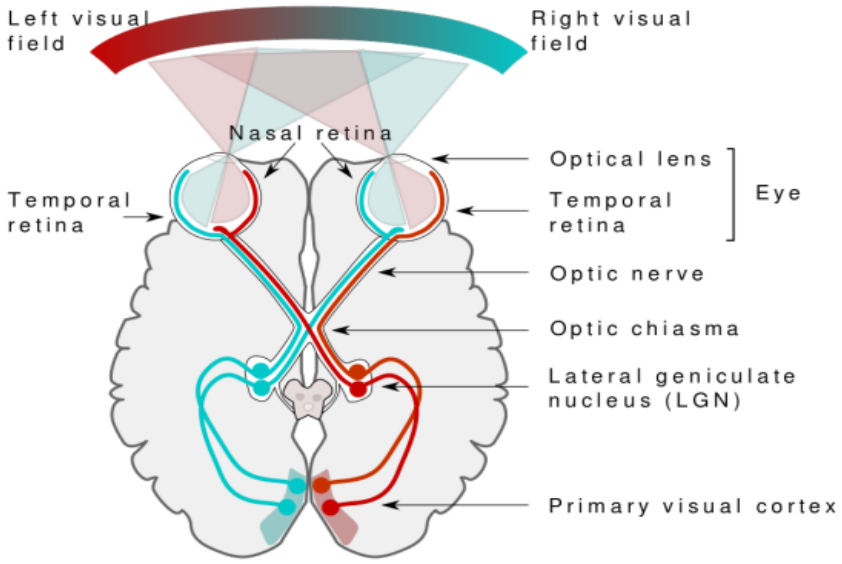
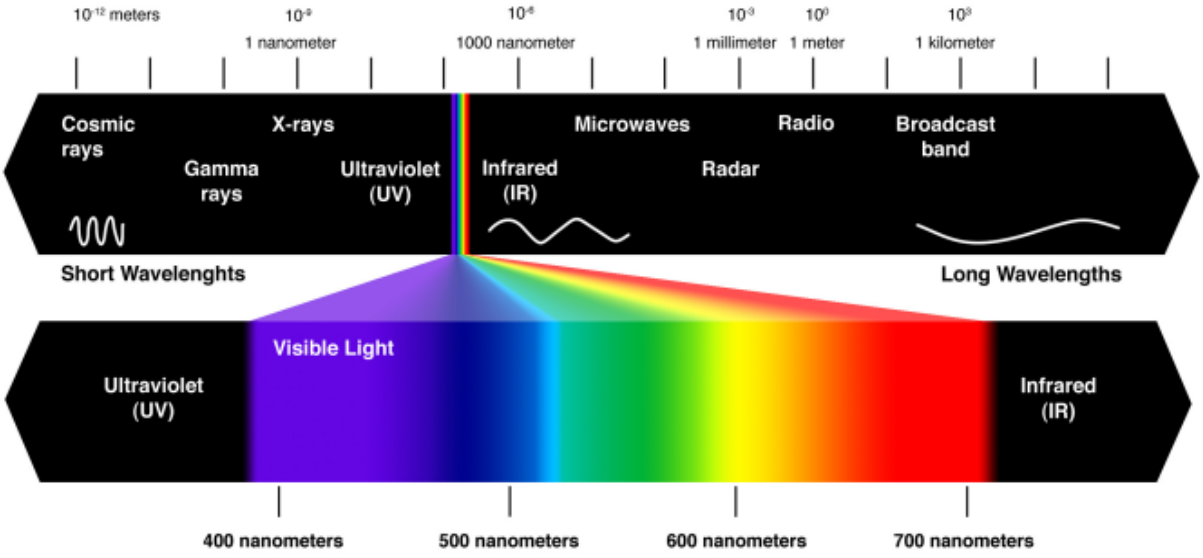
1: Vision

- **Computer vision**: field dealing with capture, analysis and interpretation of images to infer useful information in a sense of understanding the scene.

Human Visual System

- **Sensor**: approx. $130 \cdot 10^6$ receptors in the eye (compare to $3 \cdot 3840 \cdot 2160 = 3 \cdot 8.3 \cdot 10^6$ subpixels in 4k screen).
- **Datalink**: approx. 10^6 nerve fibers in the optical nerve.
- **Processing**: approx. 10^{11} neurons where each is coupled to approx. 10^4 other neurons.
- **Retina** itself already forms a **multi-layer neural network**.
- **Visible spectrum**: part of the spectrum that human eye can see ($400 - 700$ nanometers). Lies between **ultraviolet** and **infrared**.





Digital Images

- **Image**: function: $f : \mathbb{N}^N \to \mathbb{R}$. E.g., with $N = 3 : f(c, i, j) = b$, b pixel **intensity** value.
- **Intensity**: pixel value. During plotting, lowest intensity is mapped to black, highest to white (in black-white). Other **colormaps** include **parula, hot, viridis**. Parula and viridis have a clear distinct between low and high values, which is

why they are popular.
- **Multi-spectral image**: image providing information of different sensors in an aligned coordinate space (e.g., multi-spectral-cameras, as typically used in space exploration, where several sensors capture different wavelength ranges of the optical spectrum, for example spectral intervals in the visible range plus intervals in the near infrared plus intervals in the far infrared). RGB image is multi-spectral image too.
- **Multi-model image**: acquisitions of different MRI sequences of a patients anatomy.

Sampling, Resolution and Quantization

- **Sampling**: same as resolution.
- **Quantization**: range of values for pixels.
- **Shannon-Nyquist theorem**: to accurately reconstruct a continuous signal from its sampled version, the **sampling rate must be at least twice the highest frequency component present in the signal**. This minimum rate is called the Nyquist rate: $f_s \geq 2f_{max}$. E.g., if you are sampling an audio signal with frequency components up to **20 kHz** (the upper limit of human hearing), the Shannon-Nyquist theorem tells us that the sampling rate must be at least **40 kHZ**. For spatial signals like images, the concept is similar. The highest frequency in an image corresponds to how rapidly intensity values change over space. If an image is undersampled, aliasing artifacts like moire patterns can occur.

Pseudo-Color and False-Color

- **Pseudo-color**: e.g., a black-and-white thermal image, where lower temperatures are mapped to blue and higher temperatures to red.
- **False-color**: applied to images where the captured data represents wavelengths outside the visible spectrum. We map values outside of the visible spectrum to values inside the visible spectrum. E.g., a satellite image where infrared data is used to represent vegetation (with green) and urban areas (with red).

Window/Level

- **Quantization**: show only a certain subset of gray values (an interval of the histogram) at a given time using the full intensity range of the display screen to visualize this interval.
- Linear map that maps a subset of values into range $[0, 1]$ via level and window. E.g. CT scan image pixels can take values in range $[-3200, 3200]$. We take a subset of these values, e.g. $[0, 100]$ and map it to range $[0, 1]$. Each subset is known to show different objects, e.g. bones, water, etc.

Histograms

- **Histograms**: describe the intensity value distribution of an image by means of its discrete probability distribution: $H_g(i) = \{(x, y) \in B | b(x, y) = i\} \forall i \in D$ with $B = b(x, y)$ a discrete 2D image and $b \in D$. Absolute histogram is **normalized** to relative histogram by diving by the number of pixels $N \times M$. Also, $C_g(j) = \sum_{i=1}^{j} H_g(i)$ discrete cumulative distribution function. Histograms use **binning** approach (partitioned into intervals of a fixed length) instead of computing the histogram for every possible instensity value of the image.

2: Image Grids and Neighborhood Relations

Cartesian Grids

- Square grid: simple and most common form; regular grid over the image.
- Non-square grid: anisotropic pixles: $\nabla x \neq \nabla y$; frequent in medical imaging.

Non-Cartesian Grids

- Hexagonal grid: each pixel has exactly 6 neighbors; discrete geometry is simpler (distances, contour, length, area of objects, etc.)

Neighborhood (in 2D)

- N_4, 4-neighborhood: the four direct neighbors of a central pixel; up, down, left, right pixels.
- N_8, 8-neighborhood: the four direct neighbors plus the four diagonal neighbors of a central pixel: up, down, left, right and diagonal pixels.
- N_D, D-neighborhood (diagonal): the diagonal neighbors of a central pixel.

Distance Measure

$$d_e(p,q) = \sqrt{(r-x)^2 + (s-y)^2}, \text{ Euclidean}$$

- Neighbors $q \in N_4$ have $d_e = 1$ for central pixel p.
- Diagonal pixels $q \in N_D$ have $d_e = \sqrt{2}$.

Distance Properties

- $D_x(p,q) \geq 0$
- $D_x(p,q) = 0$ if $p = q$
- $D_x(p,q) = D_x(q,p)$
- $D_x(p,z) \leq D_x(p,q) + D_x(q,z)$

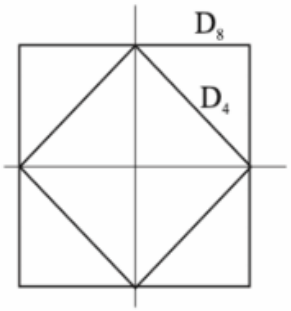
Distance Types

- D_4: Shortest N_4 path between p and q: $d_4(p,q) = |r-x| + |s-y|$
- D_8: Shortest N_8 path between p and q: $d_8(p,q) = \max(|r-x|, |s-y|)$

D_8 and D_4

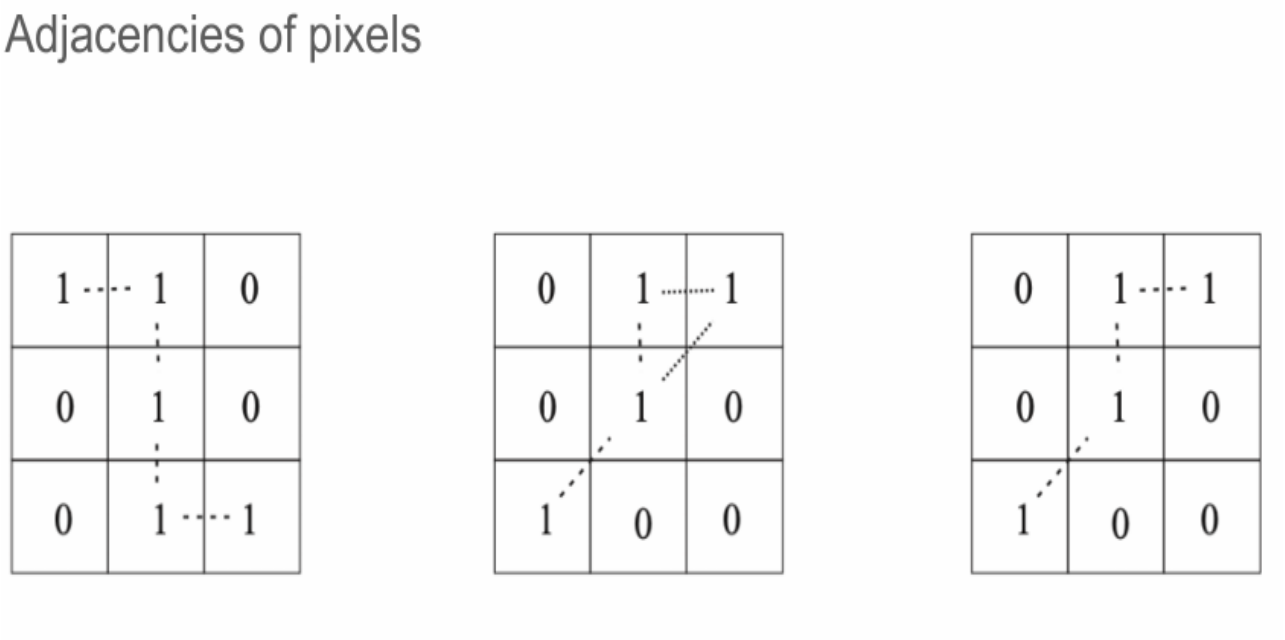
4	3	2	3	4
3	2	1	2	3
2	1	0	1	2
3	2	1	2	3
4	3	2	3	4

2	2	2	2	2
2	1	1	1	2
2	1	0	1	2
2	1	1	1	2
2	2	2	2	2



Adjacency

- 4-adjacency: two pixels p and q with values within V are 4-adjacent if q is in $N_4(p)$.
- 8-adjacency: two pixels p and q with values within V are 8-adjacent if q lies in $N_8(p)$.
- D-adjacency (diagonal): two pixels p and q with values within V are adjacent if q is in $N_D(p)$.
- m-adjacency: two pixels p and q with values within V are adjecent if
  1. q is in $N_4(p)$ or
  2. q is in $N_D(p)$ and the set $N_4(p) \cap N_4(q)$ has no pixels with values from V.
- The m-adjacency avoids inconsistencies (e.g., multiple assignments in the case of 8 adjacency).

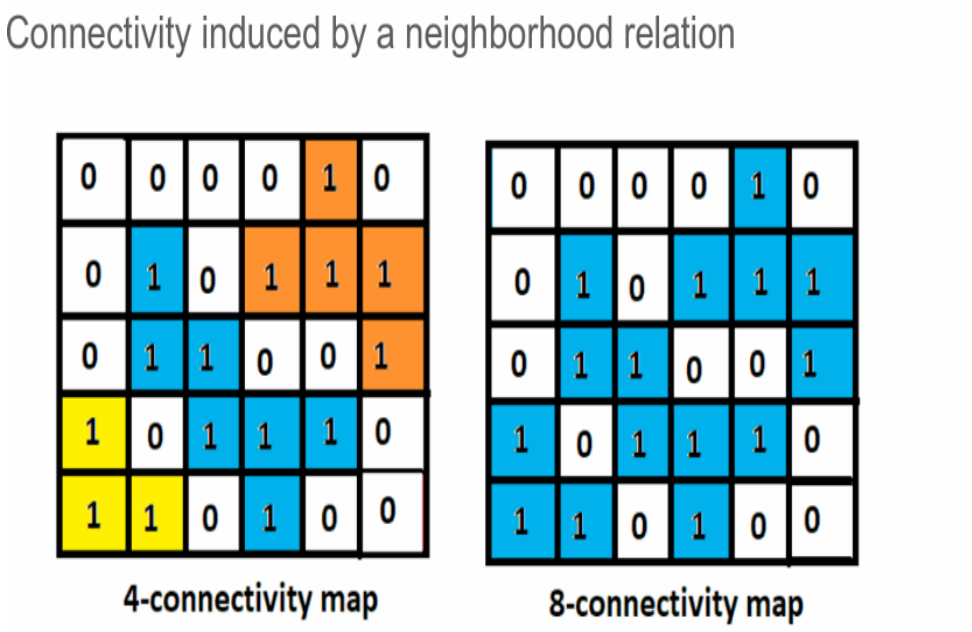


Connected Components

- **Digital path**: $S = \overline{pq}$:
  - $p = (x, y) = (x_0, y_0)$
  - $q = (r, s) = (x_n, y_n)$

$$S = \overline{pq} = [(x_0, y_0), \ldots, (x_n, y_n)]$$

- **Connected component**: let S be a set of pixels of an image, then two pixels p, q are connected in S if there is a path that is entirely in S. For each pixel p in S, the set of all pixels in S that are connected to it is called a connectivity component of S.

