bigger, and holes are filled)

$$g(x) = f(x) \oplus SE$$

Fit Operation → **EROSION**

(To wear down, to make smaller)

- If all 1 in the SE match with the input
    - Output 1
- Else
    - Output 0

	0	0	0	0	1	0	0	0	

(The object gets smaller)

$$g(x) = f(x) \ominus SE$$

Structuring Element (Kernel)

- SE can have varying sizes. They can be customized to a specific problem.

Disk

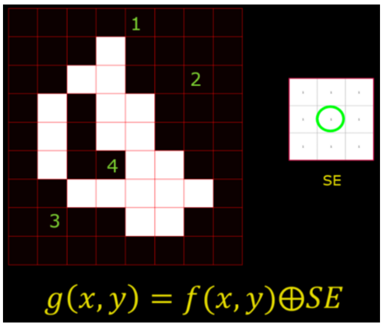
0	1	0
1	1	1
0	1	0

Box

1	1	1
1	1	1
1	1	1

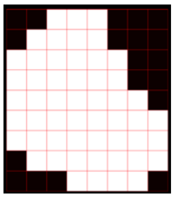
- Usually, element values are 0 or 1, but other values are possible.
- SEs have an origin, not always at the centre.
- It doesn't matter if we have empty spots in the structural elements.

Quiz: *Dilation on image*

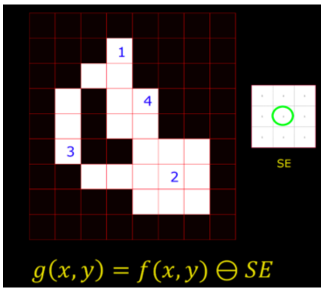


1: 1
2: 0
3: 1
4: 1

the effect of the SE with dilation

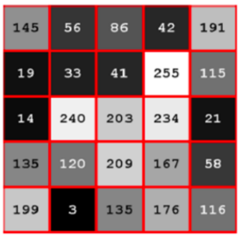


Quiz: *Erosion on image*



1: 0
2: 1
3: 0
4: 0

Quiz: *Threshold and Dilation*
A threshold of 200 is applied to the image and the result is a binary image. Now a dilation is performed with the structuring element below. How many foreground pixels are there in the resulting image?



0	0	0	1	0
0	1	1	1	1
1	1	1	1	1
0	1	1	1	0
0	0	1	0	0

Foreground pixels: 14

Counting Coins

→ Threshold and Erosion separates them

Compound operations:

Made of two or more separate parts or elements
Combing Erosion and Dilation into more advanced operations:

1. Finding the outline:

a. Dilate input image (objects gets bigger)
b. Subtract input image from dilated image and the outline remains.

$$g(x,y) = (f(x,y) \oplus SE\,) - f(x,y)$$

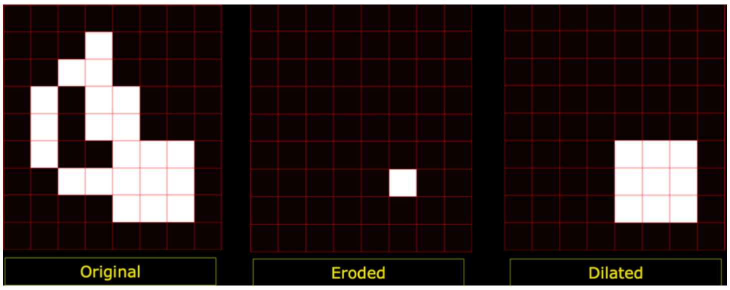
2. Opening:

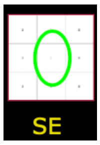
Isolate objects and remove small objects but keeping the original size and shape. Better than erosion.

→ Opening = Erosion + Dilation

- Use the same structing element
- Like erosion but less destructive.

$$g(x,y) = f(x,y) \circ SE = (f(x,y) \ominus SE) \oplus SE$$

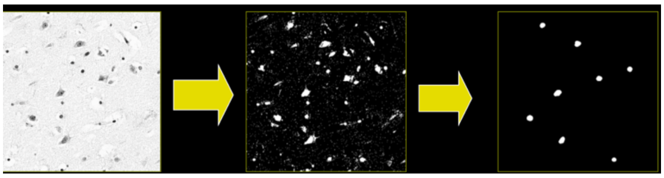




- Opening is idempotent: Repeated operations has no further effects:

$$f(x,y) \circ SE = (f(x,y) \circ SE) \circ SE$$

- The size of structing element should fit into the smallest object to keep it
- Opening example:



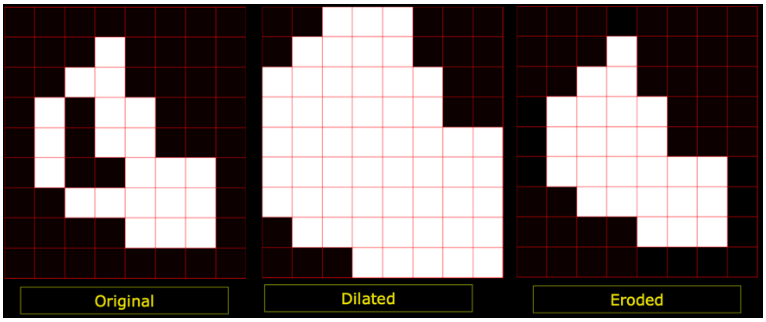
3. Closing:

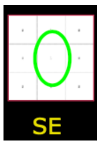
Fill holes but keeping original size and shape. Better than Dilation.

→ Closing = Dilation + Erosion

- Use the same structuring element
- Like Dilation but less destructive.

$$g(x,y) = f(x,y) \cdot SE = (f(x,y) \oplus SE) \ominus SE$$

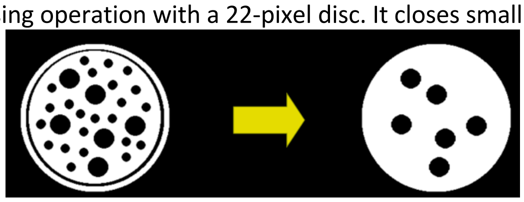




- Closing is idempotent: Repeated operations has no further effects:

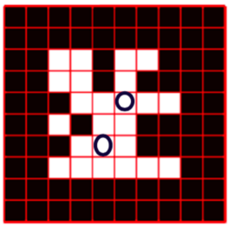
$$f(x,y) \cdot SE = (f(x,y) \cdot SE) \cdot SE$$

- Closing Example: Closing operation with a 22-pixel disc. It closes small holes.



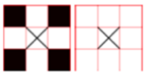
Quiz: Compound operations on image

The compound morphological operation seen below is applied to the image. How many foreground pixels are there in the resulting image?



$(I \ominus SE1) \oplus SE2$

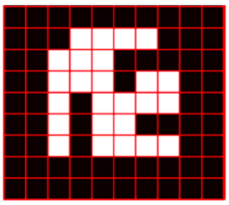
Foreground pixels: 16



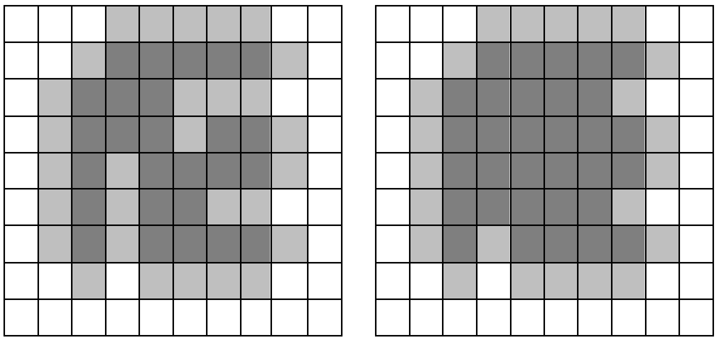
SE1 SE2

Quiz: Closing on image

Morphological closing is applied to the image using the structuring element below. How many foreground pixels are there in the resulting image?



$f(x,y) \oplus SE) \ominus SE$



Foreground pixels: 32

Class V (28/09)

Pixel Classification and advanced segmentation

Pixel Classification

Classify each pixel, independently of neighbours. Also called *labelling (=put a label on each pixel)*
By looking at the pixel value and assigning them a label, which are already defined.

Pixel value (the measurement)	$v \in \mathcal{R}$	$v \in [0{,}255]$
K classes	$C = c_1, \dots, c_k$	$C =$ *{background, soft-tissue, trabeculae, bone}*
Classification rule	$c: \mathcal{R} \rightarrow \{c_1, \dots, c_k\}$	$c: v \rightarrow$ *{background, soft-tissue, trabeculae, bone}*

Pixel classification rule

Manual inspection:

- By looking at some few pixels.

Minimum distance classification: A pixel classification rule.

- The possible pixel values are divided into ranges.
- Measure the "distance" to the other classes and select the closest class.

Ex: $c: v \rightarrow \{backgroumd, soft - tissue, trabelucae, bone\}$
For all pixel in the image do:

$$c(v) = \begin{cases} \text{background, if } v \leq (4+67)/2 \\ \text{soft} - \text{tissue, if } \frac{(4+67)}{2} < v \leq \frac{67+95}{2} \\ \text{trabeculae, if } \frac{67+95}{2} < v \leq \frac{95+150}{2} \\ \text{bone, if } v > \frac{95+150}{2} \end{cases}$$

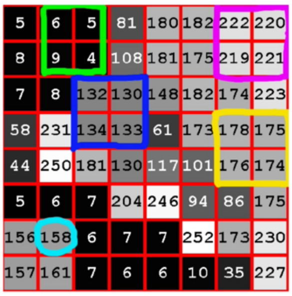
The only problem is to establish the initial pixel value for a group. Guessing range values is not a good idea.
This can be solved it by training data, starting by selecting representative regions from an image.
→ *Annotation:* to mark points, regions, lines or other significant structures.

- An "expert" is asked how many different tissue types that are possible. Then the expert is asked to mark representative regions of the selected tissue types.
- Training is only done once. Optimally, the training can be used on many pictures that contains the same tissue type.
- MATLAB tool: *roipoly*

<u>Classifier training – region selection</u>

1. Initial Analysis: Histograms. For each class, we have a distribution of pixels. Having a good statistical representation.
2. Simple pixel statistics. Calculate the mean and the standard deviation of each class. (Normal distribution). Then, we compute the minimum distance classification.
   Problem: The pixel value ranges are not always in good correspondence with the histograms.

<u>Quiz:</u> Minimum distance classification
To make a pixel classification an expert has selected representative regions in the image. They contain background (green), soft tissue (blue), fat (yellow), and bone (purple). The goal is to classify the marked pixel using minimum distance classifier.



Green: (6+5+4+9)/4 = 6
Blue: (132+130+134+133)/4 = 132,25
Yellow: (178 + 175+176+174)/ 4 = 175,75
Purple: (222 + 220 +219 + 221) / 4 = 220

158 is closed to 175,75 (yellow) = fat

Parametric classification

- I describe the histogram using a few parameters.
- Assume a model describing the signal values.
- Fit a Gaussian to the training pixels for all classes.

The pixel value ranges depend on:

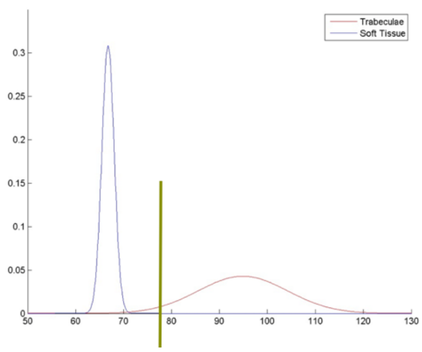
- The mean μ
- The standard deviation σ
- $N(\mu, \sigma)$

Then:

- Training pixel values (belonging to one class): $v_1, v_2, \dots, v_n$
- Estimated mean: $\hat{u} = \frac{1}{n}\sum_{i=1}^{n} v_i$
- Estimated standard deviation: $\hat{\sigma}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(v_i - \hat{u})$
- The "signal model" is a Gaussian distribution

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}}\exp(-\frac{(x-\mu)^2}{2\sigma^2}$$

<u>Quiz:</u> Two tissue types – minimum distance v = 78
To make a pixel classification an expert has selected representative regions in the image. They contain background (green), soft tissue (blue), fat (yellow), and bone (purple). The goal is to classify the marked pixel using minimum distance classifier.

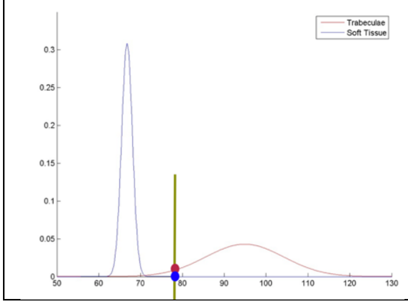


Trabeculae has much higher variation in the pixel value.

1- Find a threshold
$\mu_A(soft\ tissue) = 68,\ \mu_B(trabeculae) = 95$
T= (95+68)/2 = 81.5
2- Classification
v<81, 5 → Soft tissue

Is that fair?
Soft-tissue Gaussian says, "Extremely low probability that this pixel is soft-tissue".

Quiz: Two tissue types – parametric classification. Which tissue class?



The A (trabeculae) distribution is higher than the B (soft tissue) at v = 78.

Solution: Trabeculae

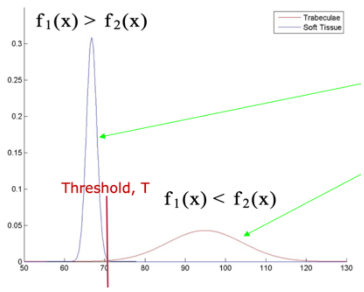
!!! Then: *Where should we set the limit?* Where the two Gaussian cross.

Parametric classification

1. Select training pixels for each class
2. Fit Gaussians $N\ (\mu_i, \sigma_i)$ to each class
3. Use Gaussians to determine pixel value ranges

Parametric classifier – ranges

- Compute where the distributions cross.



$$f_1(x) = \frac{1}{\sigma_1\sqrt{2\pi}}\exp\left(-\frac{(x-\mu_1)^2}{2{\sigma_1}^2}\right)$$

$$f_2(x) = \frac{1}{\sigma_2\sqrt{2\pi}}\exp(-\frac{(x-\mu_2)^2}{2{\sigma_2}^2})$$

Create a lookup table:

- Run though all 256 possible pixel values
- Check which Gaussian is the highest
- Store the [value, class] in the table

Alternatively – analytic solution

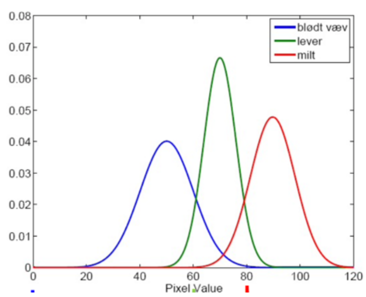
The two Gaussians:

$$\frac{1}{\sigma_1\sqrt{2\pi}}\exp\left(-\frac{(v-\mu_1)^2}{2\sigma_1^2}\right) = \frac{1}{\sigma_2\sqrt{2\pi}}\exp\left(-\frac{(v-\mu_2)^2}{2\sigma_2^2}\right)$$

Intercept at:

$$v = \frac{{\sigma_1}^2\mu_2 - {\sigma_2}^2\mu_1 \pm \sqrt{-{\sigma_1}^2{\sigma_2}^2\left(2\,\mu_2\,\mu_1 - {\mu_2}^2 - 2\,{\sigma_2}^2\ln\left(\frac{\sigma_2}{\sigma_1}\right) - {\mu_1}^2 + 2\,{\sigma_1}^2\ln\left(\frac{\sigma_2}{\sigma_1}\right)\right)}}{-{\sigma_2}^2 + {\sigma_1}^2}$$

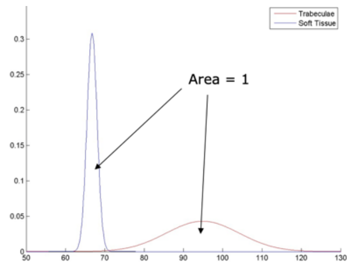
Quiz: Class ranges.



An expert has chosen representative regions in an image that contains soft tissue, liver and spleen. The image pixel minimum and maximum values are 0 and 255. To make parametric classification, the histograms are parameterized using Gaussian distributions. What are the class ranges?

Ranges: [0, 60], [60,80],[80,255]

Bayesian Classification



Bayes' Theorem
$P(A\|B) = \dfrac{P(B\|A)\,P(A)}{P(B)}$

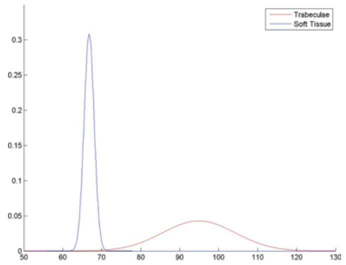
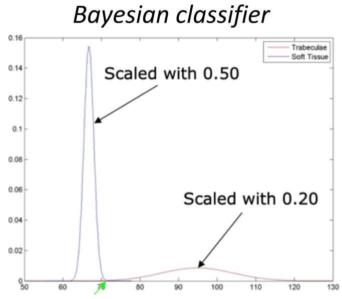
Pure parametric classifier assumes equal amount of different tissue types.

__How do we handle that?__

An expert tells us that a CT scan of a head contains
- 20% Trabecular bone
- 50% Soft-tissue

So, when picking a random pixel in the image:
- 20% chance that it is trabecular bone
- 50% chance that it is soft tissue

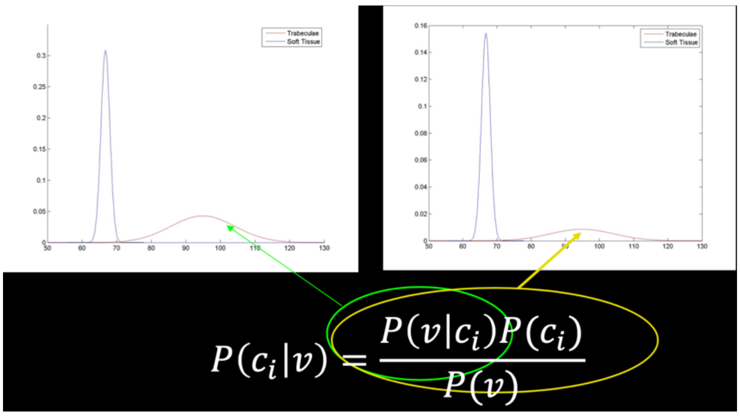
Histogram Scaling

Parametric classifier	*Bayesian classifier*
	
	Little change in class border

<u>Formal definition:</u> Given a pixel value v, what is the probability that the pixel belongs to class c_i.
Ex: If the pixel value is 78, what is the probability that the pixel is bone.

$$P(c_i|v) = \frac{P(v|c_i)P(c_i)}{P(v)}\ , \qquad P(v) \rightarrow cnt.$$

- A priori probability: $P(c_i)$ *(what is known from before)*
  Ex: From the general biology, it is known that 20% of a brain CT scan is trabecular bone. P(trabecular)=0.2
- Class conditional probability: $P(v|c_i)$. Given a class, what is the probability of a pixel with v.
  Ex: If we consider class = soft tissue. What is the probability that the pixel value is 78?



<u>Bayesian Classification - How?</u>

- Select training pixels for each class
- Fit Gaussians to each class
- Ask an expert for the prior probabilities (how much there normally is in total of each type)
- For each pixel in the image
  - Compute $P(v|c_i)$ for each class (= the a posterior probability)
  - Select the class with the highest $P(v|c_i)$.

<u>Bayesian Classification - When?</u>

- Parametric classifier: good when there is approximately the same amount of all type of tissues
- Bayesian classification: good if there are very little or very much of some types
- A more general formulation for segmentation
- When going to higher dimensional features space

High Dimensional feature space

Combine different features input to improve segmentation for:

- Different image modalities (e.g.: CT vs. MRI)
- Subject groups (health vs. disease)
- Different angles of objects
- Segmentation with more feature inputs
- To train our "model" with class examples.
  → Draw tissue specific regions for each class
  What is the optimal **Decision-boundary?**

Decision Boundary:

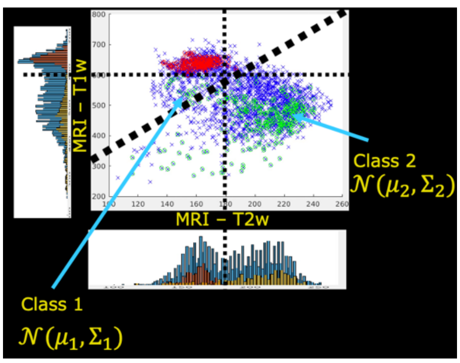
- 2D feature space
- Model assumption: Type of distribution?
- Intensity histograms per class looks Gaussian like?
  - We assume Gaussian distribution: $\mathbb{N}\,(\mu, \sigma)$
- Optimal decision boundary using Bayes Theorem:
  - Likelihood ration for belonging to C2:

We wish to find T using Bayes:

$$\frac{\mathrm{P}\,(C2 \mid x\,)}{\mathrm{P}\,(C1 \mid x\,)} > T \qquad\qquad \ln(P(C2 \mid x\,)) - \ln(P(C1 \mid x\,)) \; > \; T$$

The posterior probability: $P(C_i\,|x) = \; P(x|\mu_i\,, \sum_i)P_{ci}$
The class specific Gaussian model: $P(x|\mu_i\,, \sum_i) = \; K_i \exp\,((x - \mu_i\,)^T \sum_i^{-1}(x - \mu_i\,))$

-Data points: $x_i \; = [x1, x2]^T$
-Training set: $t_{x \in c1} \; = 0$ and $t_{x \in c2} \; = 1$
-The class mean of training: $\mu_i \; = \frac{1}{N} \sum_{n \in ci} x_n$
-The covariance matrix of training: $\sum_i = \; (x - \mu_i\,)^T(x - \mu_i\,)$

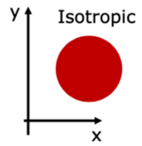
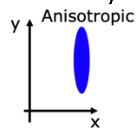
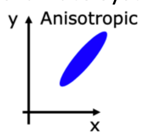


Gaussian in 2D: The covariance matrix

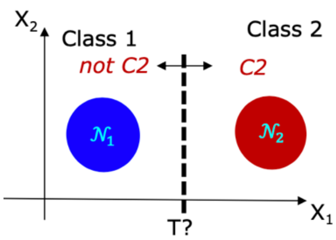
Refresh:

The covariance matrix expresses the orientation of anisotropic variance in relation to coordinate system.

$$\Sigma_i = (x-\mu_i)^T(x-\mu_i)$$

Isotropic- rotational invariant	Anisotropic- aligned with coordinate system	Anisotropic- not aligned with coordinate system
		
$\Sigma = \begin{bmatrix} \sigma_{xx} & 0 \\ 0 & \sigma_{yy} \end{bmatrix}, \sigma_{xx} = \sigma_{yy}$	$\Sigma = \begin{bmatrix} \sigma_{xx} & 0 \\ 0 & \sigma_{yy} \end{bmatrix}, \sigma_{xx} \neq \sigma_{yy}$	$\Sigma = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{xy} & \sigma_{yy} \end{bmatrix}$

Back to the Decision boundary:



- Classifier: If x belongs to C2: $\frac{P(C2 \mid x)}{P(C1 \mid x)} > T$
- Taking the logarithm: $\ln(P(C2 \mid x)) - \ln(P(C1 \mid x)) > T$
- Where the log posterior probability:
- $\ln(P(C2 \mid x)) = \frac{1}{2}(x-u_i)^T\Sigma_i^{-1}(x-u_i)$ +ln (K_i) + ln (P_i)

P_i is the prior probability for class C_i.

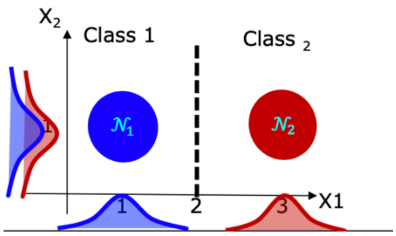
- Inserting and assuming homoscedasticity ($\Sigma_1 = \Sigma_2 = \Sigma_0$) → We have a **Linear discriminant Analysis (LDA)** classifier model (reorganize the expression)
$$ln\frac{P2}{P1} + \frac{1}{2}(u_2+u_1)^T\Sigma_0^{-1}(u_2-u_1) - x^T\Sigma_0^{-1}(u_2-u_1) > T$$
- We train the classifier to find T with examples obtained from the two distributions N1 and N2.

LDA - Optimal Decision Boundary:

We want both probabilities to be equal.

Quiz: LDA-Optimal Decision boundary

- *Define T for x belonging to C2:* $\frac{P(C2 \mid x)}{P(C1 \mid x)} > T$
- *Using Linear Discriminant Analysis (LDA):*
$$ln\frac{P2}{P1} + \frac{1}{2}(u_2+u_1)^T\Sigma_0^{-1}(u_2-u_1) - x^T\Sigma_0^{-1}(u_2-u_1) > T$$



Prior probabilities: P1=P2=0.5

$$\Sigma_1 = \Sigma_2 = \Sigma_0 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

At which value x is the optimal decision boundary T?

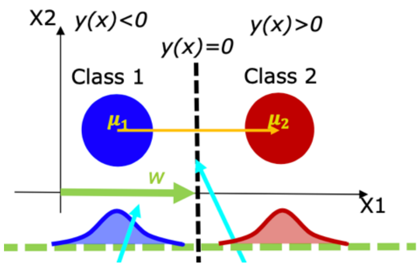
Solution: We see that x for optimal T is a threshold only along X1:

$ln\frac{P2}{P1} + \frac{1}{2}(u_2+u_1)\frac{(u_2-u_1)}{\sigma_0} = x1\frac{(u_2-u_1)}{\sigma_0}$ → $ln\frac{0.5}{0.5} + \frac{1}{2}(3+1)\frac{(3-1)}{2} = x\frac{(3-1)}{2}$ → x1=2, x2= all values

Hyper plan and projections in feature space

- We wish to predict the C2: $\frac{P(C2 \mid x)}{P(C1 \mid x)} > T$
- The LDA function for C2:

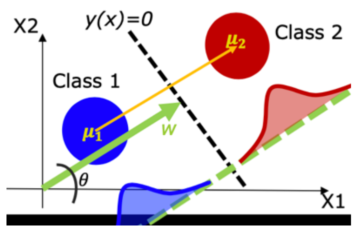
$$\underbrace{\underbrace{\ln\frac{P1}{P2}+\frac{1}{2}(\mu_2+\mu_1)^T}_{c}\underbrace{\Sigma_0^{-1}(\mu_2-\mu_1)}_{w}}_{w_0} - x^T\underbrace{\Sigma_0^{-1}(\mu_2-\mu_1)}_{w} > T$$

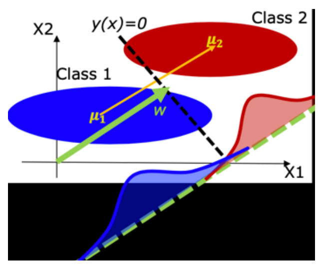


w projects in the class mean direction (i.e., weight vector) *Hyper plan*

w is normal to the hyper plan yi(x) = 0
$x^T w$ is a dot product i.e., x and c are projected onto w ($a^T b = \|a\|\,\|b\|\cos(\theta)$)

- The linear discriminant function:
  $y_{c\epsilon 2}(x) = x^T w + w_0$, where negative w_0 is the threshold
- x is assigned to C2 if $y_{c\epsilon 2}(x) > 0$
- yi(x) = 0 defines a hyper plan for the decision boundary



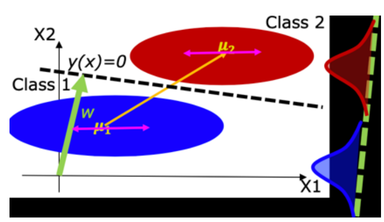


If covariance is anisotropic is not identity matrix.
- Not optimal placement of hyper plan based on mean separation
- Not optimal segmentation results
- Hyper plan does not ensure optimal separation

To improve separation: We need to adjust the weigh vector W.
→ **Fisher's Linear Discriminant**
- Uses: between-class (means) covariance: $S_B = (u_2 - u_1)^T(u_2 - u_1)$
- and: optimize (total) within class covariance $S_w = \Sigma_1 + \Sigma_2$

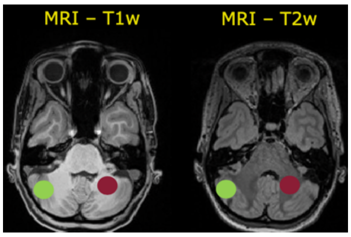
- Find projection w using cost function: $J(w) = \frac{w^T S_B w}{w^T S_w w}$
  - And differentiate: $\frac{\partial J(w)}{\partial w} = 0$
  - Which gives (simple solution): $w \propto S_w^{-1}(\mu_2 - \mu_1)$

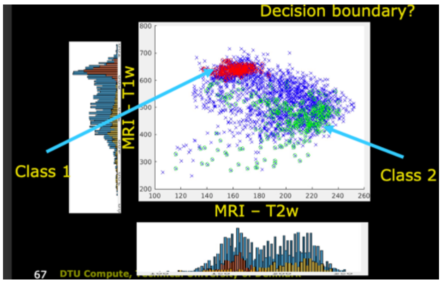
- Optimal class separation:
  → The weight vector w now account for both for class means and variances

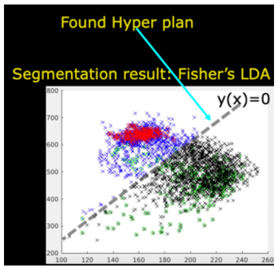


Segmentation of brain using LDA

- Fisher's Linear discriminant
- Use MATLAB function ('LDA-m')





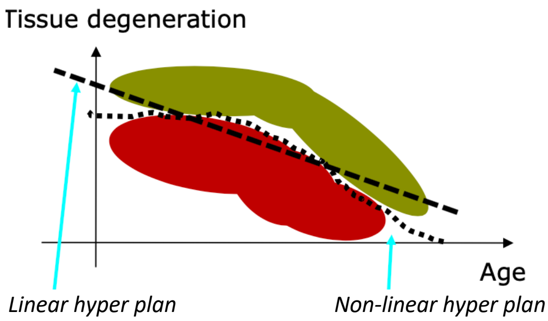


Limitations of LDA

- LDA only uses Linear Hyper plan:
- What is we need non-linear hyper plans?