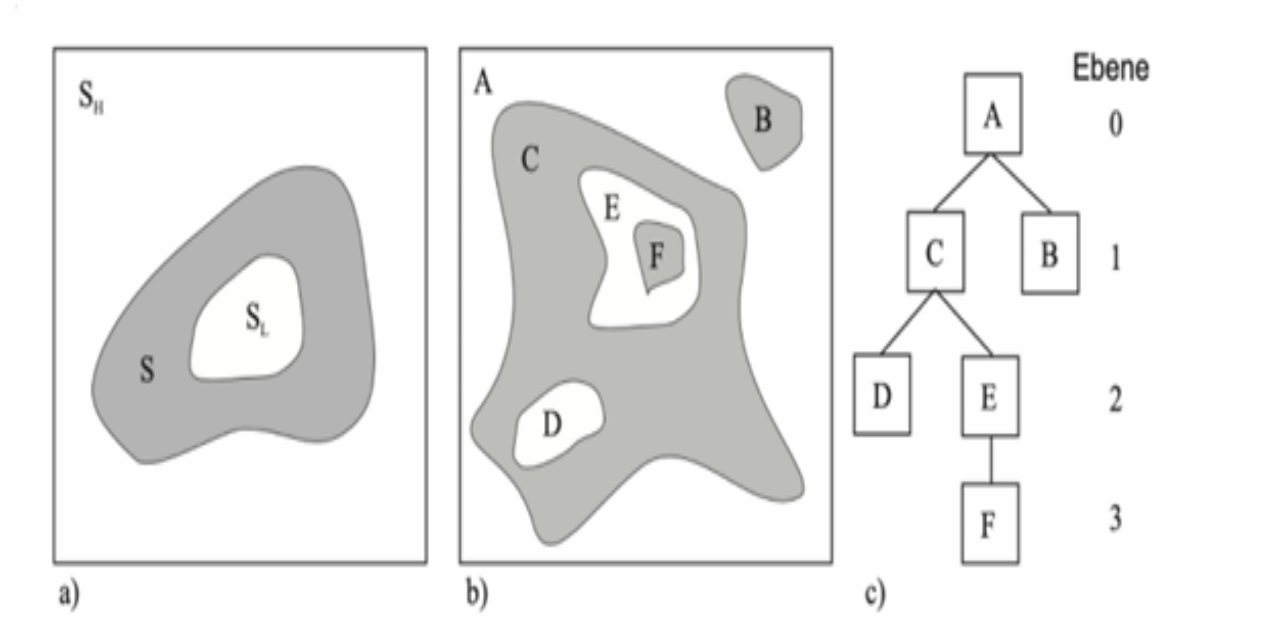


- Connectivity is:
  - Symmetric: $pZq \to qZp$ (Z is an equivalence relation).
  - Transitive: pZq and $qZu \to pZu$
  - If there is only one correlation component (?), then it is called a correlation set.

Adjacency Tree

- Simple description of segmented images:objects and background each represented as connected components.

- An image then contains components:
  - S (object or silhouette)
  - S_H (background, touching edge)
  - S_L (hole in the object)
- Surrounding relation U: the relation U says that the silhouette S surrounds the hole S_L.
- Adjacency tree: the relation U can be described as tree with the following level numbering:
  - Level 0: image B
  - Odd levels: components S_i
  - Even levels: holes S_{L_i}



Euler Number

- **Euler number**: topological property is the Euler number E, which is the difference of the number C of components minus the number H of holes: $E = C - L$.

3: Image Operations

- **Image restoration**: restoration of the original, restoration w.r.t. recording errors, noise, etc.
- **Image enhancement**: restored image signal and serves to emphasize (enhance) certain image contents, such as edges, suppression of local gray value fluctuations, etc. for subsequent evaluation.

Image Operations

- **Point operations**: consider each point of the image range individually. Value range (intensity) mappings in the form $T_p : f(r, s) \to g(x, y)$ with T_p any linear or non-linear transformation $g = T_p(f)$. Typical applications are:
  - **Image enhancement** (subjective).
  - **Segmentation** (binary).
  - **Filtering**.
  - **Inversion**: brightness values are mirrored at the center of the value range: $T_{INV} : g = (L - 1) - f, f \in [0, L - 1]$.
  - **Binarization**: brightness values below θ are set to 1 otherwise 0.
  - **Contrast enhancement**: gray values are stretched or compressed in certain areas. E.g., $f \cdot c_1$ if $f \leq \theta_1$ and $f \cdot c_2$ if $f \leq \theta_2$.
- **Local (neighborhood) operations**: consider the function values $f(q)$ of the points q in the neighborhood of a pixel $p = (x, y)$ to generate a new function value g at the point p' (usually: $p' = p$). We need to define neighborhood $N(p)$ (e.g., N4). $f(N(x, y)) \to g(x, y)$. Typical applications are:
  - **Local image enhancement** (e.g., noise reduction, edge enhancement).
  - **Local filteing**.
  - **Average**: average over neighborhood $N(p)$ as N4 neighbor points to the point p.

  - **Pairwise gray value difference**: good for edge detection. For N4: $g(x,y) = (f(x,y-1) - f(x,y+1)) + (f(x-1,y) - f(x+1,y))$.
- **Geometric operations**: mixture of point and local operations.

Filtering

- **Filter**: **filter mask** $w(r,s)$ **slides over image** $F = f(x,y)$ column by column and row by row. Filter has coefficients $w(r,s)$. For linear filters, filter response $G = g(x,y) = \sum_{\forall(r,s)\in N(x,y)} w(r,s)f(r,s)$.
- The operation is performed **point-by-point**.
- **Boundary pixels problems**: overlapping W with F at boundaries is not possible. Solutions: **padding**, **mirroring** of F at the boundaries.
- **Modified Laplace operator**: used for contour enhancement: $W_{T_L} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$
- **Gauss smoothing**: used for smoothing: $W_{T_G} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$
- **Moving average**: averaging results in blurring. The amount of blurring is proportional to the neighborhood size of the filter. To reduce the effect of blurring, apply threshold: areas with strong changes in gray values should remain unchanged, because edges are expected in the image. With simple averaging, edges are blurred, sharp details are blurred, ringing effects occur.
  - Original averaging: $f(x,y) = \frac{1}{MN}\sum_{(r,s)\in N_{xy}} g(r,s) \forall (x,y) \in G$
  - Thresholded averaging: $f(x,y) = \begin{cases} \frac{1}{n^2}\sum_{(r,s)\in N_{xy}} g(r,s) | g(x,y) - \frac{1}{n^2}\sum_{(r,s)\in N_{xy}} g(r,s) < T, T > 0 \\ g(x,y)| \text{ else} \end{cases}$
- **Median filter**: removes noise, but keeps edges. Good with **sand-pepper** effect. Each pixel is replaced by the median of the pixels in its neighborhood.

Convolutions

- **Convolution**: linear filtering of two functions $f(x,y)$ and $g(x,y)$: $f(x,y) * g(x,y) = \frac{1}{MN}\sum_{r=0}^{M-1}\sum_{s=0}^{N-1} f(r,s)g(x-r,y-s)$
- Corresponds to frequency domain to the point-wise multiplications of the frequency functions $F(u,v)$ and $G(u,v)$.
- **Convolution theorem**: convolution of two functions $f(x), g(x)$ in the spatial domain (time domain) corresponds to a multiplication of their Fourier transforms $F(u), G(u)$ in the frequency domain: $f(x) * g(x) \iff F(u) \cdot G(u)$

Image Enhancement

- **Subjective**: for the human viewer, e.g., gamma correction: $f(x,y) = c \cdot b^{\gamma}(x,y), c, \gamma \in R > 0$
- **Objective**: for machine evaluation, e.g., nonlinear noise suppression: $f(x,y) = \text{MEDIAN}(b(r,s)_{\forall(r,s)\in N(x,y)})$ with $N(x,y) = R_{xy}^{33}$
- **Histogram equalization**: enhance the contrast of an image by redistributing the pixel intensity values so that the histogram (distribution of intensities) becomes more uniform, spreading out the most frequent intensity values. This makes dark areas lighter and bright areas darker, improving overall visibility. $s_k = \text{round}\left(\frac{(L-1)}{N}\sum_{j=0}^{k} h(j)\right)$ with s_k new intensity value for pixel intensity k, L total number of intensity levels (e.g., 256 for 8-bit image), N total number of pixels in the image, $h(j)$ histogram value (number of pixels) for intensity level j, $\sum_{j=0}^{k} h(j)$ CDF up to intensity k.

Inferences

- **Noise**: random variation in data, signals, or images that obscures or distorts the desired information: $g(x,y) = f(x,y) + \nu(x,y), \mathbb{E}[Z_\nu] = 0$. Caused by sensor and transmission system. Solved by **image smoothing**, e.g., average low pass filter.
- **Ramps (in the brightness curve)**: I think its sudden changes in the values. Caused by inhomogeneous illumination. Solved by **local adaptation**, e.g., moving average.
- **Blur**: Caused by aberration, movement. Solved by **image sharpening**.
- **Geometric distortions**: caused by imaging geometry. Solved by **equalization**, e.g., affine mapping.

Image Sharpening

- **Image sharpening**: related to edge detection.
- An **edge** represents a **significant change** in the function values. We can capture this change using **derivative**. The **Euclidean norm** can be used to calculate the **length of the gradient**.

- The gradient operator returns the maximum increase of the function area (gray values) at each point, so that **large values indicate an edge**.
- Partial derivatives can be **approximated** by difference quotients:
  - Forward difference: $B_x(x,y) \approx \frac{B(x+d,y)-B(x,y)}{d}$ and $B_y(x,y) \approx \frac{B(x,y+d)-B(x,y)}{d}$
  - Backward difference: $B_x(x,y) \approx \frac{B(x,y)-B(x-d,y)}{d}$ and $B_y(x,y) \approx \frac{B(x,y)-B(x,y-d)}{d}$
  - Central difference: $B_x(x,y) \approx \frac{B(x+d,y)-B(x-d,y)}{d}$ and $B_y(x,y) \approx \frac{B(x,y+d)-B(x,y-d)}{d}$
- **Discrete gradient operators**:
  - Roberts cross: $\nabla x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \nabla y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$
  - Prewitt: $\nabla x = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}, \nabla y = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$
  - Sobel: $\nabla x = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, \nabla y = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$
  - **Kirsch operator**: determination of the **direction** of the maximum gradient. Kirsch operator defines 8 different directions. The direction ϕ of the maximum gradient: $K_\phi = \max K_{i \forall i=0,\dots,7}$.

Kirsch operator

$K_0 =$

-3	-3	5
-3	0	5
-3	-3	5

$K_1 =$

-3	5	5
-3	0	5
-3	-3	-3

$K_2 =$

5	5	5
-3	0	-3
-3	-3	-3

$K_3 =$

5	5	-3
5	0	-3
-3	-3	-3

$K_4 =$

5	-3	-3
5	0	5
5	-3	-3

$K_5 =$

-3	-3	-3
5	0	-3
5	5	-3

$K_6 =$

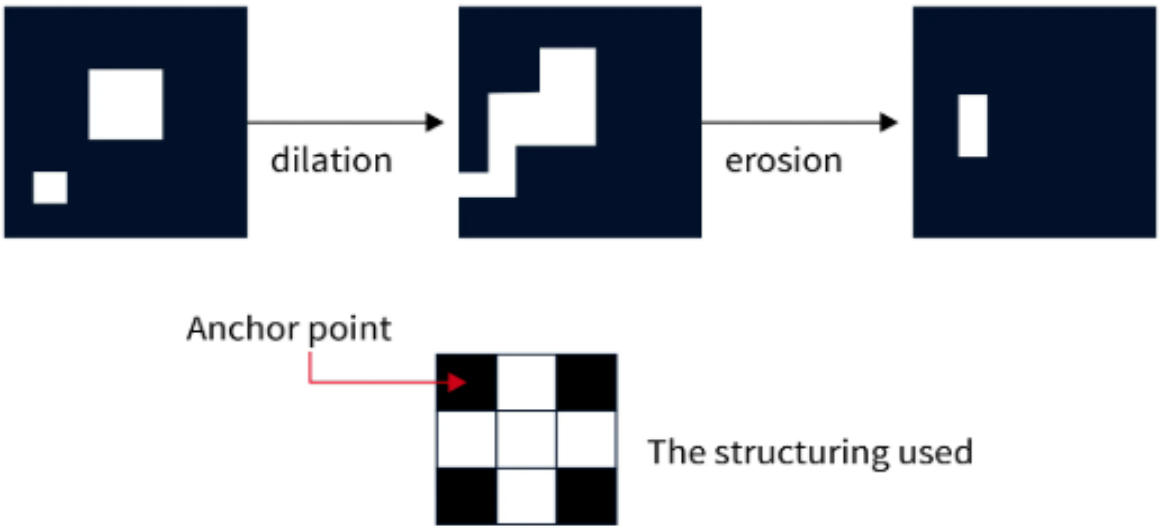
-3	-3	-3
-3	0	-3
5	5	5

$K_7 =$

-3	-3	-3
-3	0	5
-3	5	5

- Gradient operators are sensitive to noise. So, first apply **image smoothing** (median filter), then **compute gradient**.
- Note that relative maxima of the first derivative, which indicate **edge points**, are located at **zero crossings** of the **second derivative**. We use **Laplace** operator: $\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$, which is equivalent to: $\nabla B(x,y) = \nabla^2 B(x,y) = \frac{\partial^2}{\partial x^2} B(x,y) + \frac{\partial^2}{\partial y^2} B(x,y)$
- We can combine filters: first smooth, then detect. **Laplace on Gaussian (LoG)**:
  - $\nabla(g_\sigma * B) = \nabla(g_\sigma) * B$
  - $g_\sigma(x,y) = \frac{1}{2\pi\sigma^2} \exp\left[\frac{-(x^2+y^2)}{2\sigma^2}\right]$ Gaussian function to smooth signal.
  - $\nabla(g_\sigma * B) = 0$ edge criterion, zeros of the Laplace operator.
  - Looks like Mexican hat.