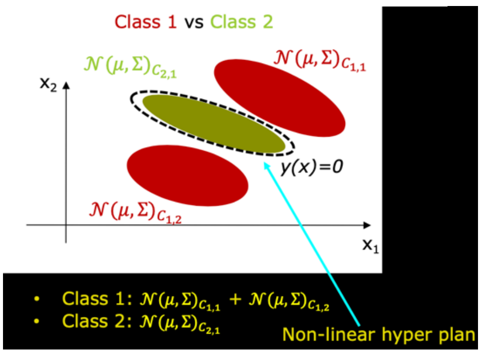
<u>Ex:</u> Classes of health controls vs. Patient

Patient vs Healthy controls



- <u>*Solution:*</u> One class can be separated: A non-linear problem

Non-linear Hyper Plans



Non-linear classifier (Machine learning):

<u>Example:</u>

- Gaussian Mixture Model

Each class is modelled using a number of Gauss distributions.

- Again, use Bayes theorem also for Gaussian Mixture Model.
- Optimisation:
  - We derive $\frac{\partial J(w)}{\partial w} = 0$ for a Gaussian mixture model
  - Iterate optimisation algorithm is used to find w.

→ Segmentation- Non-linear Hyper plans: Convolutional neural network and classification.

Class VI (05/10)

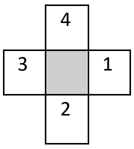
BLOB ANALYSIS AND FEATURE BASED CLASSIFICATION

What is a BLOB?

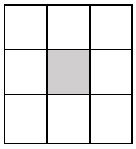
Object Recognition → Recognize objects in images
BLOB = Binary Large Object, group of connected pixels.
BLOB analysis:

- Connected component analysis
- Object labelling

▪ We want: For each object in the image, a list with its pixels.
▪ How to get: Connected component analysis
▪ Method: Connectivity. Who are my neighbours?
  - 4-connected
  - 8-connected

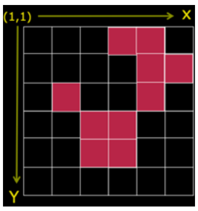


4-connected



8-connected

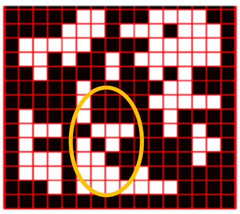
Connected component analysis



- Binary image
- Seed point: where do we start?
- Grassfire concept:
  - Delete (burn) the pixels we visit
  - Visit all connected (4 or 8) neighbours.

→ Result: An image where each BLOB (component) is labelled. Each blob now has a unique ID number.

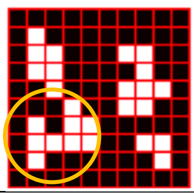
__Quiz:__ BLOBs with 4- and 8- connectivity
A BLOB analysis is perform using both 4- and 8- connectivity. How many BLOBs are found using the two different connectivities?



4-connectivity: 8 BLOBs (groups of pixels)

8-connectivity: 6 BLOBs (groups of pixels). We have less BLOBs as the two subgroups marks (in yellow) are grouped together.

__Quiz:__ BLOB extraction and binary run length coding
*A BLOB analysis using **4-connectivity** is performed. It is a 0-based (x,y) coordinate system with origin in the upper left corner. The largest BLOB is kept, and the resulting image is coded using __binary run-length code.__ What is the code?*



[row; (columns)]
The largest BLOB is the one marked in yellow.
[5;(3;3)]; [6;(1;1)]; 6;(3;4)]; 7;(1;4)]; 8;(1;1)];

Feature Selection

Feature → A prominent or distinctive aspect, quality or characteristic.

- Feature vector: vector with all the features for one object

Feature extractions

We can compute features for each BLOB that can be used to identify: size, shape, position.

From image operations to mathematical operations:

- Input: a list of pixel positions
- Output: Feature vector

First step: Remove invalid BLOBS.

- Too small or big- using morphological operations *(for example)*
- Border BLOBs

BLOB Features

- Area = number of pixels in the BLOB. It can be used to remove noise (small BLOBS)
- Bounding Box = minimum rectangle that contains the BLOB.

  Height: $y_{max} - y_{min}$ Width: $X_{max} - X_{min}$

  Bounding box ratio: $\frac{y_{max}-y_{min}}{X_{max}-X_{min}} = \frac{height}{width}$

  Bounding box area: $(y_{max} - y_{min}) \cdot (x_{max} - x_{min})$

  (It tells if the BLOB is elongated)

  - Translation: Invariant
  - Rotation: Variant. For a perfect cercle, the rotation is invariant.
  - Compactness: $\frac{\text{BLOB Area}}{(y_{max}-y_{\min}) \cdot (X_{max}-X_{\min})}$

- Centre of mass (x_c, y_c)

  $x_c = \frac{1}{N}\sum_{i=1}^{N} x_i$ $y_c = \frac{1}{N}\sum_{i=1}^{N} y_i$

- Perimeter: $\sum((f(x,y) \oplus SE) - f(x,y))$

  (In practice (In MATLAB) it is computed differently and more accurately)

- Circularity: (*How much does it look like a circle?)*

  Circle: - Area: A=πr^2

  - Perimeter: P=$2\pi r$

  New object assumed to be a circle:

  - Measured perimeter P_m
  - Measured area A_m

  Estimated perimeter from (measured) area

  - Estimated perimeter $P_e = 2\sqrt{\pi A_m}$

  Compare the perimeters: $P_m \ vs. P_e$

  Circularity = $\frac{P_m}{P_e} = \frac{P_m}{2\sqrt{\pi A_m}}$ (*normally bigger* ≥ 1)

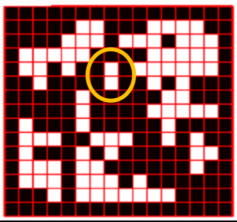
Quiz: Circularity math

$P_m > P_e$

$P_m = P_e$ *(just with a perfect circle)*

- Circularity Inverse: $\frac{P_e}{P_m} = \frac{2\sqrt{\pi A_m}}{P_m}$ (*normally* ≤ 1)

<u>Quiz:</u> BLOB Centre of Mass
The smallest BLOB is found using 4-connectivity. What is the centre of mass of this BLOB? The image has origin (0,0) and uses a (x,y) coordinate system.



$$x_c = \frac{1}{N}\sum_{i=1}^{N} x_i = \frac{1}{2}\,(7+7) = 7$$
$$y_c = \frac{1}{N}\sum_{i=1}^{N} y_i = \frac{1}{2}\,(4+5) = 4.5$$

Centre of mass: (7, 4.5)

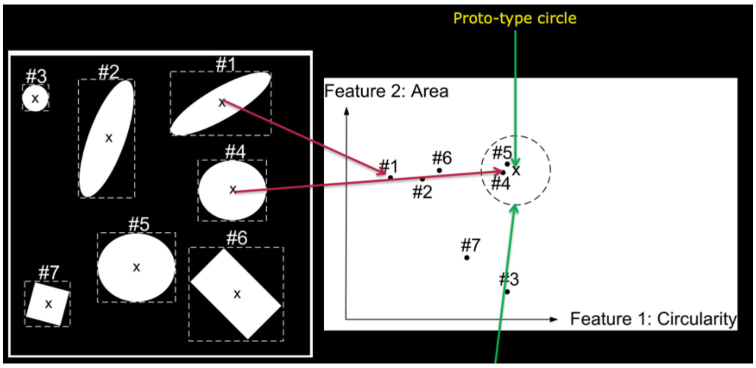
BLOB Classification

1. Put a BLOB into a **class**. Classes are normally pre-defined (car, bus, motorcycle, scooter…)
2. Then: **Object recognition.**

<u>Ex:</u> Circle classification (circle vs. not-circle). Let's make a model of a proto-type circle.

-Proto-type circle: Circularity=1, Area=6700

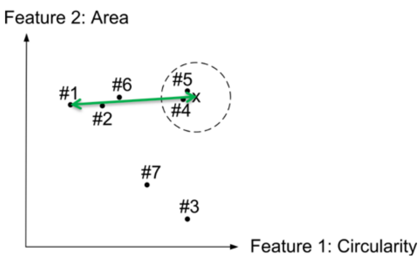
<u>Feature Space:</u>



Objects in here are classified as circles

- Some slack is added to allow non-perfect circles: Circularity: 1 +/- 0.15

To decide if an object is inside the circle:
→ Feature space distance
→ Euclidean distance in feature space

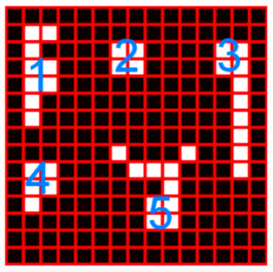


Blob 1: circularity= 0.31, Area= 6561

D= $\sqrt{(0.31-1)^2 + (6561-6700)^2}$

Dominates all, normalisation needed!

<u>Quiz:</u> BLOB Classification
A BLOB analysis using 8-connectivity has been performed on the image and the five found BLOBs have been found. The BLOB features area and compactness have been computed for the five BLOBs. A <u>reference BLOB has an area of 10 pixels and compactness of 0.5</u>.
The Euclidean distance in features space has been computed between the five BLOBs and the reference BLOB. Which of the five BLOBs has the minimum distance?



Reference BLOB:
- *Area: 10 pixels*
- *Compactness: 0.5* = $\frac{\text{BLOB Area}}{(y_{max}-y_{\min})\cdot(x_{max}-x_{\min})}$

Comp1=$\frac{9}{(2-1)\cdot(6-1)} = \frac{9}{6} = 1.5$ D1= $\sqrt{(1.5-0.5)^2+(9-10)^2}$ = 1.4142

Comp2=$\frac{4}{(7-6)\cdot(3-2)} = \frac{4}{1} = 4$ D2= $\sqrt{(4-0.5)^2+(4-10)^2}$ = 6.946

Comp3=$\frac{10}{(13-12)\cdot(9-2)} = \frac{10}{7} = 1.43$ D3= $\sqrt{(1.43-0.5)^2+(10-10)^2}$ = 0.93

Comp4=$\frac{5}{(11-9)\cdot(2-1)} = \frac{5}{2} = 2.5$ D4= $\sqrt{(2.5-0.5)^2+(5-10)^2}$ = 5.38

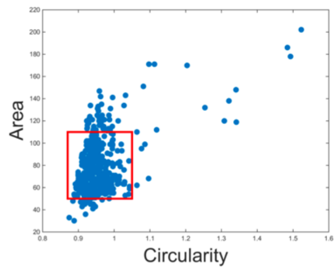
Comp5=$\frac{10}{(10-6)\cdot(12-8)} = \frac{10}{16} = 0.625$ D5= $\sqrt{(0.625-0.5)^2+(10-10)^2}$ = **0.125**

*Minimum distance with **BLOB 5.***

Cell Classification

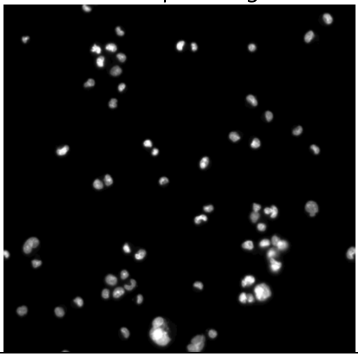
Nuclei classification

- DAPI image
- Two classes:
  - Single nuclei
  - Noise: Multiple nuclei together, debris, other noise



<u>Training and annotation:</u>
- Selection of true single nuclei marked
- Thresholding
- BLOB Analysis: Circularity, Area

<u>Using the classifier:</u>
- Threshold input image
- Morphological opening (SE 5x5)
- Morphological closing (SE 5x5)
- BLOBs found using 8-neighbours
- Border BLOBs removed
- Border features computed: Area + circularity
- BLOBs with features inside the acceptance range are *single nuclei*

DAPI input image	Found single nuclei
	

<u>Creating ground truth:</u> expert annotations

- Expert opinion on true single nuclei

<u>Four cases:</u>

1. True Positive (TP): A nuclei is classified as a nuclei
2. True Negative (TN): A noise object is classified as noise object
3. False Positive (FP): A noise object is classified as a nuclei
4. False Negative (FN): A nuclei is classified as a noise object

- Confusion matrix:

	Predicted as noise	Predicted as single-nuclei
Actual noise	TN=19	FP=2
Actual single-nuclei	FN=5	TP=51

- Accuracy: Tells how often the classifier is correct:

  Accuracy= $\frac{TP+TN}{N}$, N is the total number of annotated objects

  N= TN + TP + FP + FN

  Accuracy= $\frac{51+19}{77} = \frac{70}{77}$ = 0.91 → 91%

- True positive rate (sensitivity): How often is a positive predicted when it actually is positive.

  (com de sensible es el teu algortime per predir la clase que estas evaluant)

  FN + TP: all the experts true single-nuclei

  Sensitivity= $\frac{TP}{FN+TP}$

  Sensitivity= $\frac{51}{5+51} = \frac{51}{56}$ = 0.91 → 91%

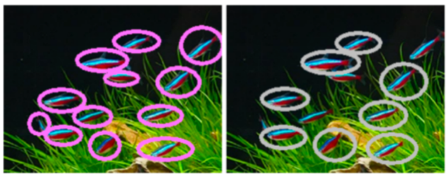
- Specificity: How often is a negative predicted when it actually is negative

  TN + FP: all the experts true noise objects

  Sensitivity= $\frac{TN}{TN+FP}$

  Sensitivity= $\frac{19}{19+2} = \frac{19}{21}$ = 0.9→ 90%

Quiz: True positive rate
You have made an algorithm that can locate neon fish in an aquarium. An expert has marked all neon fish in an image. (left) The result of the algorithm is seen in the right. What is the true positive rate of your algorithm?



True positive rate = Sensitivity= $\frac{8}{4+8} = \frac{8}{12}$ = 0.66 →67%

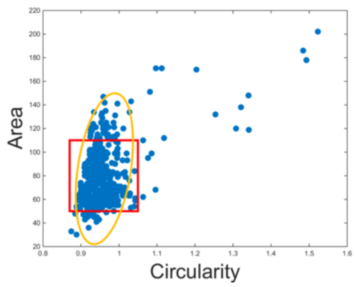
Optimising the classification:

- Changing the classification limits
- The rates will be changed: accuracy, sensitivity, specificity
- Very dependent on the task what is optimal
- Dependencies: Increasing true positive rate
    - ➢ Increased false positive rate
    - ➢ Decreased precision

Quiz: What measure is the most important?
- We want only single-nuclei cells
- We do not want to do an analysis of noise objects
- We are not interested in the true number of single nuclei

1.Low false positives
2.High true positives
3.High true negatives
4.Low false negatives. To optimize that we just want single-nuclei cells.

Advanced classification:
- Fitting more advanced functions to the samples
- Multivariate Gaussians
- Mahalanobis distances



Feature Engineering	Deep Learning
- Given a classification problem (cars vs. pedestrians) - Use background knowledge to select relevant features (area, shape, appearance) - Use multivariate statistics to classify - Depending on the selected features - Use it when data is limited - When it is rather obvious what features can discriminate -	- You start with a dummy classifier - Feed it with lots and lots of data with given labels → when having a lot of annotated data - The network learns the optimal features→ when it is not clear what features work - Layer/network engineering

Class VII (12/10)

Geometric Transformation and Image Registration

→ Moving and changing the dimensions of images.

Change Detection

<u>Ex:</u> Patient imaged before and after surgery. What are the changes in the operated organ? The patient cannot be placed in the exact same position in the scanner.

Image Registration

→ Change one of the images so it fits with the other.
Formally:

- Template image
- Reference image
- Template is moved to fit the reference. We are computing the difference.

→ It is a **Geometric Transform;** the pixel intensities are not changed. The "pixel values" just change positions (same value, but different places).

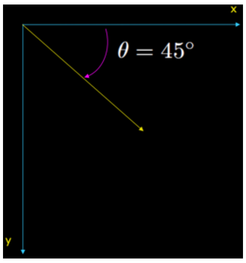
Transformations:

→ Transformation Mathematics: There is a transformation of positions.
From position (x,y) in the input image f to a new position at (x',y') in output image g.
A mapping function is needed:

$$x' = A_x(x,y) \qquad y' = A_y(x,y)$$

1. **Translation.**
   The image is shifted- both vertically and horizontally.
   $$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \text{ (matrix notation)}$$
2. **Rotation.**
   The image is rotated around the centre or the upper left corner. (!!! Use properly degrees and radians)

   The rotation coordinate system is the following:



   $$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} \text{ (matrix notation)}$$

Rigid Body Transformation = Translation and rotation of a rigid body. - Angles and distances are kept.

3. **Scaling.**
   The size of the image is changed. We usually use uniform scaling.
   Scale factors:
   - X-shear factor S_x
   - Y-shear factor S_y

   We usually use uniform scaling: $S_x = S_y$
   $$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} S_x & 0 \\ 0 & S_y \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} \rightarrow A = \begin{pmatrix} S_x & 0 \\ 0 & S_y \end{pmatrix} \text{ (transformation matrix)} \rightarrow \begin{pmatrix} x' \\ y' \end{pmatrix} = A \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

Similarity Transformation = Translation and uniform scaling - Angles are kept - Distances change

4. **Shearing.**
   Pixel shifted horizontally or/and vertically. It is less used than translation, rotation and scaling.
   Shearing factors:
   - X-shear factor B_x
   - Y-shear factor B_y

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & B_x \\ B_y & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

5. **Affine transformation**
   The collinearity relation between points.
   Ex: Three points which lie on a line continue to be collinear after the transformation.

6. **Other advanced transformations.**

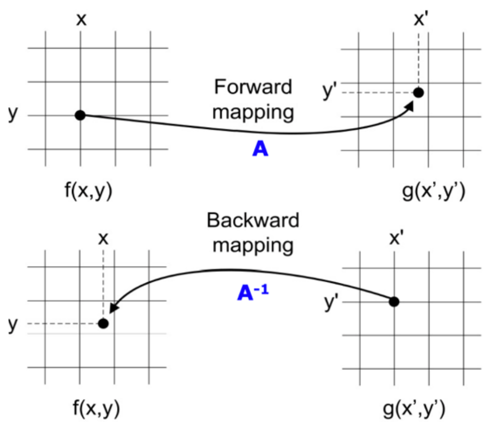
<u>Combining transformations:</u> By matrix multiplication. The order of matrix multiplication matters!

- Scale + Rotate

Scale	$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} S_x & 0 \\ 0 & S_y \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$
Rotate	$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$
Combined	$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} S_x & 0 \\ 0 & S_y \end{pmatrix} \cdot \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} = A_s \cdot A_R \begin{pmatrix} x \\ y \end{pmatrix}$

<u>Input-to-output:</u>
Run through all pixels in input image and find position in output image and set output pixel value.
But… we might have **missing values** in the output image. The input to output transform creates holes and other nasty looking stuff.

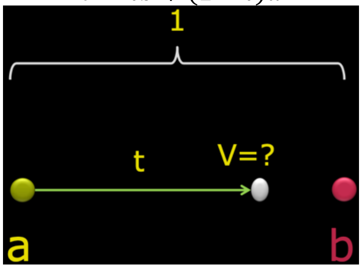
- Problem: We want to fill all the pixels in the output image and not just the pixels that are “hit” by the pixels in the input image.
- Solution: Go “backwards”. From the output to the input. → **Inverse Transformation = Backward mapping.**



→ <u>Bilinear interpolation:</u> The value is calculated from 4 neighbours, based on the distance to neighbours.

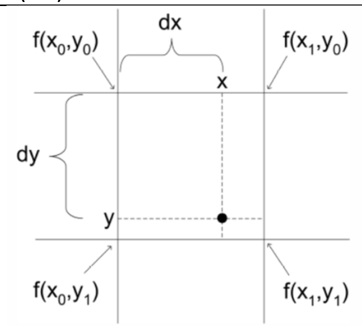
Linear Interpolation (1D)

$$v = tb + (1-t)a$$



Bilinear Interpolation (2D)

$$\begin{aligned} g(x', y') \quad = \quad & f(x_0, y_0) \cdot (1 - dx)(1 - dy) + \\ & f(x_1, y_0) \cdot (dx)(1 - dy) + \\ & f(x_0, y_1) \cdot (1 - dx)(dy) + \\ & f(x_1, y_1) \cdot (dx \cdot dy) \ , \end{aligned}$$



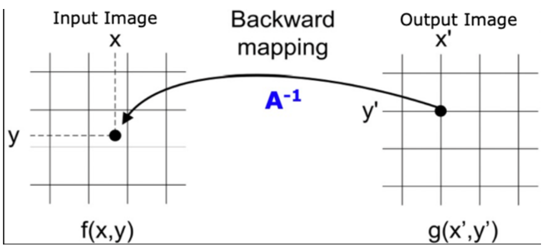
<u>Ex:</u> In a given point (x,y) = (173.1, 57.8), the four nearest pixels are:

x	y	værdi
173	57	110
174	57	140
173	58	156
174	58	101

$f(x, y) = 110 \cdot (1 - 0.1)(1 - 0.8) + 140 \cdot (0.1)(1 - 0.8) + 156 \cdot (1 - 0.1)(0.8) + 101 \cdot (0.1)(0.8) = 143$

Output-to-input transformation / Backward mapping

1. Run through all the pixel in the output image.
2. Use the inverse transformation to find the position in the input image.
3. Use bilinear interpolation to calculate the value.
4. Put the value in the output image.



$$\begin{pmatrix} x' \\ y' \end{pmatrix} = A \cdot \begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \end{pmatrix} = A^{-1} \cdot \begin{pmatrix} x' \\ y' \end{pmatrix}$$

(affine transformation) *(inverse transformation)*

<u>Ex:</u> Transformation

The point (x, y) = (45,23) is transformed using $\begin{pmatrix} 0.5 & 2 \\ 2 & 0.8 \end{pmatrix}$. The result is translated with (15,20).

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 0.5 & 2 \\ 2 & 0.8 \end{pmatrix} \cdot \begin{pmatrix} 45 \\ 23 \end{pmatrix} + \begin{pmatrix} -15 \\ 20 \end{pmatrix} = \begin{pmatrix} 53.5 \\ 128.4 \end{pmatrix}$$

Image Registration
The act of adjusting something to match a standard. Align images.

Applications:
- Monitoring of change in the individual → Change detection. Ex: Patient image before and after operation. Images need to be aligned before comparison.
- Fusion of information from different sources in a meaningful way
- Comparison of one subject with others
- Comparison of groups with others
- Comparing with atlas

Reference and template image
1. The reference image R and the template image T
2. Transform the template so it fits the reference
3. Combine geometrical transformations
4. Find the transformation matrix, A for the best match.

Similarity measures
Aim: Transform the template, so it looks like the reference. (looks like = similarity measure)

1. **Image similarity:** Subtract the two images and see "what is left"

2. **Landmark similarity:** Landmarks from the two images should be "close together".

→ Landmark Based Registration.
The landmarks place on both reference and template image should have *correspondence.*
Steps:
- Select Landmark points:
  Types:
  - Anatomical landmark: A mark assigned by an expert that corresponds between objects in a biologically meaningful way.
  - Mathematical landmark: A mark that is located on a curve according to some mathematical or geometrical property.
  - Pseudo landmark: A mark that is constructed on a curve based on anatomical or mathematical landmarks.

- Find a transformation that maps the coordinates of the reference to the coordinates of the template.

$$p' = T(p)$$

  It transforms point p into point p', where T is the geometrical transformation *(translation, rotation, rigid body transform, similarity transform)*

  The parameters of the transformation is a vector with p elements: $\omega \in R^p$
  The type of transformation determines the number of parameters.
  (translation p=2, rotation p=1, scaling p=1, rigid body transform p=3)

Objective function
- It measures how well two-point sets match, using a cost function that describe how to evaluate the match. This can be a sum-of-square distance function.
- Point sets could be landmarks.

$$F = \sum_{i=1}^{N} D\,(T\,(a_i)\,, b_i)^2$$

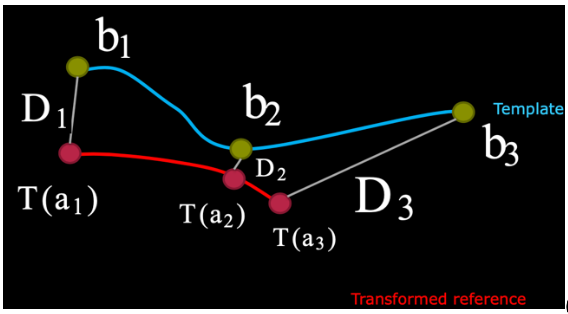
  a_i: transformed points from the reference image
  b_i: points from the template image
  D: distance between points.

$$F = \sum_{i=1}^{3} D\,(T\,(\,a_i)\,, b_i)^2 = D_1^2 + D_2^2 + D_3^2$$

Minimization / Optimization: Find the set of parameters that minimized the objective function.

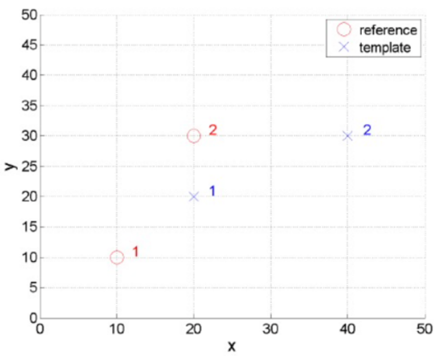
- Optimization strategy: Analytic (exact solution) vs. Numerical?

$$\hat{w} = arg \min_{w} F$$



(pure translation)

Ex: Objective function. Find the optimal translation.



$$F = D_1^2 + D_2^2 = 200 + 400 = 600$$

Optimal transformation

Optimal translation: $a'_i = a_i + t$

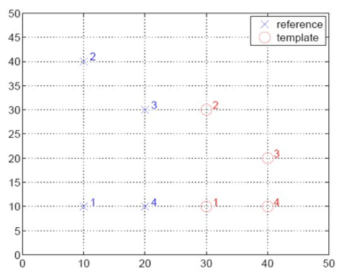
- Objective function: $F = \sum_{i=1}^{N} ||(a_i + t) - b_i||^2$
- Parameters: $w = (\Delta x, \Delta y) = t$
- To find the optimal function value: simply differentiate with respect to w:

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

- Optimal function: $\hat{t} = \bar{b}$ - $\bar{a}$ $\quad \bar{a} = \frac{1}{N}\sum_{i=1}^{N} a_i$

Ex: Optimal translation

(20, -5)



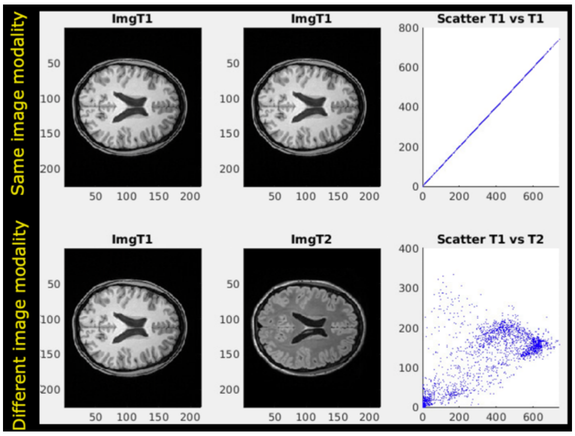
Optimal rigid body transformation: $a'_i = Ra_i + t$, $R = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$

- Objective function: $F = \sum_{i=1}^{N} ||(Ra_i + t) - b_i||^2$
- Parameters: $w = (\Delta x, \Delta y, \theta)$
- To minimum od the objective function can be found in several ways.
- The rotation can be found analytically by **singular value decomposition.**
  (Tip: Always start by matching the centre of masses

Similarity measures. Joint intensity histograms.