Dilation and Erosion in Morphology



Dilation

- **Dilation** "grows" objects by turning nearby background pixels into foreground pixels if they are close enough to a foreground object.
- **The operation is applied globally to the entire image, considering the neighborhood of each pixel independently of the others.**

$$d(i,j) := \begin{cases} 1 & \text{if } b(i,j) = 0 \wedge \exists (r,s) : (b(r,s) = 1 \wedge X^*(r,s) = 1) \\ b(i,j) & \text{otherwise,} \quad (r,s) \in N_{X^*}(i,j) \end{cases}$$

- $d(i,j)$ represents the dilated value at pixel (i,j).
- If the pixel at (i,j) in the binary image $b(i,j)$ is 0, we check whether any neighboring pixel (r,s) in the neighborhood defined by X^* is 1. If at least one such pixel exists, the value of $d(i,j)$ is set to 1.
- If none of the neighboring pixels is 1, the original value of $b(i,j)$ is kept.
- **Effect**: this operation expands or "grows" object boundaries, turning nearby background pixels (0) into foreground pixels (1) if they are close to a foreground object.

Erosion

- **Erosion** "shrinks" objects by removing foreground pixels that are near the background.
- **The operation is applied globally to the entire image, considering the neighborhood of each pixel independently of the others.**

$$e(i,j) := \begin{cases} 0 & \text{if } b(i,j) = 1 \wedge \exists (r,s) : (b(r,s) = 0 \wedge X(r,s) = 1) \\ b(i,j) & \text{otherwise,} \quad (r,s) \in N_X(i,j) \end{cases}$$

- $e(i,j)$ represents the eroded value at pixel (i,j).
- If the pixel at (i,j) in the binary image $b(i,j)$ is 1, we check whether any neighboring pixel (r,s) in the neighborhood defined by X is 0. If at least one such pixel exists, the value of $e(i,j)$ is set to 0.
- If no neighboring pixel is 0, the value of $b(i,j)$ is preserved.
- **Effect**: this operation shrinks or "erodes" object boundaries, removing foreground pixels (1) that are near the background.

They are often used together in image processing to perform more complex tasks like morphological opening (erosion followed by dilation) and closing (dilation followed by erosion).

Open and Close Operations in Morphology

Opening

- **Opening**: process of **erosion followed by dilation** using a structuring element X.
- Opening removes small objects and noise from the foreground while keeping the larger structures mostly intact. It smooths the contour of objects by thin connections and eliminating small, isolated parts of objects.

$$o = b \circ X = (b \ominus X) \oplus X$$

- $\ominus$ represents erosion, which removes pixels from the boundaries of objects.
- $\oplus$ represents dilation, which adds pixels to the boundaries of objects.
- Opening operation can be applied iteratively i-times to further enhance the effect: $o = ((b \ominus X)_{i-\text{times}} \oplus X)_{i-\text{times}}$

Closing

- **Closing**: process of **dilation followed by erosion** using a structuring element X.
- Closing fills small holes in the foreground objects and connects nearby objects by bridging gaps. It smooths contours by closing up small breaks and long thin gulfs in the objects.

$$s = b \bullet X = (b \oplus X) \ominus X$$

- $\oplus$ represents dilation, which adds pixels to the boundaries of objects.
- $\ominus$ represents erosion, which removes pixels from the boundaries of objects.
- Similar to the opening operation, closing operation can also be applied iteratively i-times for further effect: $s = ((b \oplus X)_{i-\text{times}} \ominus X)_{i-\text{times}}$

Image Segmentation

- **Image segmentation**: divide an image into meaningful sub-areas to be classified as objects or background.
- Object are characterized by **homogenity**.
- **Area-oriented segmentation**: divide image plane into regions that correspond to the objects of interest or background. The segmentation is **complete** because regions cover the whole image.
- **Thresholding**: use multiple thresholds to divide image into multiple segments (comparing its gray value to a threshold). Note that sometimes a small change in the threshold or input image can have a significant effect in the result.
- **Superposition**: overlaying something on an image. This introduces a gradient or varying intensity in the background, which alters the overall pixel intensity distribution.
- **Automatic threshold determination**: one can use a statistically motivated approach (Otsu method). With a bimodal histogram, one determines values that maximizes the interclass variance.
- **Region growing**:
  1. Start from one image point of a homogeneous region.
  2. Successively add neighbor pixels if such a pixel fulfills a homogeneity criterion.
  3. Process ends when all image points have been assigned to one region.

Texture

- **Texture**: periodicity of basic patterns or stochastic elements.
- Texture is a local property in the image.

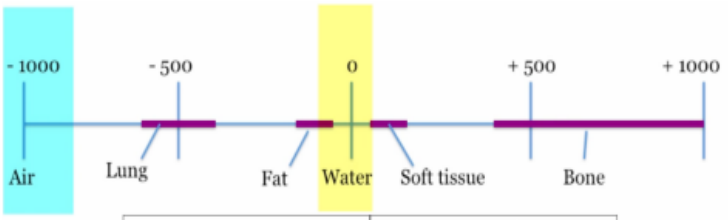
4: Medical Imaging

X-ray

X-rays: electromagnetic wave of high energy and very short wavelength, which is able to pass through many materials opaque to light. For example, there are 5 densities:

1. Air (darkest)
2. Fat
3. Fluid and soft tissue
4. Bone
5. Metal (whitest)

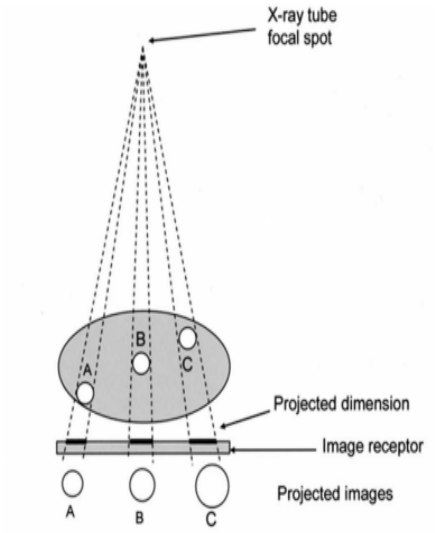
Hounsfield units

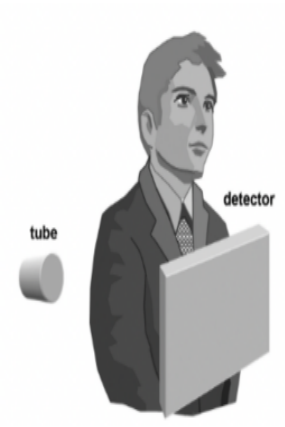


Bone	+400 -------- +1000
Soft tissue	+40 ---------- +80
Water	0
Fat	-60 ---------- - 100
Lung	-400 -------- -600
Air	- 1000

Hounsfield unit = a measure of radiodensity

X-ray radiography





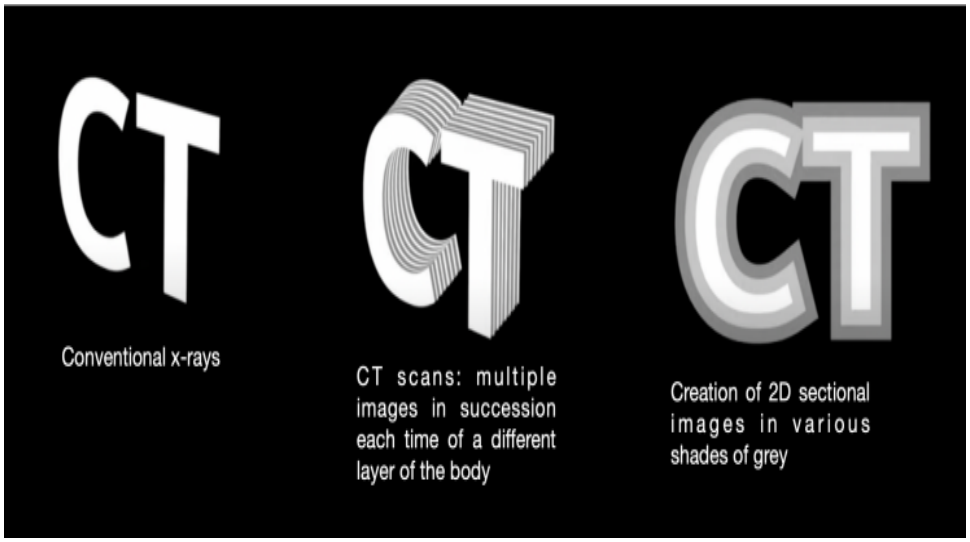
There are directions to X-ray beams. For example, through back to front, i.e. **PA (Posterior-Anterior)**. If lungs are the object of interest, natually the image will have:

1. Poor low-contrast resolution
2. Bone & tissue superimposed
3. Reduced visibility of the object of interest

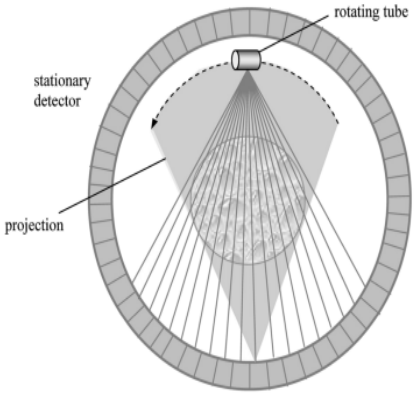
Tomography

Tomography: imaging by sections or sectioning that uses any kind of penetrating wave. Conventional X-ray tomography (limited clinical application):

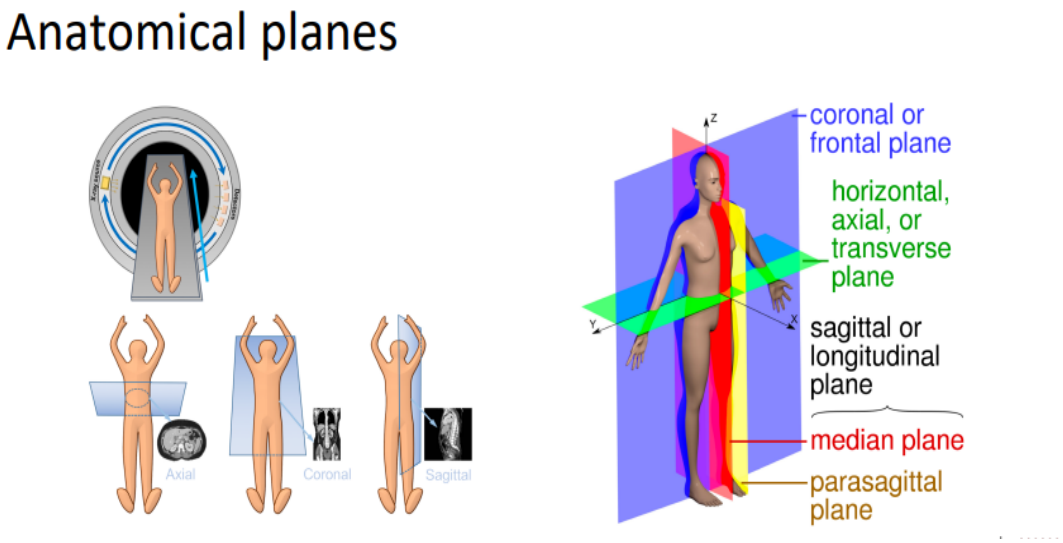
1. Planes close to the focal plane undergo litte blurring.
2. Conventional tomography blurs the overlying structure.
3. Contrast between different structures inside the focal plane is not enhanced.
4. The blurred overlying structures superimposed on the tomographic image significantly degrade the quality of the image.
5. Large x-ray dose to the patient.



4th generation



Anatomical Planes



MRI

- Our body: hydrogen atoms. 1 proton and 1 electron. The proton rotates around its axis.
- Patient placed in strong uniform magnetic field aligning positively charged hydrogen ions (hydrons) or protons.
- Impose a temporary radiofrequency pulse to scatter hydrogen protons: creation of electric charge.

- By applying magnetic pulses, the tissue properly dependent response of the previously aligned protons can be measured by a detector.
- Time from excitation to detection (TE).
- Time before pulses (TR).

Ultrasound

- Sound waves.
- Different transducers (change shape with electrical impulse: sound waves (2-20MHz)).
- Waves travel differently in the body: transmitted, reflected, in between.

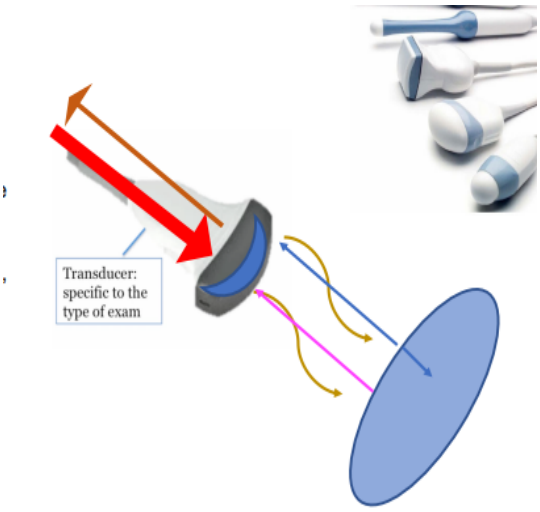


Image File Types

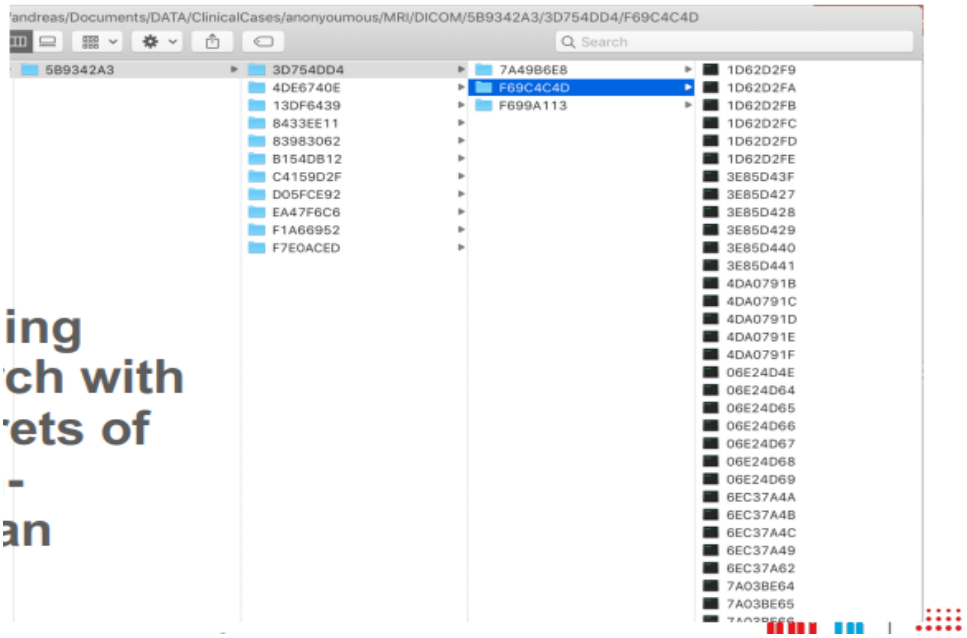
- BMP: 8 bit RGB (3 x 8 = 24 bit per pixel) uncompressed.
- JPEG: 8 bit RGB (3 x 8 = 24 bit per pixel, with lossy compression).
- TIFF: 8 bit or 16 bit (!) RGB (3x8 or 3x16=48 bit per pixel), lossless compression possible, meta data in header possible, multiple image versions per file possible.
- PNG: often 8 bit RGBA , lossless compressed, 4x8=32 bit per pixel.
- GIF: 256 colors using 8 bit colormap (8 bit per pixel + colormap in header)

These formats are not suitable for medical imaging:

1. Limited range.
2. Made for 2D data.
3. No encoding geometry.
4. Encoding of metadata is difficult.

Medical imaging formats:

1. DICOM (.dcm)
    1. “Everbody” can define own “tags” an add specific metainformation -> enables product innovations (e.g. a new scanner type that can encode things not know at the time the DICOM standard was defined)
    2. 3D imaging normally stored as “series” of 2D images, joined via header information (SeriesID tag).
    3. Extremely annoying format for research with potentially hundreds of files for one high-resolution 3D scan.



2. NiFTI (.niii / .nii.gz)
    1. One file per scan.
    2. Able to store high dimensional data (3D, 4D, ...) in one file (think of arrays)
    3. Stores up to two transformation matricies in the header to encode image geometry, called *voxel2world*:
        1. An affine matrix (homogenous coordinates): defines the transformation from voxel coordinates (the grid of 3D pixels representing the image data) to world coordinates (real-world space, such as the patient's anatomy or physical space). It might define:
            1. **Translation**: moving the entire image by a fixed amount in the x, y, and z directions.
            2. **Rotation**: rotating the image around one or more axes.
            3. **Scaling**: changing the size of the image along different axes (stretching or shrinking).
            4. **Shearing**: distorting the image in such a way that axes are not perpendicular anymore.
        2. A unit quaterion representation (3 imaginary numbers).
    4. Does not store metadata - ensures a lot of data protection by design.

More on affine matrix: in NiFTI, the affine matrix uses homogeneous coordinates, which means it is a 4×4 matrix that allows for these transformations to be applied efficiently. The matrix operates on 3D voxel coordinates (x_v, y_v, z_v) and transforms them into 3D world coordinates (x_w, y_w, z_w) with a formula like:

$$\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & t_x \\ a_{21} & a_{22} & a_{23} & t_y \\ a_{31} & a_{32} & a_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_v \\ y_v \\ z_v \\ 1 \end{bmatrix}$$

- The top left 3x3 submatrix represents the combination of **rotation**, **scaling** and **shearing**.
- The rightmost column $\begin{bmatrix} t_x & t_y & t_z \end{bmatrix}^T$ represents translation (shifting the image).
- The bottom row $\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T$ is part of the homogeneous coordinate system, allowing transformations to work on 3D points.

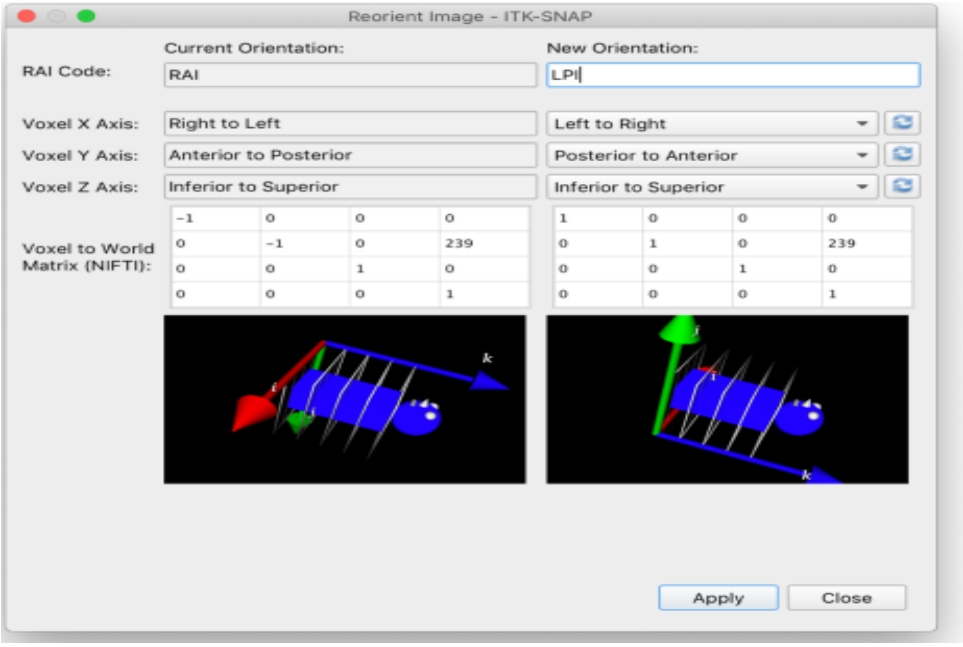
Remember:

- Affine: $x' = Ax + b$, i.e. adds translation.
- Linear: $x' = Ax$

Ordering Information in Images

There are 24 different ways to order the image information in a 3D image. Most common are **LPI, RAS, RAI**. We will use a - to indicate the first form and + to indicate the second, i.e. LPI- means:

- Left → Right
- Posterior → Anterior
- Inferior → Superior



Voxel2World Matrix

- Many libraries to load nifti data apply the *voxel2world* matrix stored in the nifty header at loading time to get a well defined image in mm scale.
- Ideally you then always get a defined orientation in your software, regardless how the data was stored in the file. Often LPI-.
- Unfortunately, the matrix might be wrong.
- Orientation mishaps might not be easily noticed (e.g., similar-looking structures, flipping an image left-to-right or top-to-bottom might not be apparent if anatomical landmarks are not clear).

5: Image Registration

Registration

- **Registration**: finding a geometric transformation that best aligns an image (or an object in space) with an other image (or object in space) (informal). Example: f(tilted triangle) = perfect triangle. Whats $f()$?

- (Image-)registration is mostly only a part of a greater pipeline.

Types of Registration

1. **Dimensionality**: e.g., aligning a 2D CT scan slice with a corresponding 3D MRI volume to ensure consistency between the different dimensions of the images.
2. **Registration basis**:
   - Point-based: registering two X-ray images by matching predefined anatomical landmarks, such as bones or joints.
   - Surface-based: registering two MRI brain scans by aligning the outer surfaces of the brain structures.
   - Intensity-based: aligning two CT scans by matching their intensity values (pixel/voxel intensities).
3. **Geometric transformation**: type of mathematical mapping, number of degrees of freedom, object searched for. Example: using an affine transformation to register two images of the same organ, which allows for scaling, rotation, and translation, accounting for minor positional differences.
4. **Interaction**:
   - Manual: a radiologist manually adjusts the position of a brain scan to align it with a reference image.
   - Semi-automatic: the software suggests an alignment of two heart images, but the user fine-tunes it.
   - Interactive: the user interacts with the image registration software to provide feedback, refining the alignment in real-time.
   - Automatic: a system automatically registers a pre-surgical MRI scan with an intra-operative ultrasound image without user intervention.
5. **Optimization method**: type of geometric transformation determination. Example: using gradient descent to find the optimal transformation that minimizes the difference between two MRI images of the same brain structure.
6. **Modalities**: registration of two same modalities (intra-modal/mono-modal) usually different from registration of two different modalities (inter-modal/multi-modal).
   - Intra-modal: e.g., registering two CT scans of the same organ taken at different times.
   - Inter-modal: e.g., aligning an MRI image with a PET scan to combine structural and functional information.
7. **Subject**: registration of two images of the same patient (intra-patient), two different patients (inter-patient) or with one atlas.
   - Intra-patient: e.g., registering a CT scan taken before surgery with one taken during surgery for the same patient.
   - Inter-patient: e.g., aligning brain scans from two different patients to compare anatomical differences.
   - Atlas: e.g., aligning a patient's brain MRI to a standard brain atlas for anatomical comparison.
8. **Object**: anatomical area being registered, e.g., head, spine or heart. Example: registering two cardiac MRIs to track changes in heart structure over time.

Geometric Transformations

A transformation T is a mapping that assigns each point x of point cloud X a transformed point x':

$$x' = T(x), x \in X$$

- Geometric transform don't transform image intensities.
- Geometric transforms transform (x, y) pairs, i.e. grid.
- We "interpolate" new intensity values on the new grid to obtain new image.

Rigid Transformations Rigid transformation example:

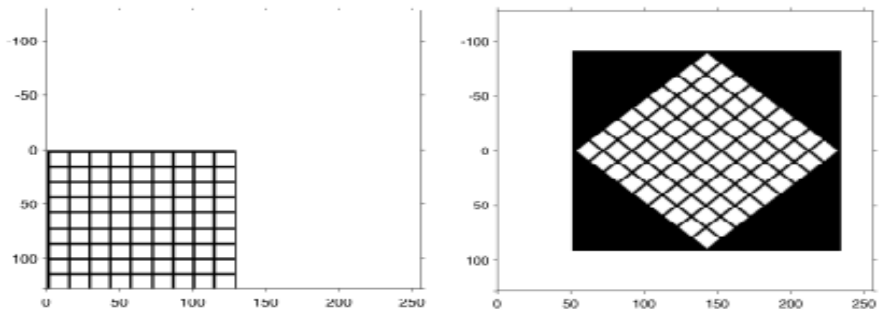


Figure 4: a) Untransformed lattice. (b) Grid after rigid transformation (rotation by angle $-\frac{pi}{4}$ and subsequent translation by $(50, 0)^T$).

Rigid Transformations 2D Parameters:

- Rotation angle ϕ
- Translation vector $t = (t_x, t_y)^T$
- Total 3 degrees of freedom (parameters)

$$x' = T(x) = R_\phi x + t, R_\phi = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}$$

$$x' = T(x) = R^h_\phi \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}, R^h_\phi = \begin{bmatrix} \cos\phi & -\sin\phi & t_x \\ \sin\phi & \cos\phi & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Properties:

- Distances between all points from X correspond to distances between all points from X' (distances are preserved).
- Straight lines remain straight lines.
- Angles between two straight lines are preserved.

Rigid Transformations 3D Rigid transformations in 3D: Parameters:

- Rotation angle ϕ_x, ϕ_y, ϕ_z (Euler angles)
- Translation vector $t = (t_x, t_y, t_z)^T$
- Total 6 degrees of freedom (parameters)

$$x' = T(x) = Rx + t, R = R_z R_y R_x$$

$$R_z = \begin{bmatrix} \cos\phi_z & -\sin\phi_z & 0 \\ \sin\phi_z & \cos\phi_z & 0 \\ 0 & 0 & 1 \end{bmatrix}, R_y = \begin{bmatrix} \cos\phi_y & 0 & \sin\phi_y \\ 0 & 1 & 0 \\ -\sin\phi_y & 0 & \cos\phi_z \end{bmatrix}, R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi_x & -\sin\phi_x \\ 0 & \sin\phi_x & \cos\phi_x \end{bmatrix}$$

Remarks:

- Here the rotation is first around the x-axis, then around the y-axis and finally around the z-axis.
- The use of homogeneous coordinates is also possible in 3D.
- Homogeneous coordinates only allow for a more compact notation. Although they do not formally bring any further advantages, they are often used in practice since 4×4 matrix multiplications are natively supported by GPUs.
- Can also use Quaternions to apply rotation:
  - Extension of the complex numbers in the form $a + bi + cj + dk$
  - One-time rotation around a rotation axis w with angle θ_w.

Linear Transformations Linear transformations are all transformations, that can be computed using a Matrix multiplication (non-homogeneous transformation matrix).

$$x' = T(x) = Ax$$

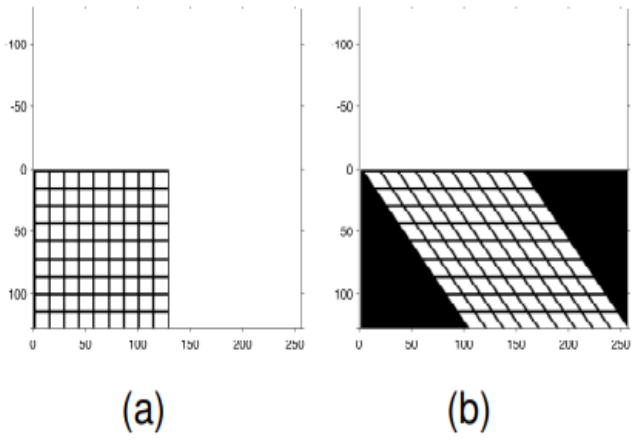


Figure 5: (a) Original grid (b) Grid after linear transformation.

Properties:

- A linear transformation does not include translation.
- Linear transformations feature 4 degrees of freedom in 2D and 9 degrees of freedom in 3D.

Affine Transformations Affine transformation also involves a translation:

$$x' = T(x) = Ax + t$$

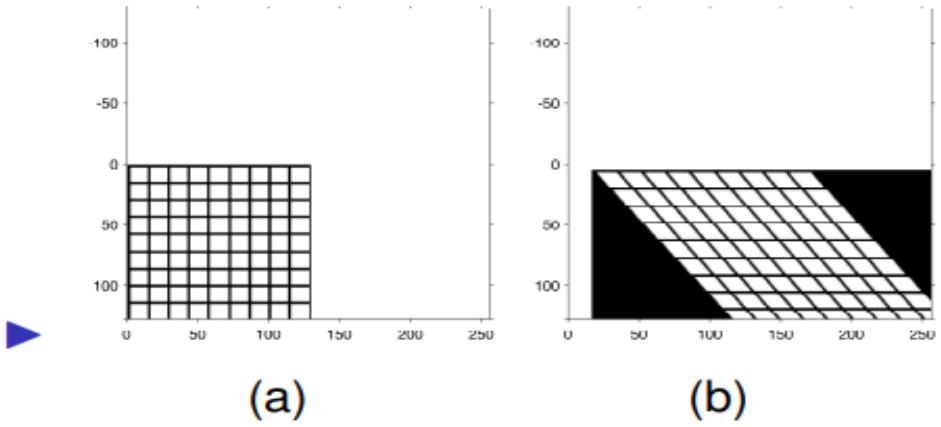


Figure 6: (a) Original grid. (b) Grid after affine transformation.

Properties:

- Distances are not necessarily preserved (due to scaling and shearing).
- Straight lines remain straight lines.
- Parallelism is preserved, but angles between non-parallel lines may change.