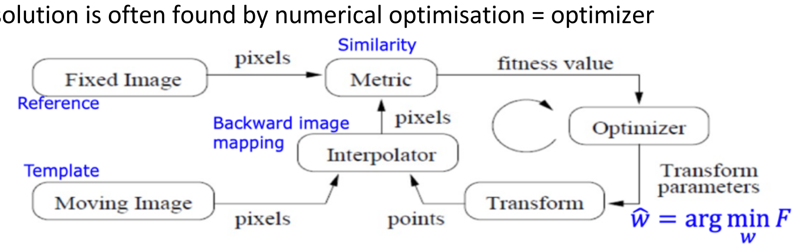
Obtaining landmarks is really time consuming.
An alternative could be: *Joint intensity histograms.*

Similarity measures to find transformation.
- The perfect registered: Optimal joint intensity agreement
- Small translation difference: Lower joint intensity agreement



Many methods exist, but two types dominate:
1. **Cross-correlation based.** Fat to estimate, not optimal choice if different image modalities.
2. **Joint entropy based (= Mutual Information MI).** Slow to estimate, robust also if different modalities.
   - An information content measure
   - Entropy (Shannon-Weiner): $H = -\sum_i p_i \log p_i$
   - Joint entropy – Mutual Information: $H = -\sum_{x,y} p_{x,y} \log p_{x,y}$
     Similarity measure: The more similar the distributions, the lower the joint entropy compared to the sum of the individual entropies.
     $$H(X,Y) \leq H(X) + H(Y)$$

The image registration "pipeline"
- Register Template image to Reference image via geometrical transformations
- Select a similarity measure to map coordinates from template
- Objective function – Find optimal parameters:
- The solution is often found by numerical optimisation = optimizer



Class VIII – 26/10

Hough Transformation and Path Tracing

Line Detection

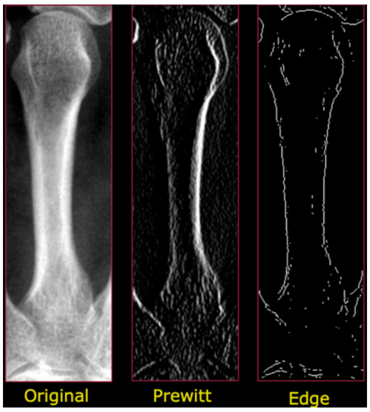
Find the lines in an image

What is a line?

- It can be the entire object (large scale)
- Can also be the border between an object and the background (small scale)
- Normally only locally defined

<u>Enhancing lines:</u>

- We want to locate the borders → enhance them
- Filtering: Prewitt
- Edge detection



Prewitt:

Vertical

-1	0	1
-1	0	1
-1	0	1

Horizontal

-1	-1	-1
0	0	0
1	1	1

- The result of the edge filter is a selection of white pixels → the line
- Some of them define a line *(Not a perfect straight line, linelike)*
- How do we find the collection of points that define the line?

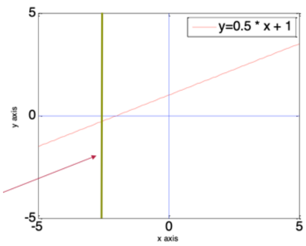
<u>Mathematical line definition:</u>

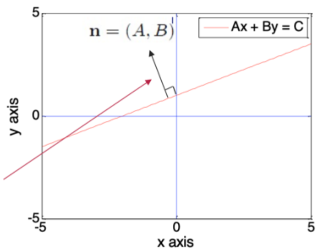
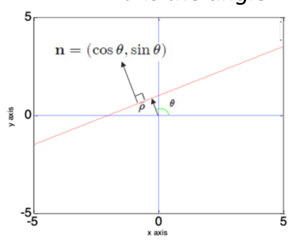
(Slope-intercept form)

$$y = ax + b\,,$$

where a is the slope and b is the intercept

Cannot represent lines
That are vertical



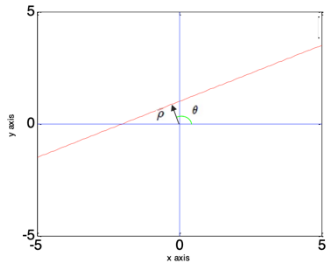
General definition (the normal form)	Normal form parameterisation
$Ax + By = C$ *with,* $A^2 + B^2 = 1$ 	$x\,cos\theta + y\,sin\theta = \rho$ *Where* - ρ *is the distance from the origin* - θ *is the angle* 

$$(cos\theta)^2 + (sin\theta)^2 = 1$$
$$A^2 + B^2 = 1$$

Therefore, a line can be defined by two values:

- ρ
- θ

A line can therefore also be seen as a point in a ***(θ, ρ)-space***

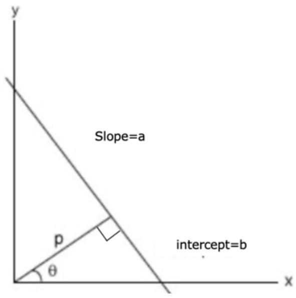


<u>Converting lines between definitions:</u>
From normal form to the slop-intercept form

- The normal form: $\rho = x\ cos\theta + y\ sin\theta$
- The slope-intercept form: $y = ax + b$

$\rho = x\ cos\theta + y\ sin\theta$
$-x\ cos\theta + \rho = \ y\ sin\theta$
$-x\ cot\theta + \rho\ cosec\theta = \ y$
$y = x \cdot (-\boldsymbol{cot\theta}) + \boldsymbol{p(cosec\theta)}$

Slope=a *Intercept=b*

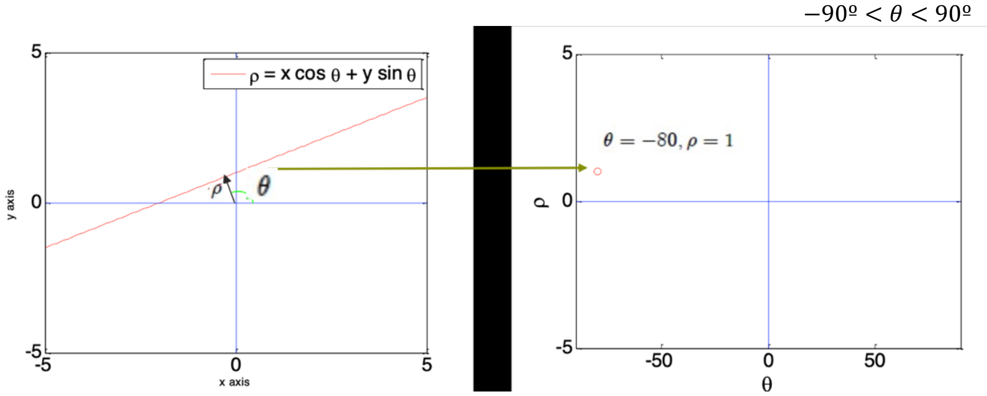


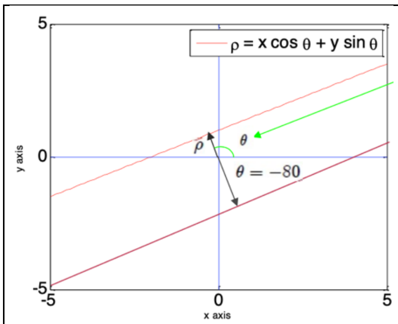
About angles

- $\theta \in$ [0º, 180º] *in the course notes*
- $\theta \in$ [-90º, 90º] *in MATLAB and in this presentation*

Hough Space

$-90º < \theta < 90º$





More about angles
$\theta = \ -80º$ → why ? = $100º$

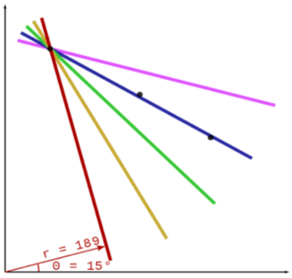
MATLAB only allows $-90º < \theta < 90º$
→ look at the mirror-projection of the normal

ρ is used to determine if it is the “upper” or “lower line”

<u>Hough space</u> = Basically a tool to find a line through points.

1. Point coordinates: (x,y)
2. Define origin
3. Hough parameters space: (r, θ)
4. Map all possible lines through a point for different θ
5. "Vote" which line fit best through points

Points:



Hough space:

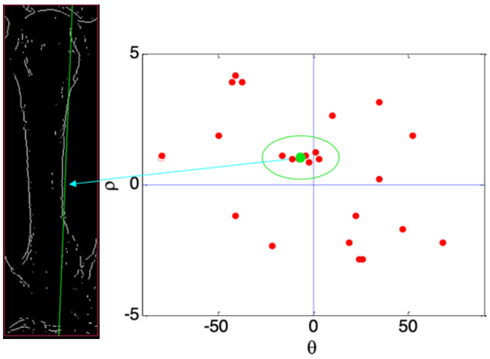
θ	r
15	189.0
30	282.0
45	355.7
60	407.3
75	429.4

How to use the Hough space

What if every little "line-segment" was plotted in the Hough-space?

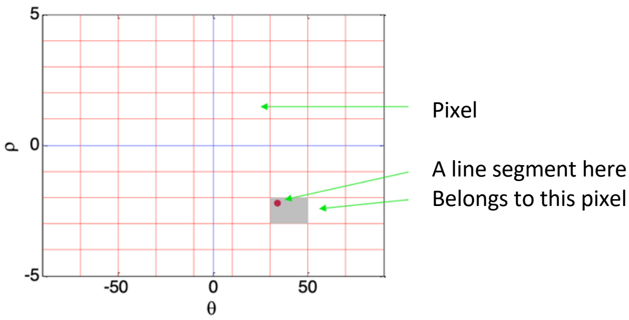
<u>Filled Hough-Space</u>

- All "line segments" in the image examined
- A "global line" can now be found as a cluster of points (although is practice it is difficult to identify clusters)



<u>Hough transform in practice:</u>

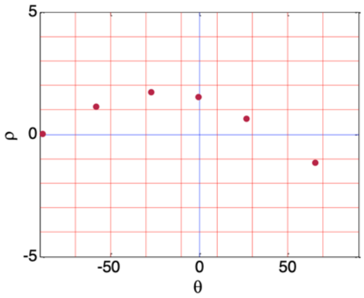
- Hough Space is represented as an image
- It is quantisized – made into finite boxes

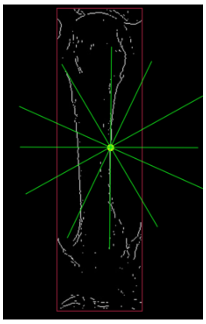


<u>Hough transform as a voting scheme:</u>

- The pixels in the Hough space are used to vote for lines.
- Each line segment votes by putting one vote in a pixel
- The pixels are also called accumulator cells

Hough transform per pixel:
- In practise we do not use line segments
- Each pixel in the input image votes for all potential lines going through it.



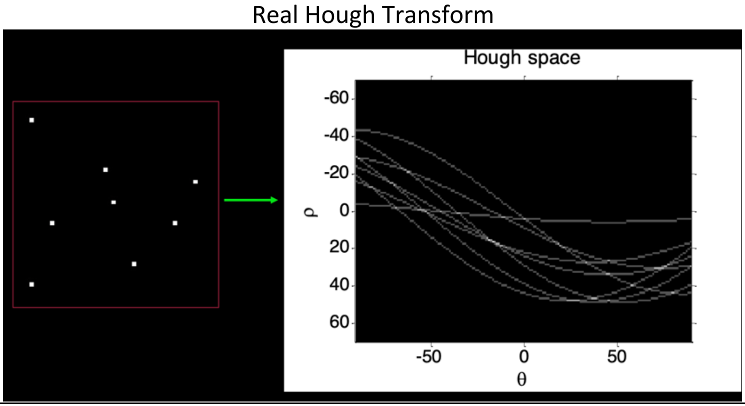


- Sinusoid
- (x,y) are fixed

$$x \cos\theta + y \sin\theta = \rho$$

➢ Go through all θ and calculate ρ

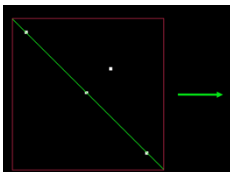
Real Hough Transform

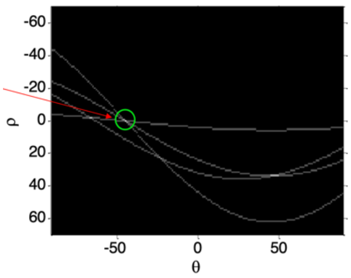


Real Hough Transform and lines

Spot the line!

A maximum where Hough Pixel has value 3



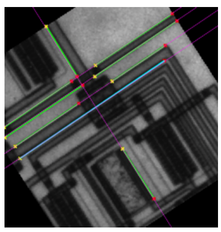


Finding the lines in Hough space

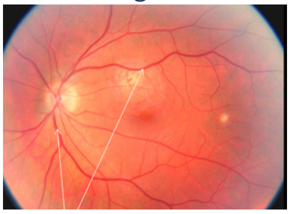
- The lines are found in Hough space where most pixels have voted for there being a line
- Can be found by searching for maxima in Hough Space

Practical guide to the Hough Transform

1. Start with an input image
2. Detect edges and create a binary image
3. Compute Hough transform and locate the maxima
4. Draw the lines corresponding to the found maxima
5. Here the cyan line is the longest



Path Tracing



Arteries and veins.
Fundus image

- The diameter as function of distance to the optic cup tells something about the patients' health
- We need to find the arteries and veins
- Path tracing is one solution

- Path= Defined as a curve in an image defined as something that is different from the background.
- In the case above it is a dark line
- Pre-processing can for example turn edges into dark lines.