Projective Transformations In case of projective transformations the parallelism of straight lines is no longer preserved:

$$x' = T(x) = \frac{Ax + t}{< p, x > + \alpha}$$

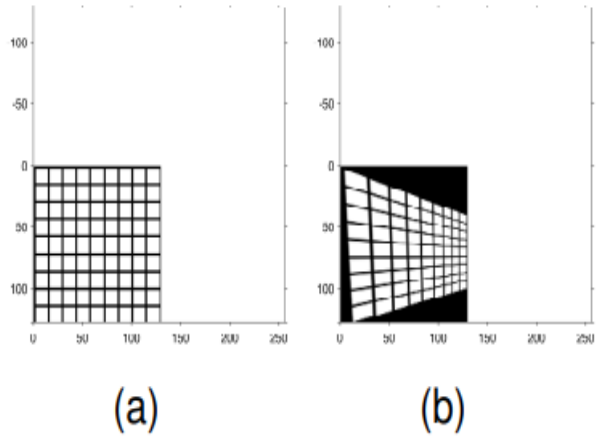


Figure 7: (a) Original grid. (b) Grid after projective transformation.

Properties:

- Projective transformations can be expressed in 3D with fully populated 4×4 matrix in homogeneous coordinates (16 degrees of freedom).
- Perspective transformations are a subset of projective transformations:
  - Projection of spatial data onto a plane (3D to 2D).
  - In photography, X-rays, displaying 3D scenes on a 2D monitor.

Curved/Deformable transformations Curved/deformable transformations allow transformations where straight lines are not preserved.

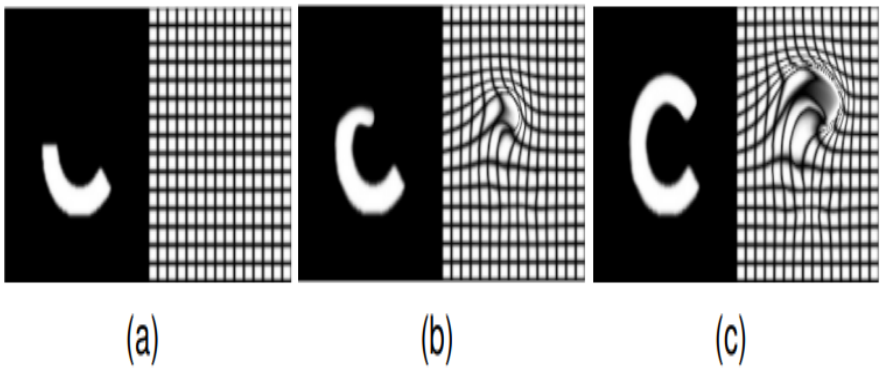


Figure 8: (a) Untransformed image with untransformed grid. (b) Partially transformed image with partially transformed grid. (C) Fully transformed image with fully transformed grid. Figures from [1].

Properties:

- Diffeomorphic transformations T are deformable transformations which satisfy:

  - Mapping T is bijective (total, injective, surjective), i.e. there is a complete list of pairs between X and $X' = T(X)$ and inverse mapping exists.
  - T and the inverse T^{-1} are continiously differentiable (i.e. both are smooth with no jumps).
- A simple deformable transformation can be modeled as a polynomial and using B-Spline-segments.
- In practise deformable transformations are non-analytic description of the transformation using warp fields.

Warp Field

- A warp field W defines for each point $x \in X$ a displacement vector d.
- For a 3D image $f(x, y, z) : N^3 \to N$ the warp field is a mapping $W(x, y, z) : N^3 \to R^3$:

$$x' = T(x) = x + W(x)$$

- We can consider a warp itself as a vector valued image.

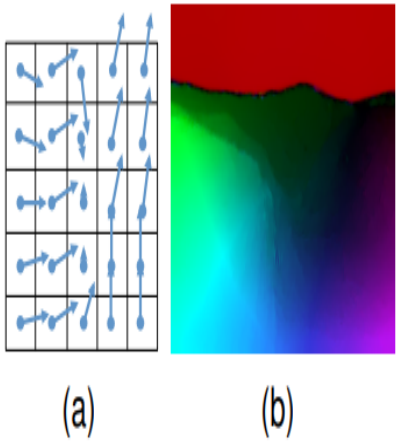


Figure 9: (a) 2D warp field for an image of size 5 x 5. (b) Example of warp field represented as RGB image from ([3]).

Overview

- Comparing the complexity of the transformation types (in terms of degrees of freedom):
  - rigid ∈ linear ∈ affine ∈ projective ∈ deformable

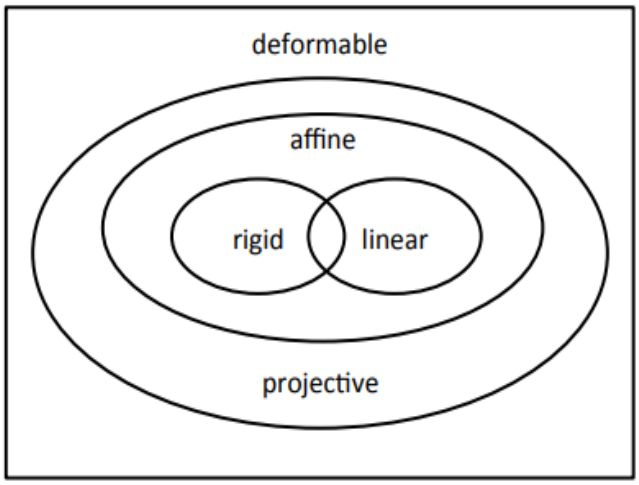


Figure 10: Venn Diagramm of discussed transformation types.

Point-based Registration

Point-based/landmark-based registration: Alignment of two point clouds with each other.

- Inputs: point-clouds X and X':
  - Anatomical landmarks.
  - External markers (fiducials), e.g. screws.
- Unknown: transformation T. The goal is to find a transformation T that best aligns the two point clouds. This transformation can include translations (shifting), rotations, scaling, or other operations that move one set of points to match the other.
- Variants:
  - Alignment of two point clouds of the same size with homologous landmarks (exact assignment of a landmark from X and X'). In this variant, both point clouds X and X' have the same number of points, and there is a direct correspondence between the points. For instance, the first point in X corresponds to the first point in X', the second point in X corresponds to the second point in X', and so on.
  - Alignment of two point clouds with different number of points. There may not be a one-to-one correspondence between points in the two sets, so the registration process must find an optimal way to match as many corresponding points as possible or approximate a transformation based on the points that do match.

The general purpose of point-based registration is to find the transformation T that minimizes the distance between corresponding points in the two point clouds, ensuring that they align as closely as possible.

$$\forall i \in \{1, ..., N\}: \quad T\mathbf{x}_i = \mathbf{x}'_i \tag{21}$$

$$\text{mit } T = \begin{pmatrix} T_{1,1} & T_{1,2} & T_{1,3} \\ T_{2,1} & T_{2,2} & T_{2,3} \\ T_{3,1} & T_{3,2} & T_{3,3} \end{pmatrix}. \tag{22}$$

The N LESs in Eq. 21 can be combined into a single system as follows:

$$\underbrace{\begin{pmatrix} T_{1,1} & T_{1,2} & T_{1,3} \\ T_{2,1} & T_{2,2} & T_{2,3} \\ T_{3,1} & T_{3,2} & T_{3,3} \end{pmatrix}}_{?} \underbrace{\begin{pmatrix} x_{1,1} & x_{2,1} & \dots & x_{N,1} \\ x_{1,2} & x_{2,2} & \dots & x_{N,2} \\ x_{1,3} & x_{2,3} & \dots & x_{N,3} \end{pmatrix}}_{\text{bekannte Punkte X}} = \underbrace{\begin{pmatrix} x'_{1,1} & x'_{2,1} & \dots & x'_{N,1} \\ x'_{1,2} & x'_{2,2} & \dots & x'_{N,2} \\ x'_{1,3} & x'_{2,3} & \dots & x'_{N,3} \end{pmatrix}}_{\text{bekannte Punkte X'}} \tag{23}$$

- Theoretically, for affine transformations, 3 homologous points in 3D are sufficient to define a solution.
- Practically, however, it is useful to use more points so that errors in the definition of the landmarks/setting of the markers have less influence on the result. Using more points allows the registration process to distribute small errors across all the points, leading to a more accurate and stable solution.

Optimization

Linear Squared error $e(T)$ of point-based registration for N homologous point pairs:

$$e(T) = \sum_{i=1}^{N} |x_i - T(x_i)|^2 = \sum_{i=1}^{N} |x_i - x'_i|^2 = \sum_{i=1}^{N} (D_e(x_i, x'_i))^2 = \sum_{i=1}^{N} \sum_{d=1}^{D} (x_{i,d} - x'_{i,d})^2$$

- D: dimensionality of the points ($D = 2$, $D = 3$)
- e: function of T
- X and X': constant parameters.

Objective:

- Find T that minimizes e.

- Solution depends on the type of T (rigid, linear, affine, projective, deformable).

$$T^* = \text{argmin}_T e(T) = \text{argmin}_T \sum_i^N |x_i - T(x_i)|^2$$

- Find solution using derivatives of the error function $e(T)$:
  - At $e'(T) = 0$ we have a minimum, maximum, saddle point.
  - $e''(T) > 0$
  - Instead of first derivative, the gradient is set equal to 0: $\nabla e = 0$
  - Instead of second derivative, check if Hessian matrix is positive definite (positive eigenvalues).
- Find solution using linear algebra:
  - T can be determined by solving N linear systems of equations.
  - $TX = X' \iff X^T T^T = (X')^T$
  - For each $j \in \{1, 2, 3, \dots\}$ there is exactly one system of equations to solve (the j-th column of T is determined in each case).
  - For $N \leq 3$ the LES is underdetermined and infinitely many solutions exist (at least 3 points needed)
  - For $N = 3$ no or exactly one solution exists: $Ay = B \iff y = A^{-1}B$
  - For $N > 3$ the LGS is overdetermined and no exact solution exists $\rightarrow$ least-squares approach to error minimization.
  - If we use 3 basis vectors to approximate vector b that lies in a higher dimensional space $N > 3$, the hyperplane spanned by the 3 basis vectors can only capture points that lie within this lower-dimensional plane.
  - Like in PCA, when you use only 3 principal components to represent data in a higher-dimensional space (e.g., an N-dimensional space), you're projecting the data onto the 3-dimensional subspace. This means that you lose some information because the points that originally lay outside the 3D subspace cannot be exactly represented by the selected 3 principal components.
- Least-squares approach with pseudoinverse, since A^{-1} does not exist: $Ay = B \iff y = A^+B \iff y = (A^TA)^{-1}A^TB$

Affine

- Above techniques cannot be used for other transformation types, since additional requirements are imposed here. For example:

$$T^* = \text{argmin}_T e(T) = \text{argmin}_T \sum_i^N |x_i - (T(x_i) + t)|^2$$

Rigid

- Same for rigid, but with T rotation matrix only (columns are orthonormal and $det(T) = 1$)
- Can be solved by SVD (generalization of eigenvalue decomposition for non-square matrices).

Other

- There are also weighted variants:

$$T^* = \text{argmin}_T e(T) = \text{argmin}_T \sum_i^N w_i |x_i - (T(x_i) + t)|^2$$

- Different weights along the x, y, z axis:

$$T^* = \text{argmin}_T e(T) = \text{argmin}_T \sum_i^N |W_i(x_i - (T(x_i) + t))|^2$$

Surface Based Registration

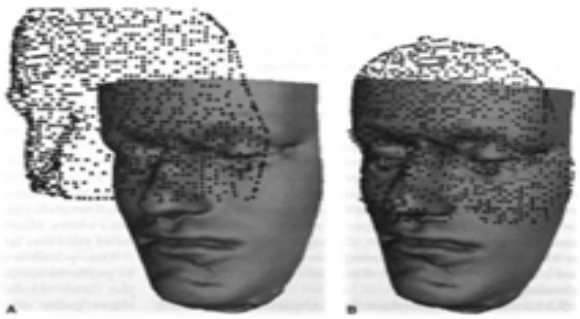


Figure 11: Figure from [5]

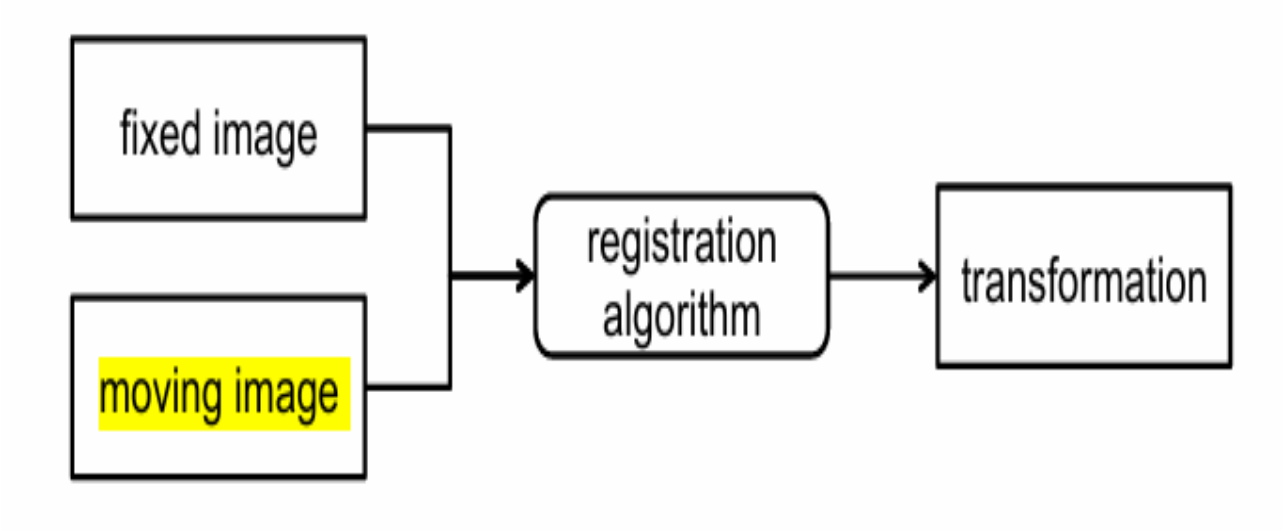
- Usually, no homologous points can be identified directly, i.e. alignment of two point clouds without known homologous points.
- General idea: transform into point-based registration problem by determining points on the surface.
- Typically, a surface point cloud X' derived from an image dataset and a "sampled" pointcloud X with $|X'| >> |X|$, e.g. 20000 and 200.

ICP-Algorithm

You are given X', a dense, detailed point cloud, possibly derived from a 3D surface or image dataset, containing many points, and X, a "sampled" point cloud with fewer points, representing a simpler or less detailed version of the same object. The goal is to align these two point clouds (which have different numbers of points and no direct correspondence between the points) using the Iterative Closest Point (ICP) algorithm.

1. Assign Closest Points
   - For each point in X, find the closest point in X'.
   - Create a temporary correspondence between the points in X and the closest points in X'.
2. Apply Point-Based Registration
   - Perform point-based registration for the corresponding points found in Step 1.
   - Compute the transformation (translation, rotation, etc.) that minimizes the distance between X and X'.
3. Repeat Until Convergence
   - Repeat Steps 1 and 2 until convergence is reached:
     - Recompute the closest points in X' for each point in X.
     - Apply the updated transformation.
     - Stop when the transformation between iterations becomes insignificant, indicating the point clouds are aligned.

Image Based Registration



- In image-based/voxel-based registration, image intensities are compared rather than points.
- If matching points can be determined in two images, the images can be efficiently registered using the methods discussed so far.
- The goal of image-based registration is to align two images as best as possible.
- For this purpose, the image intensities are compared with each other.
- Input:
  - $I_{\text{fixed}}(x)$: the static reference image is called **fixed image**.
  - $I_{\text{moving}}(x)$: the image which is transformed to the fixed image is called **moving image**.
- Output:
  - Transformation T mapping all voxel coordinates x from I_{moving} to voxel coordinates x' in I_{fixed}.
- E.g. used when all images must be in the same coordinate system, e.g. MRI ane CT images of the same patient.

▶ Here: Alignment of a patient image (T1 MRI) with a brain atlas (T1 MRI).

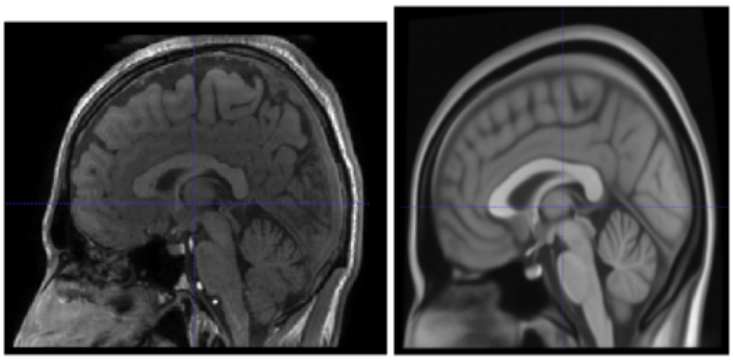


Figure 13: (a) T1-MRI of a patient. (b) T1-MNI152 brain atlas.

Algorithm

1. Input: I_{fixed}, I_{moving}
2. Define transformation T_p
3. Transform I_{moving} using T_p, get I'_{moving}
4. Interpolate I'_{moving}: when the moving image is transformed, its pixels or voxels might not align perfectly with the grid of the original image. For example, if the image is rotated or scaled, some pixels may end up at non-integer positions in the new coordinate system. Interpolation is used to compute the intensity values at these new positions by using the

values of neighboring pixels in the original image (nearest neighbor interpolation, bilinear interpolation).
5. Compute metric using I_{fixed}, I'_{moving}
6. Optimize T_p
7. Repeat from 3 until convergence.

Interpolation

- Nearest-Neighbor Interpolation: The intensity value in the closest voxel is used: $x' = \text{round}(T(x))$

$$\mathbf{x}' = \text{round}(T(\mathbf{x})).$$

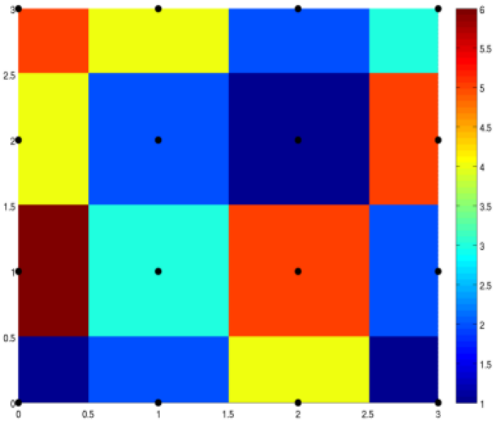


Figure 21: Example of the result of a *nearest-neighbor* interpolation. Figure from Wikipedia.

- Linear interpolation: The intensity value is averaged proportionally from the neighboring voxels.

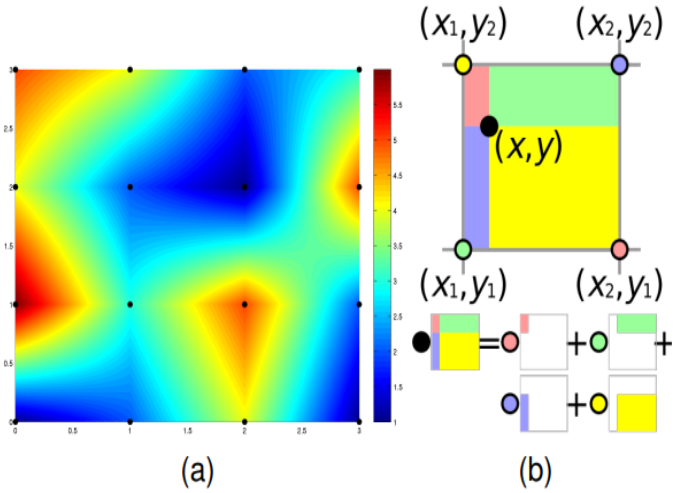


Figure 22: (a) Example of the result of bilinear interpolation. (b) Schematic representation of bilinear interpolation as a mixture of the intensity values of the 4-neighborhood. Figures from Wikipedia (http://en.wikipedia.org/wiki/File:Bilinear_interpolation_visualization.svg).

- In 2D: biliniear interpolation
- In 3D: triliniear interpolation
- Cubic interpolation: Cubic function.

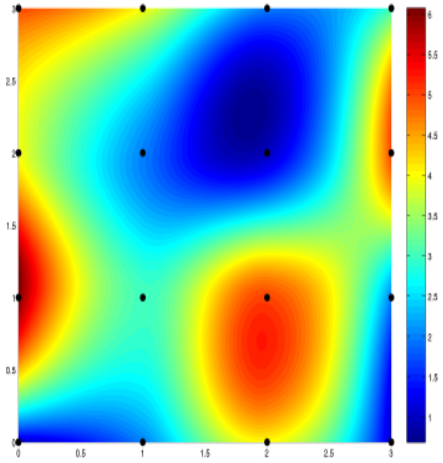


Figure 23: Example of the result of a bicubic interpolation. Figure from Wikipedia

Metrics

$$T^* = \text{argmin}(f(T)) = \text{argmin}(e(T) + s(T))$$

- $e(T)$: data term, similarity of the images $I_{\text{fixed}}(x)$ and $I'_{\text{moving}}(x)$
- $s(T)$: regularization term: used to avoud overfitting, e.g. constraining parameters to certain interval.

Scalar Product

$$D_{SP}(a, b) = -|a^T b|$$

- Scalar product for angle determination between two vectors.
- The smaller the angle between a and b, the larger the amount of the scalar product.
- Since the distance is to be minimized, the negative scalar product is used.
- Simple measure used for intra-modality registrations.

Sum of Squares Difference (SSD)

$$D_{SSD}(a, b) = \frac{1}{N} \sum_i^N |a_i - b_i|^2 = \frac{1}{N} |a - b|^2 = \frac{1}{N} (a - b)^T (a - b)$$

- Simple measure used for intra-modality registrations.
- Intra-modality registration refers to the process of aligning images from the same imaging modality (e.g., two MRI scans, two CT scans, or two X-rays). The SSD is well-suited for this purpose because of its simplicity and the assumption that pixel intensities in the two images should be similar. Small differences in pixel values should correspond to small misalignments or noise, making SSD a good metric. This metric is not good for images from different modalities, because it would not capture meaningful differences, since large differences in pixel intensities might be due to the different imaging characteristics rather than misalignment.

Correlation Coefficient (CC)

$$D_{CC}(a, b) = -\frac{\sum_i^N ((a_i - \overline{a})(b_i - \overline{a}))}{\sqrt{\sum_i^N (a_i - \overline{a})^2 \sum_i^N (b_i - \overline{b})^2}} = -\frac{(a - \overline{a})^T (b - \overline{b})}{|a - \overline{a}||b - \overline{b}|}$$

- Corresponds to the (known) normalized cross correlation.
- A high correlation corresponds to a high agreement.
- Since minimization problem is considered, negative correlation is used.
- Difference from SSD:
  - Invariant to intensity value scaling $(a * s)$.
  - Invariant to addition of a constant intensity offset $(a + c)$.

Normalized Mutual-Information

$$D_{NMI}(a,b) = -\frac{H(a)+H(b)}{H(a,b)}$$

- **Entropy** $H(a) = -\sum_s P_a(s)\log_2 P_a(s)$ ($H(b)$ analog). Measure of included randomness in a (larger values $\rightarrow$ more randomness).
- s are the occurring intensity values in the image: $s = \{a_i \,|\, 1 \leq i \leq N\}$.
- $P_a(s)$ is the probability density (PDF), which is approximated by the normalized histogram in the discrete case.
- **Joint entropy** $H(a,b) = -\sum_{s,t} P_{a,b}(s,t)\log_2 P_{a,b}(s,t)$
- Joint frequency density $P_{a,b}(s,t)$ is approximated with normalized 2D histogram.
- Since a minimization problem is considered, **negative NMI** is used.
- Can be used for **inter-modality registrations**, therefore of **outstanding importance** in practice.
- Mutual information tells you how much information is shared between two images. In simple terms, it measures how much knowing the pixel intensities of one image reduces the uncertainty about the pixel intensities in the other image.
- NMI normalizes the mutual information by dividing it by the entropy of the individual images to avoid bias from image size or entropy scale differences.
- Suppose you have two images, one from a CT scan and one from an MRI scan. The pixel values in these images represent very different physical properties, so directly comparing them using something like SSD (sum of squared differences) wouldn't work. Instead, NMI looks at the statistical relationship between the pixel values in the two images. It does this by analyzing the distribution of pixel intensities (using histograms) and measuring how much information about one image is provided by the other. If the two images are well-aligned, the NMI value will be high because the distributions will match more closely, indicating that the images share more information.

Optimization

Gradient Method

- Iterative method.
- $T^{n+1} = T^n + \beta\nabla f(T^n)$
  - β: Step size
  - Decreasing with n.
- Requirement: sufficiently good initialization, since it cannot be guaranteed that a global maximum is found.

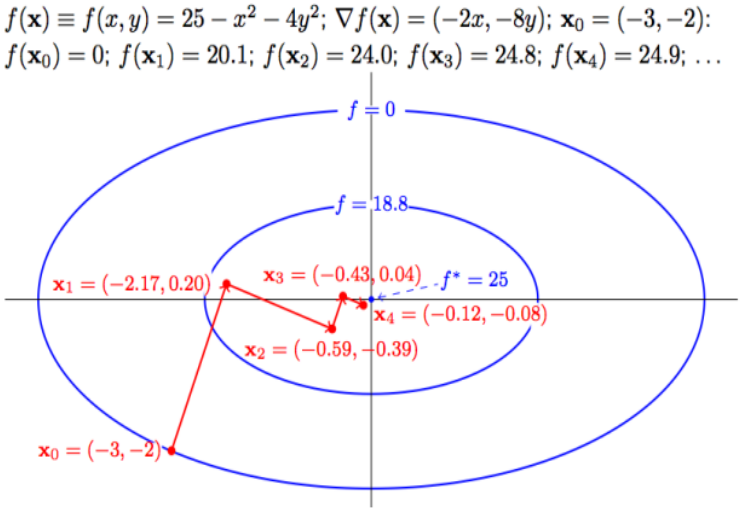


Figure 25: Schematic of the gradient method in 2D for optimizing the nonlinear function $f(x,y)$. Figure by Georgy Gimel'farb, University of Auckland.

Digression: Multidimensional Histograms

- 1D Histograms.

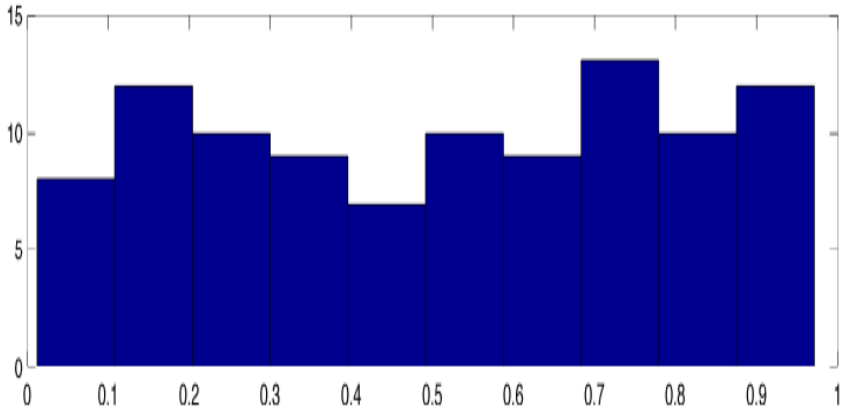


Figure 17: 1-dimensional histogram of a 100-element random vector with values from the range $[0;1]$ (generated in MATLAB using $x = rand(1,100)$).

- 2D Histograms:
  - Can be used to compare the gray values/intensities in the images
  - Spatial arrangement of values is ignored.
  - If spatial information is needed, the histograms of image patches/superpixels can be considered.
- 3D Histograms: for 3D value ranges.

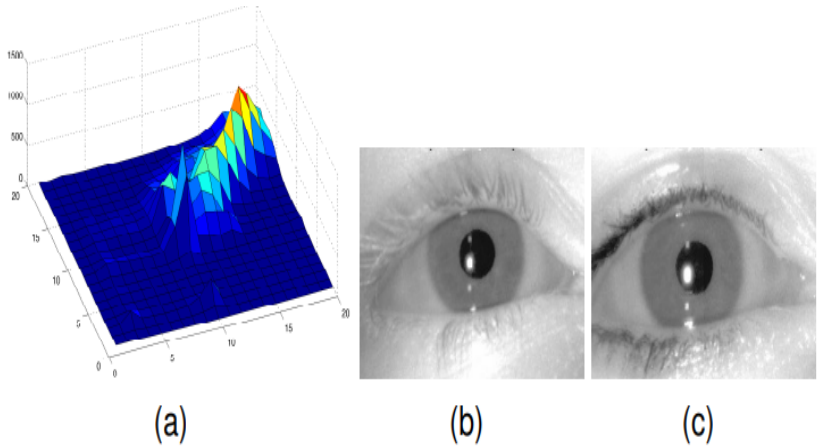


Figure 19: (a) Two-dimensional histogram with 20 *bins*, the first dimension is the image in (b), the second dimension the image in (c). The two images (b) and (c) are different here, so the pairs of values of the accumulator are no longer on a main diagonal.