Dynamic Programming

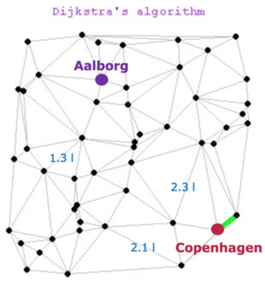
Break up large problem into many small sub-problems.
A classic algorithm:

- Dijkstra's algorithm
- One source to all nodes shortest path

- <u>**Path tracing**</u>

GPS devices use path tracing. It is based on **graph algorithms** (where a city is a node, and a road is an edge. The weight of the edge is the fuel cost).

How do come from Copenhagen to Aalborg using the least fuel?



- <u>**Images as graphs:**</u>
  - Each pixel is a node
  - Pixel neighbours are connected by edges
  - The edge cost (c(r,c)) is the pixel value
  - Directed graph
  - Imagine a car driving on the image
  - Called ***cost image***
  - The pixel value at (r,c) equals the cost
  - The path P consist of pixels
  - The sum of pixel values in the path:

$$c_{tot} = \sum_{(r,c)\in P} C\ (r,c)$$

Simplified problem:

- Track dark lines
- Path going from top to bottom
- No sharp turns- smooth
- Problem:
  - From the top to the bottom
  - Sum of pixel values should be minimal → We are looking for the dark path

- **Path Cost:**
  A path is defined as (r,c) coordinates: $c_{tot} = \sum_{(r,c)\,\in P} C\,(r,c)$

140	190	73	19	60
130	212	14	100	145
150	20	80	135	120
157	140	33	199	100
121	234	45	210	86

P= [(1,3),(2,3),(3,2),(4,3),(5,4)]

What is C_{tot} (total cost)? → 350

!!! This is NOT the optimal path.
How do we compute the path P that has a minimum C_{tot}?
Test all possible paths? It is impossible to test all number of possibilities.

Quiz: A path has been found in the image
P= [(1,4), (2,4), (3,5), (4,5), (5,5),(6,4)]. A MATLAB matrix coordinate system is used. What is the total cost of the path?

208	157	234	19	145	79
62	121	73	14	120	135
237	90	193	135	3	42
89	212	192	199	86	154
50	149	97	238	41	67
64	140	145	33	203	167

$C_{tot} = 19+14+3+86+41+33 = 196$

- **Path restriction: The rules**
  - Path is only allowed to
    - Go down
    - Move one pixel left or right
  - Longer jumps no allowed

Accumulator image – Dynamic Programming

→ It keeps track of the accumulated cost for efficient paths finding.

Computing the accumulator image:

1. Step 1. Copy first row of input image.

140	190	73	19	60
130	212	14	100	145
150	20	80	135	120
157	140	33	199	100
121	234	45	210	86

140	190	73	19	60
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

2. Step 2. Fill second row.

$$A(r,c) = I(r,c) + \min\left(A(r-1,c-1), A(r-1,c), A(r-1,c+1)\right)$$

Ex:

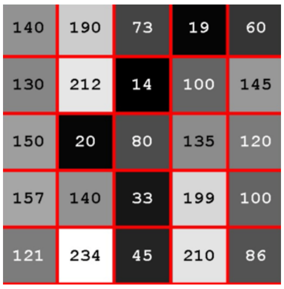
$$A(2,3) = 14 + \min(190, 73, 19) = 14 + 19 = \mathbf{33}$$

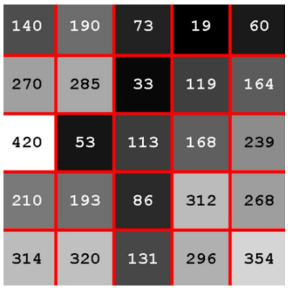
140	190	73	19	60
130+140	212+73	**14+19**	100+19	145+19
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

140	190	73	19	60
270	285	**33**	119	164
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

3. Step 3. Fill all rows by looking at the previous row.

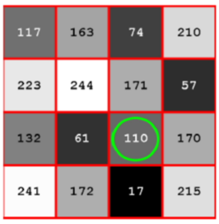
$$A(r,c) = I(r,c) + \min\left(A(r-1,c-1), A(r-1,c), A(r-1,c+1)\right)$$





Quiz: Accumulator Image

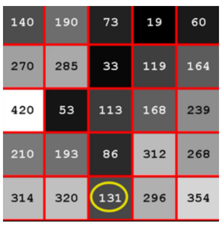
An optimal path has been found in the image. What is the value of the accumulator image in the marked pixel?



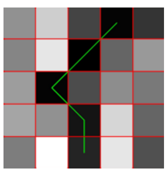
117	163	74	210
223+117=340	244+74=318	171+74=245	57+74=131
132+318=450	61+245=306	110+131=**241**	170+131=301
241+306=547	172+241=413	17+241=258	215+241=456

$$A(3,3) = 110 + \min(318, 245, 131) = 110 + 131 = \mathbf{241}$$

4. Step 4. The end of the optical path can now be found.



5. Step 5. Trace the path in the backtracing image.
   - It keeps track of where the path came from
   - Each pixel stores the column number



0	0	0	**0**	0
1	3	**4**	4	4
1	**3**	3	3	4
2	2	**2**	3	4
2	3	**3**	3	5

Quiz: *Backtracing*
An optimal path has been found in the image. The backtracing image is seen below and the optimal path ends in the marked pixel. A MATLAB coordinate system is used. What is the optimal path?

0	0	0	0	0
1	3	3	3	5
2	2	2	4	4
1	1	4	5	5
1	1	4	4	4

P= [(1,3),(2,2),(3,1),(4,1),(5,2)]

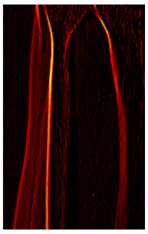
Pre-processing

We would like to track paths that are not dark curves.

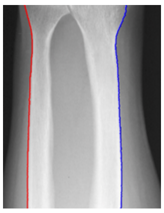
Quiz: *X-ray pre-processing*

- ***A) Gaussian smoothing*** → It could remove some of the noise.
- ***B) 255-I*** → It inverts the colour.
- ***C) Gradient filter***
- *D) Registration* → No need. We do not have two images.
- *E) Morphological operation*

Example:



1. Pre-processing
Edge filtered image (Gaussian smoothing followed by Prewitt)



2. Path tracing on pre-processed image
Paths found on pre-processed image and intensity inverted
Path that minimizes the cost.

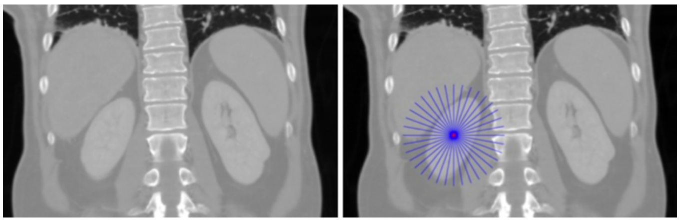
Quiz: *Optimal Path*
A 5x5 image is filled with values given the grey level run length encoding:
2,180,1,15,3,112,1,8,4,177,1,20,4,195,1,12,3,242,2,25,3,9.
After that the optimal path is found. What is the total cost?

180	180	**15**	112	112
112	**8**	177	177	177
177	**20**	195	195	195
195	**12**	242	242	242
25	25	**9**	9	9

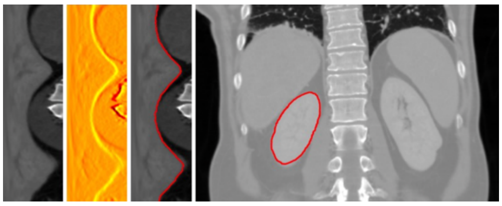
$C_{tot} = 15 + 8 + 20 + 12 + 9 = 64$

Locating Circular Structures

Processing step to do path tracing



- Define origin inside structure
- Send out spokes



- Each spoke is a line in a new image (surface-layer detection)
- Prewitt
- Dijkstra’s algorithm
- Map back the spokes into image

Class IX (2/11)

Statistical Models of Shape and Appearance

Typical Scenarios:

1. *Ex1:* Doctor X believes that he can "see" on a hand X-ray if the patient is in risk of arthritis. Specifically, Doctor X is sure that the shape of joints is an estimator of arthritis.
2. *Ex2:* MR images have been captured of a large group of people. Cognitive abilities measured as well. Is there a correlation between how brain looks and how we behave? Does the shape of corpus callosum tell us something?
3. *Ex3:* An experienced hearing aid fitter has been seen a lot of ears. Some hearing aid users are very difficult to fit. There is a large variation in the shape of ears. Ear canals change shape when people chew. Is it possible to learn about the shape and use it?

Shape Analysis

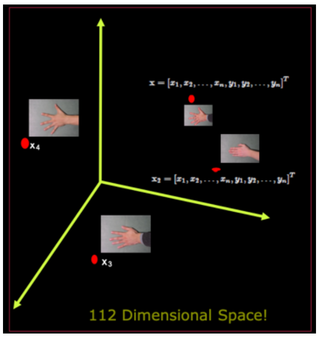
→ Shape is defined using ***landmarks*** (placed by an expert). It is all geometrical information that remains when location, scale and rotational effects are removed.
Ex: Outer contour of the hand.

→ **Point correspondence:** The landmarks are placed on the same place on all shapes in the training set.

- The shape is represented as an array of (x,y) coordinates
- All coordinates are put into one vector.
- Thus n=56 points → vector with 112 elements. A vector can also be seen as a point in space.
  1: $\{x_1, y_1\}$
  2: $\{x_2, y_2\}$
  N: $\{x_n, y_n\}$
  $\mathrm{x} = [x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_n]^T$

  Not 2D space, not 3D space, not 4D space… 112-dimensional space!
  Ex: A hand has a position in this space.
  Another hand appears
    - In the same space
    - Different positions = different shape

  All hands have a place in this space.
  The space covers much more than the 'anatomical' space.

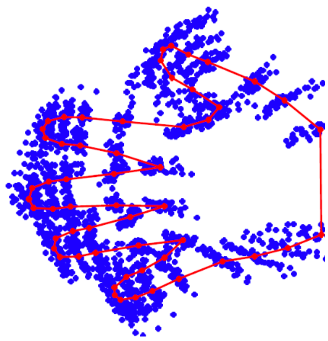


- Shape analysis: Similar shapes are placed on "planes" in the shape-space = ***Manifold*** (also called)

Shape alignment

- 40 training images of hand
- 56 landmarks on each
- Placed in random location (translation + rotation)

→ The landmarks from all hands need **alignment.**



Group wise registration.

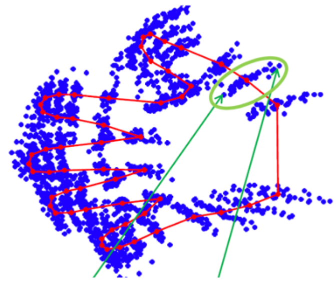
- Not one-to-one
- All to the **average shape** $\bar{\mathrm{x}}$
  $\bar{\mathrm{x}} = \frac{1}{N}\sum_{i=1}^{N} x_i\ , \quad \bar{\mathrm{x}} = [x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_n]^T$

Procrustes Analysis (alignment) → Landmark based registration

1. Average Shape is set to shape #1 (start by average shape = shape #1)
2. Align Shape #2 to Shape #1
   Shape #2 is transformed to fit the average shape (translation, rotation, scaling = similarity transformation).
   Result: Shape #2 is placed on top of the average sha
3. Align all shapes to Shape #1.
4. Register all shapes to the average shape. (Landmark based registration).
5. Recompute the average shape
6. If average shape changed return to step 2.

Individual landmark variation (over the training set)
What shape is the variation?

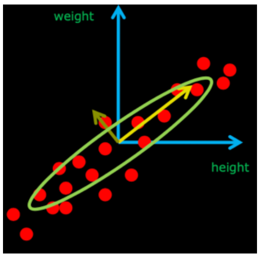
→ *Principal Component Analysis (PCA)*
For a geometrical description of your variance
(Apart from removing noise dimensions, redundancy)



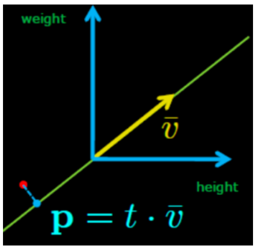
Shape #16 Shape #27

PCA Analysis

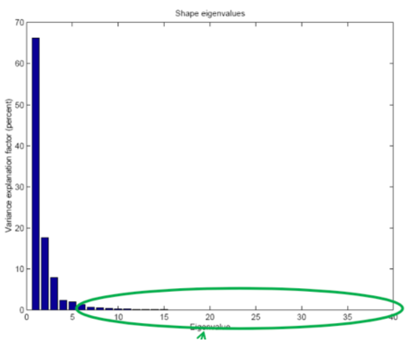
(Main axis in data to project point to the line, eigenvectors, eigenvalues, size of eigenvalues describe explained variance)



- PCA analysis on individual landmarks
- Describes the major direction of variation
- Landmarks are correlated
- The movement over the shape is connected
- Return to shape space



- Points projected to the line
- A point can now be described by one parameter *t*
- We have reduced the number of dimensions



Noise dimensions

How many dimensions should we keep?

- Plot the Eigenvalues
- Explains how important each dimension is
- Cut away noise dimensions

PCA in shape space:

- Instead of doing PCA on 2D points we do it on 112D points.
- Examine if our 40 shapes are lying on a plane in 112D space.
- We find the directions that spans the maximum variation in shape space.