- ND Histograms.

6: Disease Detection with Computer Vision

Examples

- Dermatology: dealing with skin. E.g., skin image → algorithm → cancerous or not.
- Ophthalmology: diagnosis and treatment of eye disorders.

- Histopathology: examining tissue under microscope. E.g. where cancer has spread. Images are too large (100000x100000), so images are divided into patches, labeled and fed into a network.

Class Imbalance, Multi-task and Small Training Set Size

- **Prevalence**: normal cases are a lot more prevalent than disease cases.
- **Binary cross entropy loss**:

$$L(X, y) = \begin{cases} -\log P(Y = 1|X) \text{ if } y = 1 \\ -\log P(Y = 0|X) \text{ if } y = 0 \end{cases}$$

- **Class imbalance**: when there is a class imbalance, the loss contribution of prevalent classes will be much higher. So, the model will train to recognize those classes more. To deal with this, we give **more weight to minority class**:

$$L(X, y) = \begin{cases} w_p \times -\log P(Y = 1|X) \text{ if } y = 1 \\ w_n \times -\log P(Y = 0|X) \text{ if } y = 0 \end{cases}$$

- w_p and w_n is the **frequency** of a class in the dataset. E.g., $w_p = \frac{6}{8}$ and $w_n = \frac{2}{8}$. This way, minority class contributes **with the same value** to the total loss as prevalent class.
- **Resampling**: resample the dataset such that we have an equal number of instances in classes. However, this means we **throw away data from prevalent classes** (undersampling). There are **oversampling** and **undersampling** methods.
- **Multi-label**: having more than 2 classes to classify. The prediction probabilities go from e.g., $(0.2, 0.8)$ to $(0.3, 0.1, 0.8)$. We can use **multi-label loss**, where we add losses from all the tasks: $L = L_{mass}(X, y) + L_{pheumonia}(X, y) + L_{edema}(X, y)$. To account for task imbalance, we modify the individual losses in the following way:

$$L_{mass}(X, y) = \begin{cases} w_{p,mass} \times -\log P(Y_{mass} = 1|X) \text{ if } y = 1 \\ w_{n,mass} \times -\log P(Y_{mass} = 0|X) \text{ if } y = 0 \end{cases}$$

- $w_{p,mass}$ and $w_{n,mass}$ now relate to class imbalance for an individual class.

$$L(X, Y) = \sum_{tasks} [y_k \cdot w_{p,k} \cdot -\log P(Y_k = 1|X) + (1 - y_k) \cdot w_{n,k} \cdot -\log P(Y_k = 0|X)]$$

- k: indexes of the tasks.

Small training set size challenge: ways to deal with this challenge:

1. **Transfer learning**: **pretrain** a model on a different dataset, like ImageNet. Then **fine-tune** the pretrained model on the dataset you have. **Early layers** capture **low-level** features, while **deep layers** capture **high-level** features, specific to a task. So, when we fine-tune a network, we **freeze early layers**.
2. **Data augmentation**: generate more samples for training. **Do augmentations reflect variations in real world? Do augmentations keep the label the same?** There are: **rotate + flip**, **rotate + color noise**, **gaussian noise**.

Model Testing

- We split data into **train**, **validation** and **test** sets. Training set is used for the development of models. Validation set is used for hyperparameter tuning and selection of models. Test set is used for reporting of results. We can also use **cross-validation** to **reduce variability in our estimate of the model performance**.
- **Data leakage**: phenomenon when there is same data in training and test sets. E.g., make sure there are **no same patients** in the **train and test** sets. This leads to overly optimistic model perfomance.
- **Sampling**: sample test set so that there is at least X of minority class (e.g., $X = 50$). Same for validation set. This is called **stratification**. First sample test set, then validation and last train. We use same sampling strategy to ensure that validation and test sets have same class distribution.
- **Ground truth**: there are 2 challenges:
  1. **Consensus voting**: use a group of human experts to determine the ground truth. Think of **ensemble** of models.
  2. **Additional testing**: **more definitive test** which provides additional information to set the ground truth (e.g. use CT scans with addition to X-Ray image).

7: Evaluating Models

$$\text{Sensitivity} = \text{Recall} = P(+|1) = \frac{TP}{TP+FN} = P(\text{Positive Test}|\text{Condition Present}) = (\text{all diseased patients})$$

$$\text{Specificity} = P(-|0) = \frac{TN}{TN+FP} = P(\text{Negative Test}|\text{Condition Absent}) = (\text{all non-diseased patients})$$

$$\text{Prevalence} = P(1) = \frac{TP+FN}{TN+FP+TP+FN}\left(\frac{\text{diseased patients}}{\text{total number of patients}}\right)$$

$$\begin{aligned}\text{Accuracy} &= \text{Sensitivity} \times \text{Prevalence} + \text{Specificity} \times (1 - \text{Prevalence}) \\ &= P(\text{correct} \cap 1) + P(\text{correct} \cap 0) \\ &= P(\text{correct}|1)P(1) + P(\text{correct}|0)P(0) \\ &= P(+|1)P(1) + P(-|0)P(0)\end{aligned}$$

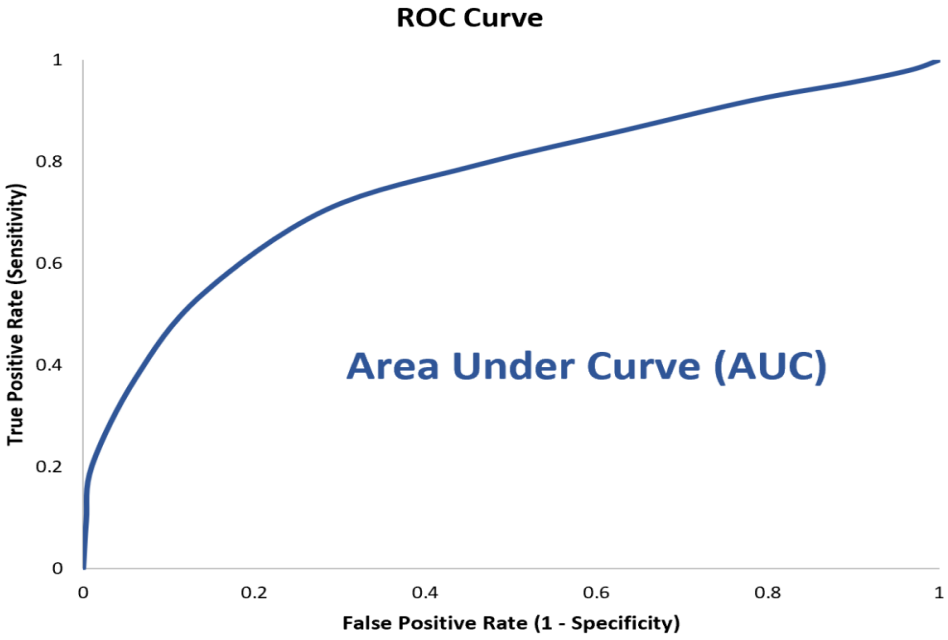
$$\text{PPV (Positive Predictive Value)} = P(1|+) = \text{Precision} = \frac{TP}{TP+FP}(\text{total positive results})$$

$$\text{NPV} = P(0|-) = \frac{TN}{TN+FN}(\text{total negative results})$$

PRAUC

The model outputs probabilities of clases. We need to choose a threshold. As we vary threshold, metrics values also vary. As $t \to 1$, Recall $\to 0$, as the number of FN increases. Precision and Specificity $\to 1$.

The model is primarily predicting negatives as the threshold increases. In the graph below, we start with $t = 1$, which decreases to $t = 0$.



Confidence Intervals

- As n increases, confidence intervals shrink.
- With 95% confidence p lies in the interval $[\mu - x, \mu + x]$

8: Image Segmentation on MRI Images

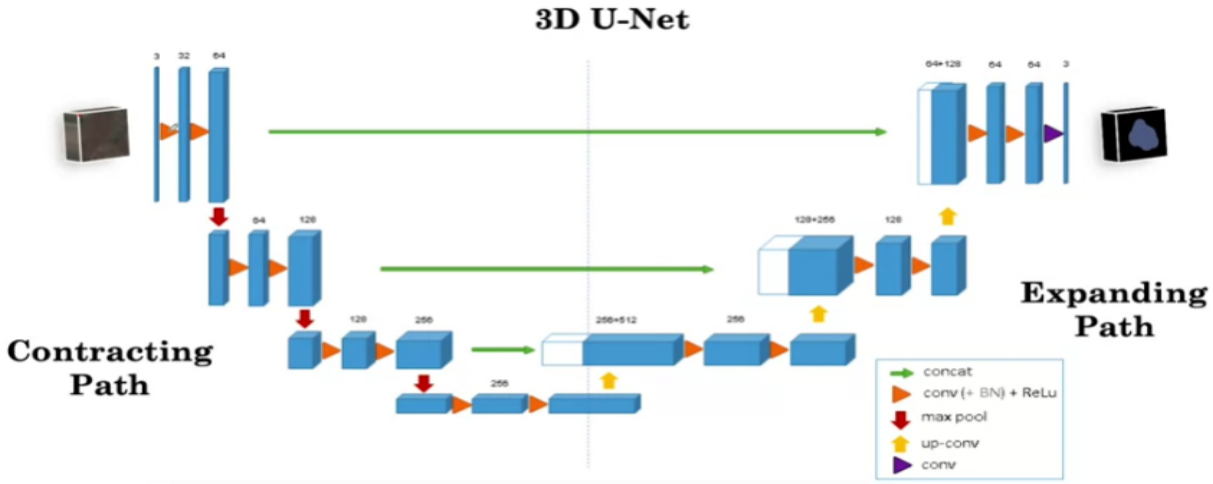
Segmentation

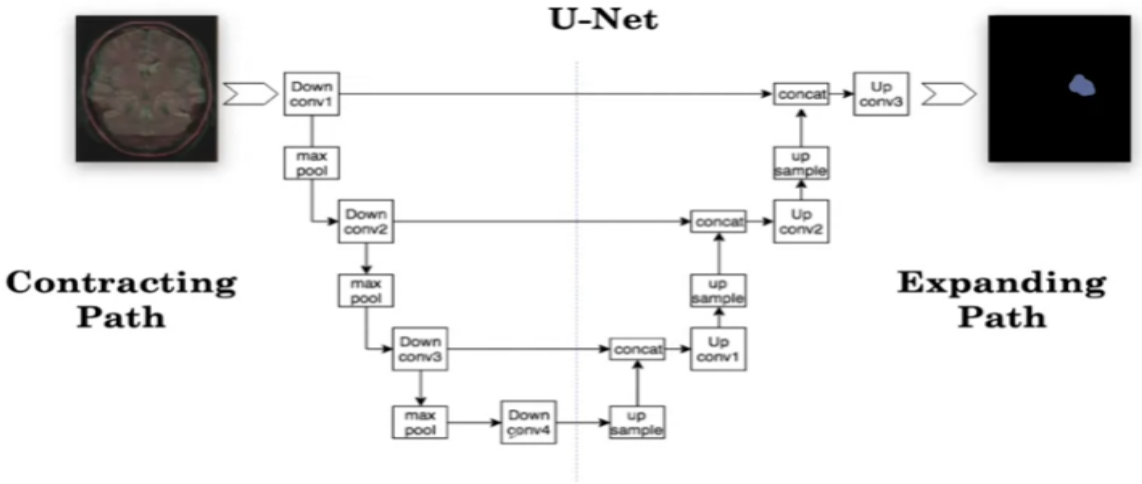
Determine the class of each pixel in the image. There are:

- 2D approach: passing each slice of 3D image into a segmentation model and get an output for each slice, then combine them. This is bad: if there a tumor in 1st slice, there is also a tumor in 2nd slice. The model does not actually learn that.
- 3D approach: pass whole image to model. To reduce the memory usage, we break up the image into pieces and pass pieces as inputs. However, we might lose a spatial context.

Segmentation Architectures

U-Net (2D and 3D)





Metrics

Dice Similarity Coefficient for 1 Class

$$DSC_1(A, B) = \frac{2 \times |A \cap B|}{|A| + |B|}$$

$$DSC_1(f, x, y) = \frac{2 \sum_{i,j}^{n} p_{ij} q_{ij} + \epsilon}{\sum_{i,j}^{n} p_{ij} + \sum_{i,j}^{n} q_{ij} + \epsilon}$$

- x: input image.

- p: model output.
- q: ground truth.
- ϵ: to avoid division by 0.
- $DSC = 0$: complete mismatch.
- $DSC = 1$: perfect match.

Dice Similarity Coefficient for Multiple Classes

$$DSC_N(f, x, y) = \frac{1}{N}\left(\sum_{n=1}^{N} DSC_n(f, x, y)\right)$$

- N: number of classes

Loss

Soft-Dice Loss for 1 Class

$$L_{dice,1}(p, q) = 1 - \frac{2\sum_{i,j}^{n} p_{ij}q_{ij} + \epsilon}{\sum_{i,j}^{n} p_{ij}^2 + \sum_{i,j}^{n} q_{ij}^2 + \epsilon}$$

- p: predictions, map of probabilities of a binary class for each pixel.
- q: ground truths.
- $L_{dice} = 0$: perfect match.
- $L_{dice} = 1$: complete mismatch.

We take $1 - ...$ so that **low loss corresponds to high overlap** and **high loss corresponds to low overlap**.

Soft-Dice Loss for Multiple Classes

$$L_{dice,N}(p, q) = 1 - \frac{1}{N}\sum_{n=1}^{N} \frac{2\sum_{i,j}^{n} p_{nij}q_{nij} + \epsilon}{\sum_{i,j}^{n} p_{nij}^2 + \sum_{i,j}^{n} q_{nij}^2 + \epsilon}$$

- N: number of classes

Challenges

1. **Generalization**: tuberculosis is quite prevalent in India, but unlikely to be as prevalent in the hospitals where we've trained our model in the US. Before applying the model in India, we'd want to test the model on its ability to detect tuberculosis.

External Validation

Evaluate a model on a set that was taken from different population (distribution). If we find that we're not generalizing to the new population, then we could get a few more samples from the new population to create a small training and validation set and then fine-tune the model on this new data.

Types of Data

- **Retrospective**: historical data.
- **Prospective**: real-world data.

Measuring Outcomes

- **Decision curve analysis**: help quantify the net benefit of using a model to guide patient care.
- **Randomized controlled trial**: we compare patient outcomes for patients on whom the AI algorithm is applied versus those on whom the AI algorithm is not applied.
- Check algorithmic **bias**.
- Model **interpretation for humans** (why and how a model makes a certain decision).