1.Compute the shape average

$$\bar{\mathbf{x}} = \frac{1}{s}\sum_{i=1}^{s}\mathbf{x}_i$$

2. Do the eigenvector analysis
Compute the covariance of the shape data.

$$\mathbf{S} = \frac{1}{s-1}\sum_{i=1}^{s}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T$$

3. Visualizing variation (1st component analysis)

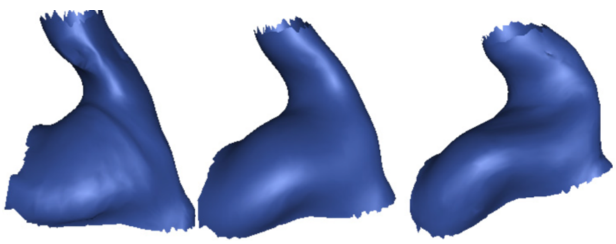
$$\mathbf{x} \approx \bar{\mathbf{x}} + \mathbf{\Phi}\mathbf{b}$$, $\mathbf{\Phi}$ contains the t eigenvectors

Visualisation of the major variation of the shape over a population

Examples:

Hearing Aid Design

- Main variation of the shape of the ear canal
- Found using principal component analysis
- First mode of variation
- 7 modes explain 95% of the total variation



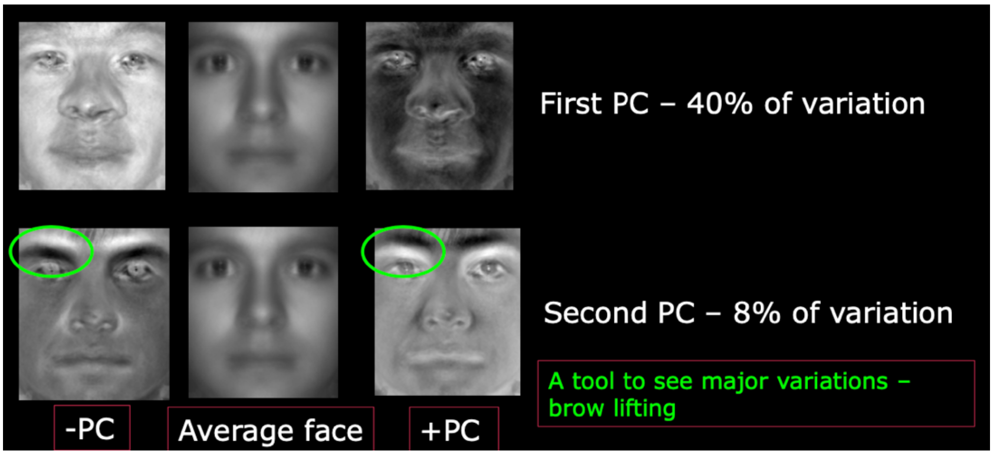
Average-1. Mode *Average* *Average+1. mode*

Modelling shape and appearance

- A model that can both model the shape of an object and the appearance (the texture).
- **Texture:** The pattern of intensities (or colors) across an image patch

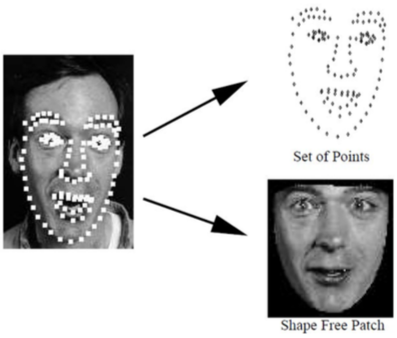
- Face Data→ Analysing the deviation from the mean face. We want to do the principal component analysis on the deviations from the average face.

- Visualizing the PCA faces. Main deviations from the average face.



- Eigenfaces: Modelling texture
  - The modelling of the pure appearance
  - Without removing variation in shape
  - No decoupling of shape and appearance

- Decoupling shape and texture
  - Warp each face to average shape using landmarks
  - Non-linear geometrical transformation
  - Sample the texture from the warped face



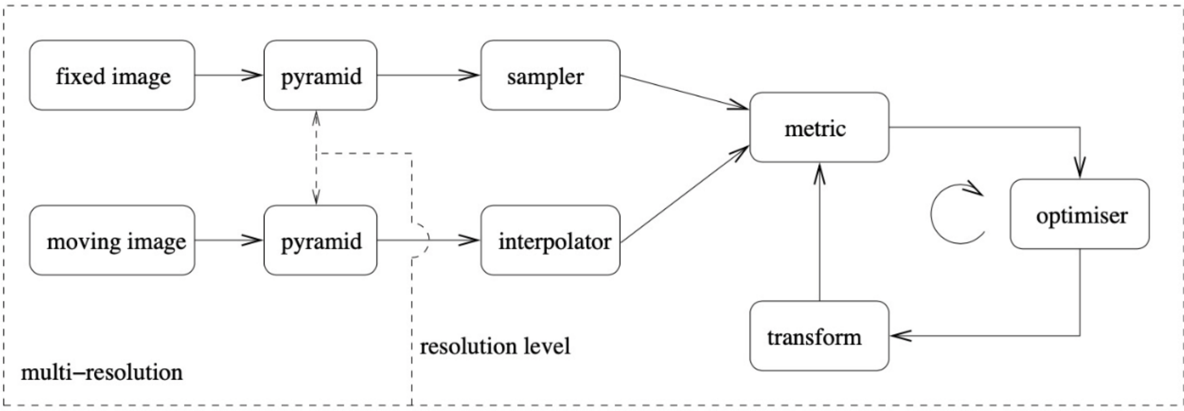
- Eigenfaces on warped faces
  - Same PCA modelling as the Eigenfaces approach
  - Just slightly different notation

- Combined shape and appearance model

Class X (8/11)

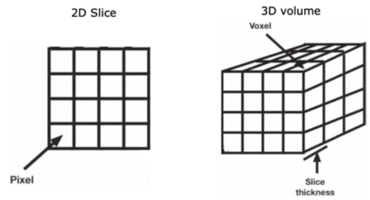
Advance Image Registration

Image Registration pipeline



- Fixed Image: Template image
- Moving Image: Reference Image

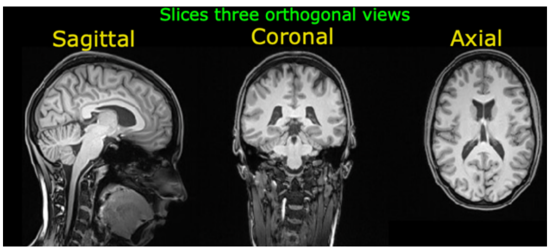
Image volumes



- Image slice: 2D (NxM) matrix of pixels
- Image volumes: 3D (NxMxP) matrix of voxels
    - An element is a volume pixel = **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. Surface
    - Surface insight
    - Limited to clear surfaces



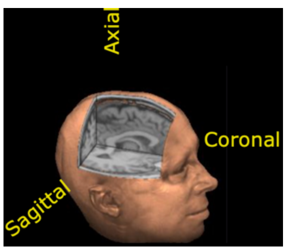
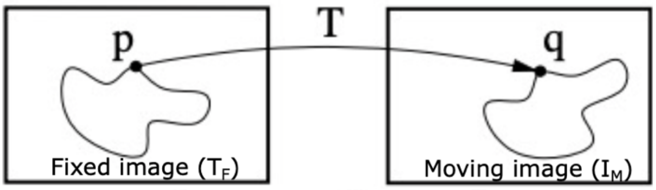


Image volumes:

- Stacked sliced: 2D to 3D
    - Object cut into slices, imaged and stacked
- Registration challenges
    - Geometrical distortions between slices
    - Stacking 3D volumes
    - Multi image resolution: Fit Region-of-interest image to whole object image

1.Geometric transformations

Based on transformation matrices.



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

Translation 2D vs. 3D

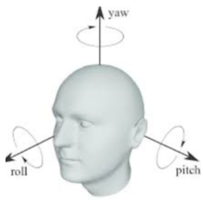
- The image is shifted
  - 2D: Inspect one slice plan → (x,y)-plans → $\begin{pmatrix}\Delta x\\ \Delta y\end{pmatrix} = \begin{pmatrix}60\\ 20\end{pmatrix}$
  - 3D: Inspect three slice plans→ (x,y,z)-plans → $\begin{pmatrix}\Delta x\\ \Delta y\\ \Delta z\end{pmatrix} = \begin{pmatrix}60\\ 20\\ 15\end{pmatrix}$

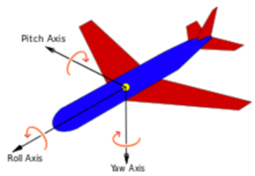
Rotation 3D

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

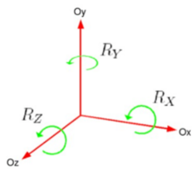
<u>Coordinate system</u>

- Three elements' rotations round the axes of the coordinate system.
- Pitch, Yaw and Roll: defined differently for different systems (type related to the forward direction)





- Three composed element rotations → angles: α, β, γ
- The order matters (several conventions exist)
- The most important is to know your origin



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

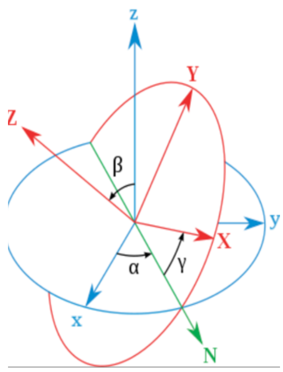
Pitch *Roll* *Yaw*

<u>3D Rotation coordinate system</u>

- The Euler angel convention:
  - ➢ α: Around the **z-axis.** Defines the **line of nodes (N)**
  - ➢ β: Around the **X-axis** defined by **N**
  - ➢ γ: Around the **Z-axis** from **N**
- The order of coordinate system rotations:

  Rotation order around the:

  1. Z-axis: Initial: Original frame (x,y,z): α
  2. X-axis: First coordinate system rotation (*X,Y,Z*): β
  3. Z-axis: Second coordinate system rotation (*X,Y,Z*): γ



Quiz: Affine 3D transformation.
How many possible parameters?

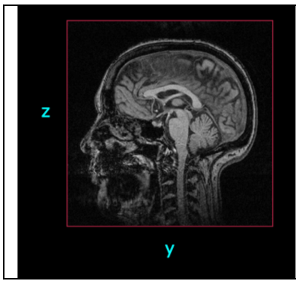
- ***12 parameters***
  - *Translation: p=3*
  - *Rotation: p=3*
  - *Scaling: p=3*
  - *Shearing: p=3*

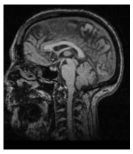
Scaling

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

$$\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{bmatrix}$$

Anisotropic Scaling:



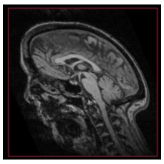


$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{bmatrix}$$

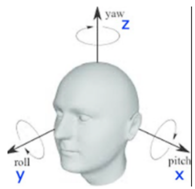
Shearing

- Pixel shifted horizontally or/and vertically
- Three parameters

$$\begin{bmatrix} 1 & S_{yx} & S_{zx} \\ S_{xy} & 1 & S_{zy} \\ S_{xz} & S_{yz} & 1 \end{bmatrix}$$



Shearing (z,y)-plan

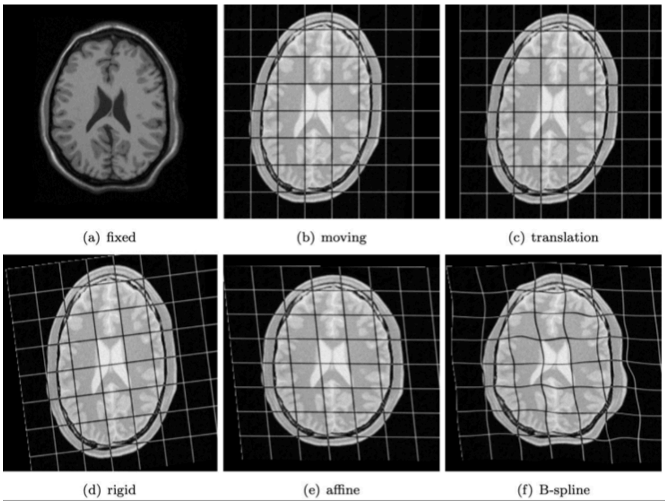


Combining transformation

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 $R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} R_y = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} R_z = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 & 0 \\ -\sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Scaling $A_S = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	Shear $A_Z = \begin{bmatrix} 1 & S_{xy} & S_{xz} & 0 \\ 0 & 1 & S_{yx} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Affine Transformation $A = A_T \cdot (R_x \cdot R_y \cdot R_z) \cdot A_z \cdot A_s$ *Rigid*	

Different transformations

- **Linear:** Affine transformation
- **Non-linear:** Piece-wise affine or B-spline.
    - **But** first apply the linear transformations!
    - First remove linear components, and then you apply the non-linear transformation.



2.Similarity measures

Anatomical Landmarks:
- Time consuming to obtain manually
- Alternative: **Joint Intensity Histogram**

Mean square difference (MSD)
- Compare difference in intensities
    - Same similarity measure we used for anatomical landmarks
    - Super-fast to estimate
- Many local minima's (sub optimal solutions)
    - Intensities are not optimal for this similarity metric

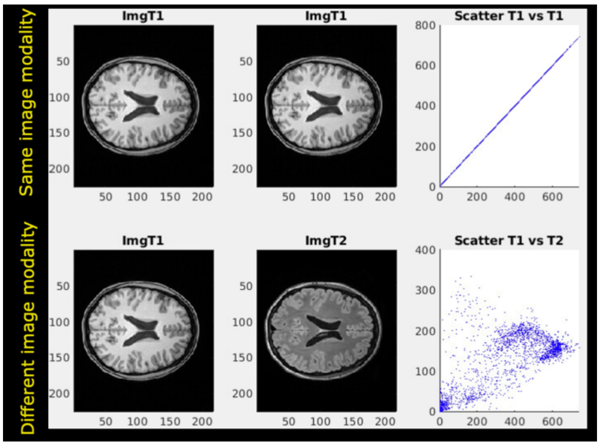
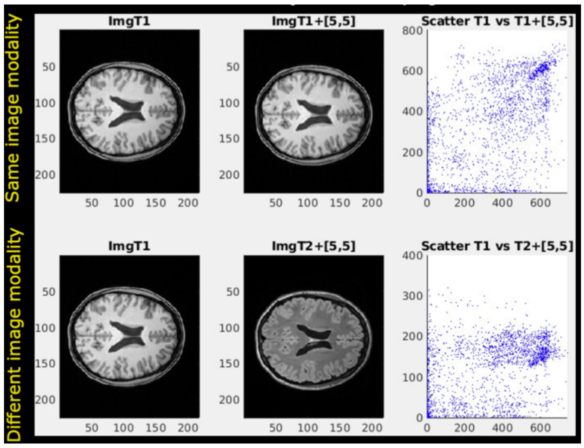
$$\mathrm{MSD}(\boldsymbol{\mu}; I_F, I_M) = \frac{1}{|\Omega_F|} \sum_{\boldsymbol{x}_i \in \Omega_F} \left(I_F(\boldsymbol{x}_i) - I_M(\boldsymbol{T}_{\boldsymbol{\mu}}(\boldsymbol{x}_i)) \right)^2$$

Cross-correlation
- 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

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

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

Joint intensity histograms:

Perfect registered: Optimal joint intensity agreement	Small translation difference: Lower joint intensity agreement
	

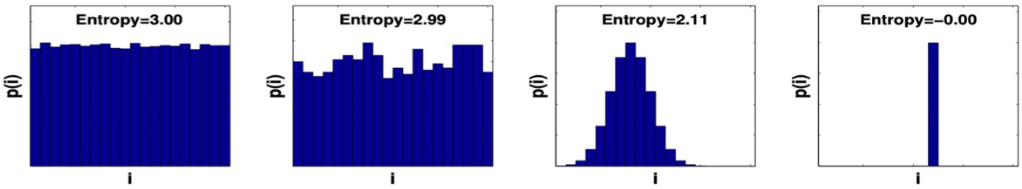
Entropy:

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

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

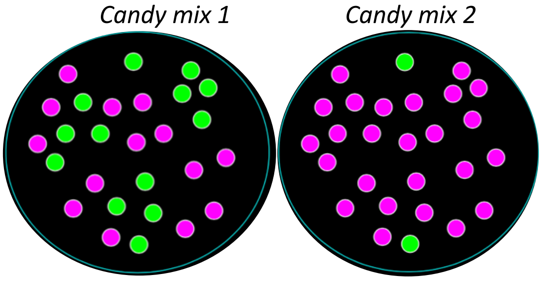
Where b is the base of the logarithm.

- *Bits: b=2 and bans: b=10*
- *Entropy is typically in bits. i.e., typical used in digital information*



<u>Quiz:</u> Highest entropy?
I went to the candy shop and wanted to select the cady mixture that have the highest entropy. Each candy mixture includes in total 27 pieces.
Which one should I select?

Candy mix 1 *Candy mix 2*



<u>Solution:</u> ***Candy mix 1.*** *It is the one that is more spread.*

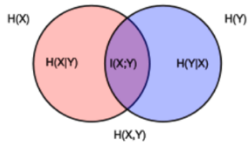
<u>Quiz:</u> What is the entropy of the candy mix 1?

Green: 13, Pink: 14, Total: 27 → pG=13/27, pP=14/27. → Entropy= pG·log$_2$(pG) +pP· log$_2$(pP)=0.99
<u>Solution:</u> ***0.99.***

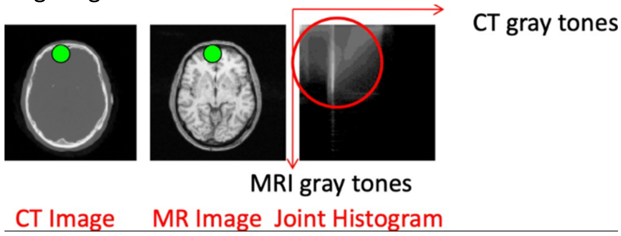
<u>Joint Entropy – Mutual information:</u>

- Joint Entropy: $H = -\sum_{X,Y} p_{X,Y} log\, p_{X,Y}$
- Similarity measure: The more similar the distributions, the lower the joint entropy compared to the sum of the individual entropies.

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



- Contrast in joint histograms: The histogram of the two images must reflect contrast to similar structures for image registration to be successful



3.The Optimizer

- We have an objective function describing:
  A **cost function (C)** based on a similarity metric
  - Quantifying how well a geometric transformation $T(w)$ map an image *(Reference/moving,* I_M) into another *(Template/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

$$w \in R^p$$

- 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

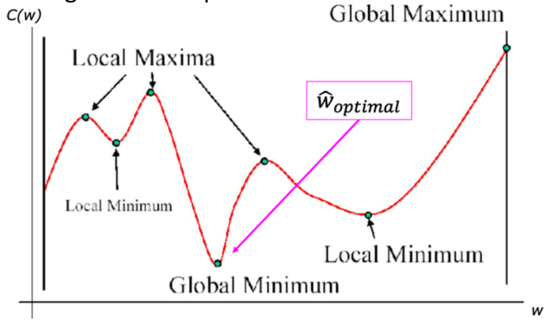
Optimization by minimization

- Find the parameter set that minimizes the objective function
- How to find the solution?
  - Analytical: Works fine for landmark registration with few points
  - Numerical: Iterative approaches to search for a solution

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

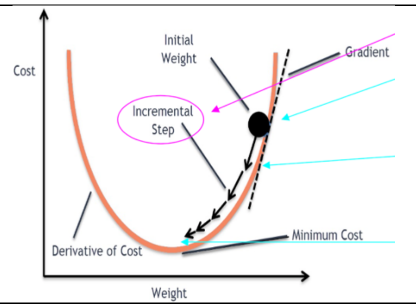
    We simply differentiate w.r.t. w: $\frac{\partial C}{\partial w}$ = 0

- The challenge:
    - w spans a p-dimensional space $w = [w_1, w_2, \dots, w_p]^T$
    - 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 minimum
      -Search all possible combinations of w? (not a good idea)
      -Systematically search the parameter space = good idea
    - Iterative optimisation strategies: Stepwise searching the parameter space
    - Many methods exist: Gradient based, genetic evolution, etc.

- Gradient decent
    - **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: $C(w_n) \geq : C(w_{n+1})$



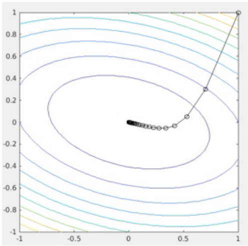
Procedure
1. Define a step length γ
2. Start guess of a position $\nabla C(w_0)$
3. Find gradient
4. Take a step
5. Repeat 2) & 3) $\nabla C(w_1)$
6. Solution: global minima

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

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: $\gamma = 0.1$
  (If the step size is too small $\gamma = 0.0001$ → many steps → do not find a solution)
  (If $\gamma = 0.2$ → few steps → optimal step size)
  (If the step size is too large $\gamma = 0.3$ → unstable → sensitive to local minima's)
  SOLUTION: Dynamic step length

- Max steps: 1000
- Start position: $x_0 = [1,1]^T$



- Can find a solution from any starting point of the space
  $x_0 = [0,-1]^T$
- No local minima's nearby

Iteration 37(end)

- Noisy Data: We cannot find optimum

<u>Quiz:</u> What is the updated position xnew?
Model fitting uses a cost function: C(x)= $x_1^2 + x_1x_2 + 3x_2^2$
and an iterative optimizer: Gradient descent with step length of 0.2.

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

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

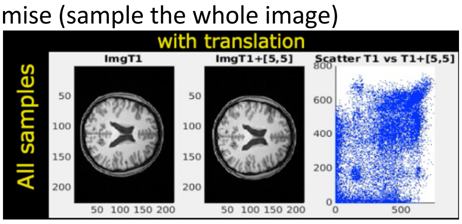
2. Update the step: $x_{new} = x - \nabla C \cdot$stepLength
- $x_{new} = [1,0]^T - 0.2 \cdot [2,1]^T = [0.6, -0.2]^T$

<u>Solution:</u> $[0.6, -0.2]^T$

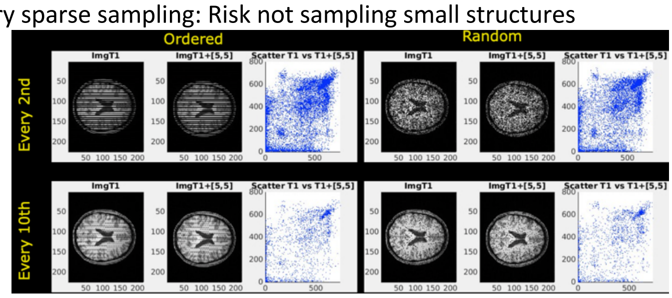
4. The Sampler

How many data points for a robust similarity measure?

- 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
  - Efficient selection of image points
- Sparser sampling: Similar scatter plot
  - Define a good compromise (sample the whole image)



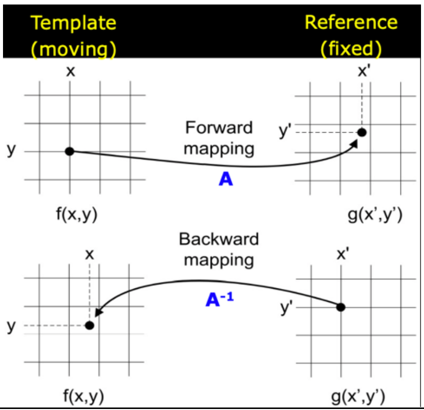
- Ordered vs Random:
  - Spatial dependency: Dependent on large homogeneous structures
  - Very sparse sampling: Risk not sampling small structures



5. Interpolation

Remember: **Forward vs Backward mapping**

- In a nut shell: going backward we need to invers the transformation



Interpolation methods

- Enhances structural boundaries
    - Higher-order interpolation methods: Reduce blurring
- May visually appear "sharper"
    - Do not change image information
    - Only of combining interpolated images with different information of the same object
    - Different angles of moving object e.g. car

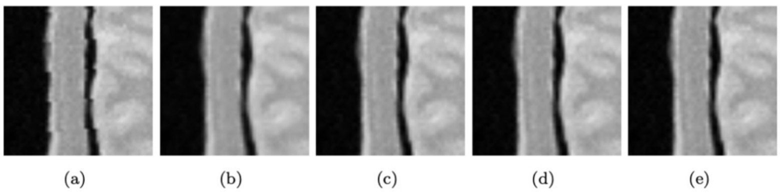
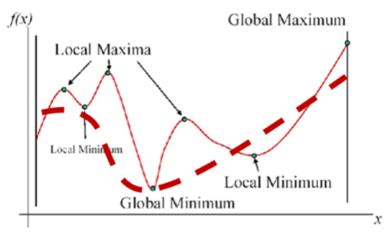


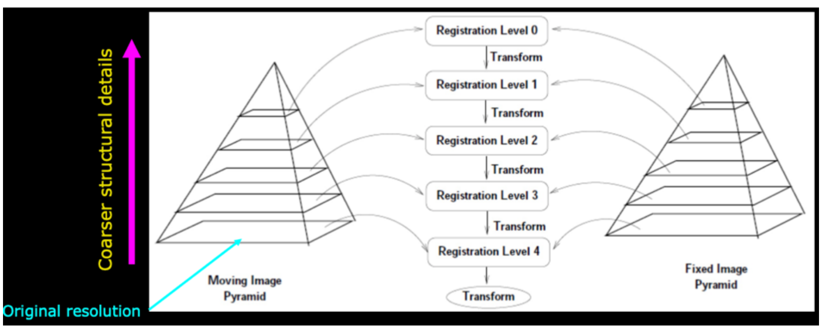
Figure 2.4: Interpolation. (a) nearest neighbour, (b) linear, (c) B-spline $N = 2$, (d) B-spline $N = 3$, (e) B-spline $N = 5$.

6. Pyramid

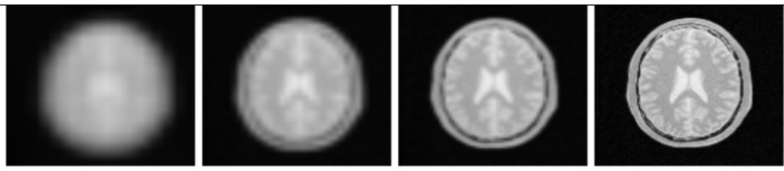
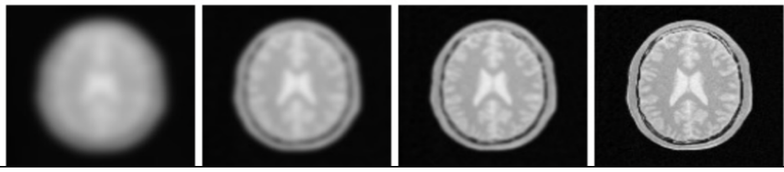


The Pyramid Principle

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

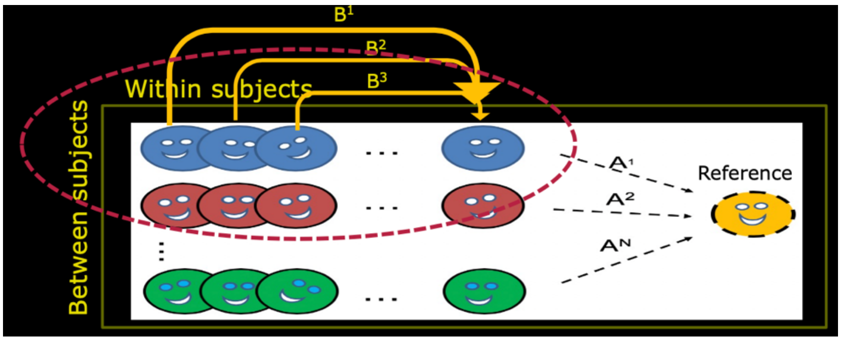


- Lower image resolution
  - Down sampling → memory reduction, fewer data
- Less structural details
  - Smoothing → complex methods settings become more general

Down sampling	 (a) resolution 0 (b) resolution 1 (c) resolution 2 (d) original
Smoothing	

Combining Image Registration pipelines

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



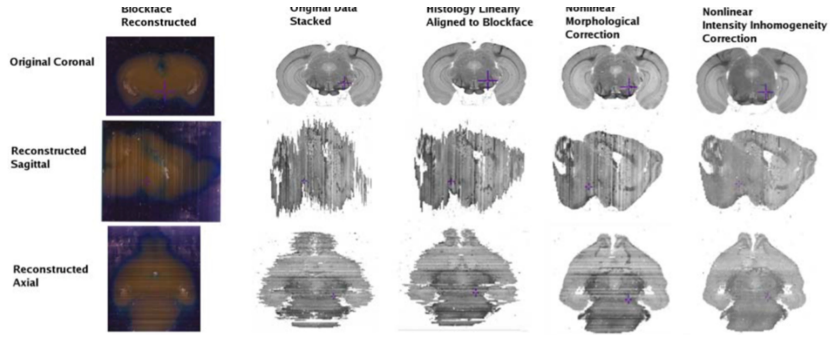
<u>Quiz:</u> Quality inspection- How
How to quality ensure (QA) the image registration results?

- ***Use a similarity measure***
- *Visual Inspection*
- *No need it to – just works*
- *Sum of square difference*
- *Search the internet of experience*

<u>Image Registration pipeline strategy</u>

Within subjects and between challenges

- E.g. Histology 2D → 3D: structural difference between slices
- Visually inspect your results



Within subjects across time points (temporal)

- Remove image distortions + subject motion