9: Deep Learning Based Image Analysis

Training NNs

1. Select loss function.
2. Initialize weights.
3. Update weights using backprop:
    1. Feed a batch of training samples through the network, compute loss.
    2. Compute partial derivatives of loss with respect to weights.
    3. Update weights with a certain learning rate along negative gradient.
4. Repeat step 3 for a number of epochs.
    1. **Epoch** denotes one pass over all training data.
    2. **Learning rate** high initially, reduced over time.
    3. **Early stopping**: use a validation loss to decide when to stop training.

Activation Functions

Softmax

Converts raw scores s_j into predictions of conditional probabilities:

$$P(y = k|x = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

- x_i: feature vector of sample i
- y: correct class label for input x.

$$\begin{bmatrix} 3.0 \\ 1.0 \\ -2.5 \end{bmatrix} \rightarrow^{\exp} \begin{bmatrix} 20.08 \\ 2.7 \\ 0.08 \end{bmatrix} \rightarrow^{normalize} \begin{bmatrix} 0.87 \\ 0.12 \\ 0.00 \end{bmatrix}$$

Losses

Total Loss

$$L = \frac{1}{N} \sum_i L_i$$

Hinge Loss

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

- s_j: score for class j.
- y_i: correct class label.

Cross-Entropy Loss

$$L_i = -\ln \left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \right)$$

$$\begin{bmatrix} 3.0 \\ 1.0 \\ -2.5 \end{bmatrix} \rightarrow^{\exp} \begin{bmatrix} 20.08 \\ 2.7 \\ 0.08 \end{bmatrix} \rightarrow^{normalize} \begin{bmatrix} 0.87 \\ 0.12 \\ 0.00 \end{bmatrix} \rightarrow^{\log} -\ln(0.12) = 2.12$$

Optimization

- Given a random initialization of W, we need to slowly change the parameters to minimize the loss function.
- **Gradient descent**:
    - while True:
        - weights_grad = eval_grad(loss_fun, data, weights)
        - weights += -lr * weights_grad

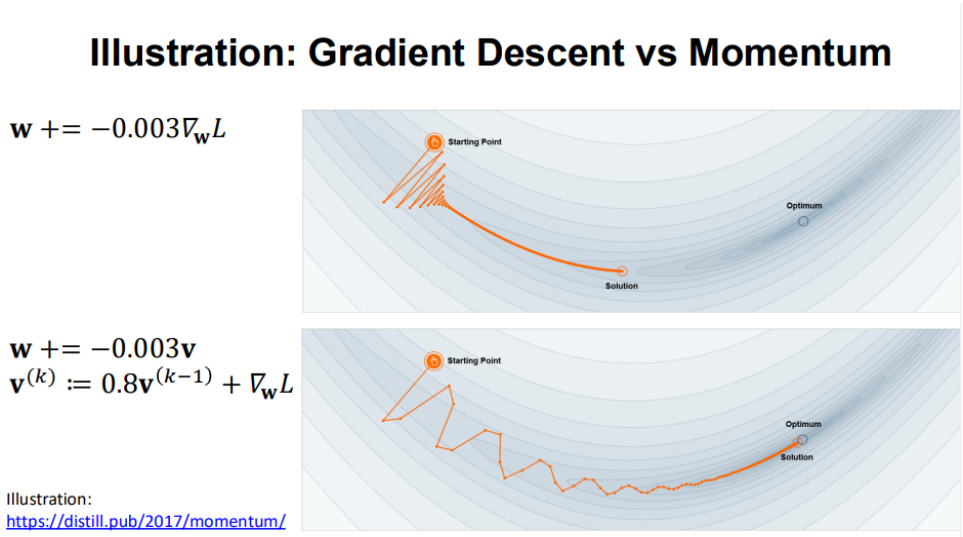
- **Stochastic (mini-batch) gradient descent**: only use a small, randomly selected part of the training set to compute gradient. Improves gradient estimation speed but causes inaccuracies ("noise").

Forward/Backward Pass

- **Forward pass**: values are propagated forward from the input stage through the hidden stages to the output stage where a prediction is made and the loss is computed.
- **Backward pass**: partial derivatives of the loss with respect to the weights are computed using the chain rule, propagating values from the final loss back towards the input layer.

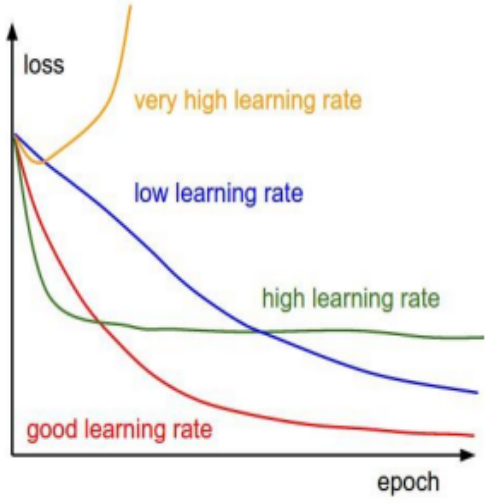
Momentum Update

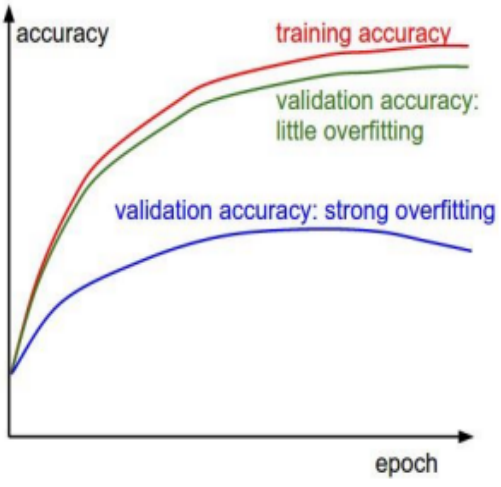
- Weight update in **standard gradient descent**: $w+ = -\lambda \nabla_w L$
- **Momentum** update: $w+ = -\lambda v$, $v^{(k)} := \mu v^{(k-1)} + \nabla_w L$
  - Velocity v is initialized to $v^{(0)} = 0$, and increases over training iterations if the gradient is consistent. This can compensate gradient noise and curvature in the loss function and help overcome shallow suboptima.
  - Momentum $0 \leq \mu < 1$ and learning rate λ are hyperparameters.
  - Physical analogy: unit mass ball rolling over the loss landscape.



Monitoring Training Loss

- Monitoring the training loss allows us to check that optimization is converging, and to adjust the optimizer or learning rate when needed.
- Comparing training to validation loss allows us to assess over/underfitting, and to adjust complexity or regularization parameters accordingly.





Learning Rate Decay

- **Step decay**: reduce the learning rate after a few epochs, or when validation loss reaches a plateau.
- **Linear decay**: linear reduction of an initial learning rate λ_0 to a final learning rate λ_T, effective from iteration $k = T$: $\lambda_k = (1-\alpha)\lambda_0 + \alpha\lambda_T, \alpha = \min\left(\frac{k}{T}, 1\right)$
- **Exponential decay**: reduction with a decay rate γ: $\lambda_k = \lambda_0 e^{-\gamma k}$

Optimizers

AdaGrad

- Problem: the update $-\lambda\nabla_w L$ has a larger effect on parameters w_i that more strongly influence the loss. Training requires limiting λ to most sensitive neurons. Other neurons require higher λ to make suitable progress.
- Solution: AdaGrad determines parameter specific learning rates based on accumulating squares partial derivatives: $w+ = -\frac{\lambda\nabla_w L}{\sqrt{c}+\epsilon}, c+ = (\nabla_w L)^2$
  - c has same dimension as w.
  - initially $c = 0$.
  - square, square root, division are component-wise.

Adam: AdaGrad + Momentum

Adam combines adaptive learning rates and momentum updates.

$$v := \beta_1 v + (1-\beta_1)\nabla_w L$$
$$c := \beta_2 c + (1-\beta_2)(\nabla_w L)^2$$
$$w+ = -\frac{\lambda v}{\sqrt{c}+\epsilon}$$

- Cache c is leaky with factor β_2 to avoid low learning rates.
- Not shown: separate equations for "warm-up phase" in which v and c are close to 0.
- Typical $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$

Weight Initialization

- **All zero**: results in same gradient everywhere in backward pass, and therefore to exactly the same weight updates. **Do not use!**
- **Small random numbers**: initialize using a zero-centered Gaussian distribution with 0.01 standard deviation. This works for shallow networks but activations "die" in deeper ones: $\hat{y} = W_{l-1}W_{l-2}\dots W_2 W_1 X$
- **Xavier et al.**: Initialize weights from a Gaussian distribution with standard deviation $\sqrt{1/N}$ where N is the number of input connections.
  - Reason: attempt to achieve same variance in output as in inputs: $z = w_1 x_1 + \dots + w_N x_N + b$. When adding independent Gaussian random variables, variances add. Thus, $var(w_i) = \frac{1}{N}$
- **He et al.**: derivation above neglects the nonlinear activation. If ReLu is used, then initialization from Gaussian with standard deviation of $\sqrt{2/N}$ works better.
- **Remark**: biases are often initialized to 0.

Batch Normalization

- Problem: constraining all neurons to zero mean / unit variance makes the network less powerful.
- Solution: introduce learnable parameters β, γ and set $H'' = \gamma H' + \beta$ with H: activations, unprocessed outputs of a layer of the network before applying any normalization, H' is the result of applying BN: $B' = \frac{H-\mu}{\sigma}$ with μ: mean of the activations H in the batch, σ: std of the activations H in the batch.
- Effect: activations can have arbitary mean and variance.
- Benefit: modified network is easier to train. Mean and std depend directly on β, γ and not on complex interdependence between the layers.
- Facilitates:
  - Faster learning.
  - Higher accuracy.
  - Permits larger learning rates.
  - Reduces sensitivity to bad weight initialization.
- It is common to standardize input to the first layer.
- No consensus on whether BN should be used before or after activation.

Regularization

- Regualization is used to avoid overfitting.
- **L2 regularization**: $L = \frac{1}{N}\sum_i L_i + \lambda R(W), R(W) = \sum_k \sum_l W_{k,l}^2$ with i indexing the input, k the neuron and l the input feature.
- **Dropout**: randomly ignore neurons and their corresponding input and output connections with probability p during training.
  - Used during testing to generate probabilistic output.
- **Early stopping**: use parameters from epoch in which validation error was minimal.

Data Augmentation

- **Data augmentation**: altering input image:
  - Random crops on the original image.
  - Translations.
  - Horizontal reflections.
  - Warping.
  - Changes in color or contrast.

Hyperparameter Tuning

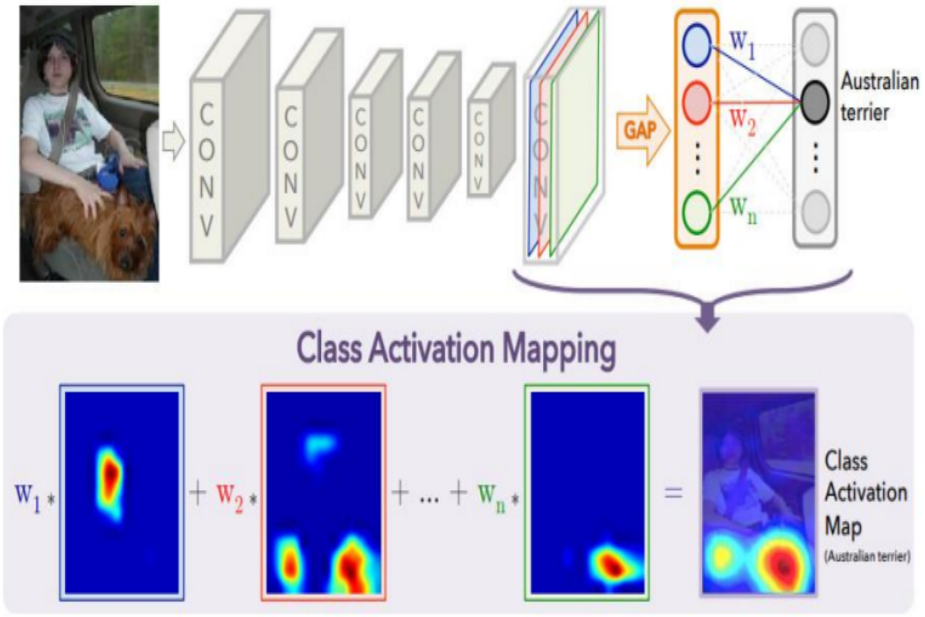
- Hyperparameters: all non-trainable, such as optimizer, learning rate, number of epochs, regularization type and strength.
- Several hyperparameters can have a strong effect on the final result, default settings might not be optimal for all inputs and tasks.
- Settings are determined by trying values within a plausible range through grid search, random search, efficient search.

Image Analysis

- **Clever Hans effect**: when NNs learn, e.g., they might use signs of treatment or choice of imaging device instead of signs of the disease itself for classification, like a presence of drain tube indicates treatment of a pneumothorax.
- **Lack of fairness**: NNs might discriminate against people based on their gender or ethnicity (everything comes from given data). Use **Grad-CAM** to revel where NN *looks*.
- **Domain shift**: accuracy is often reduced, sometimes drastically, when characteristics of training and test images differ.
- **Adversarial attacks**: NNs can be misled by targeted image manipulations, even if they might not be visible to the human eye, like adding slight noise to an image: $X + 0.04 \cdot \text{ noise } = \text{misclassified}$.
- **Reconstruction of confidential training data**: model updates (gradients) sent in *federated learning* can sometimes be reverse-engineered to reconstruct recognizable images, compromising privacy.
  - **Federated learning**: training process happens locally at individual institutions (e.g., hospitals), and only the updates to the model's parameters (gradients) are shared with a central server. Federated learning keeps the sensitive data confidential, reducing the risk of exposure.

Visualizing NNs

- **Occlusion sensitivity**: studies how hiding parts of the input impacts the network's confidence. Computation involes a large number of forward passes. Results depend on the size and shape of the occluder.
- **Class activation maps**: alternative for localizing objects using networks trained for classification. Neither requires multiple forward passes nor back propagation. Assumes that network uses **Global average pooling (GAP)**. Class activation maps are simply computed as a **weighted average** of the **activations** in the **final convolution layers**. Then result is upsampled to original image resolution and overlaid. Justification:
  - GAP performs $F_k = \sum_{x,y} f_k(x, y)$
  - Class scores obtained as $S_c = \sum_k w_k^c F_k = \sum_k w_k^c \sum_{x,y} f_k(x, y)$
  - Swapping sums yields $S_c = \sum_{x,y} \sum_k w_k^c f_k(x, y)$
  - Class activation map defined as $M_c(x, y) = \sum_k w_k^c f_k(x, y)$
  - Therefore $S_c = \sum_{x,y} M_c(x, y)$



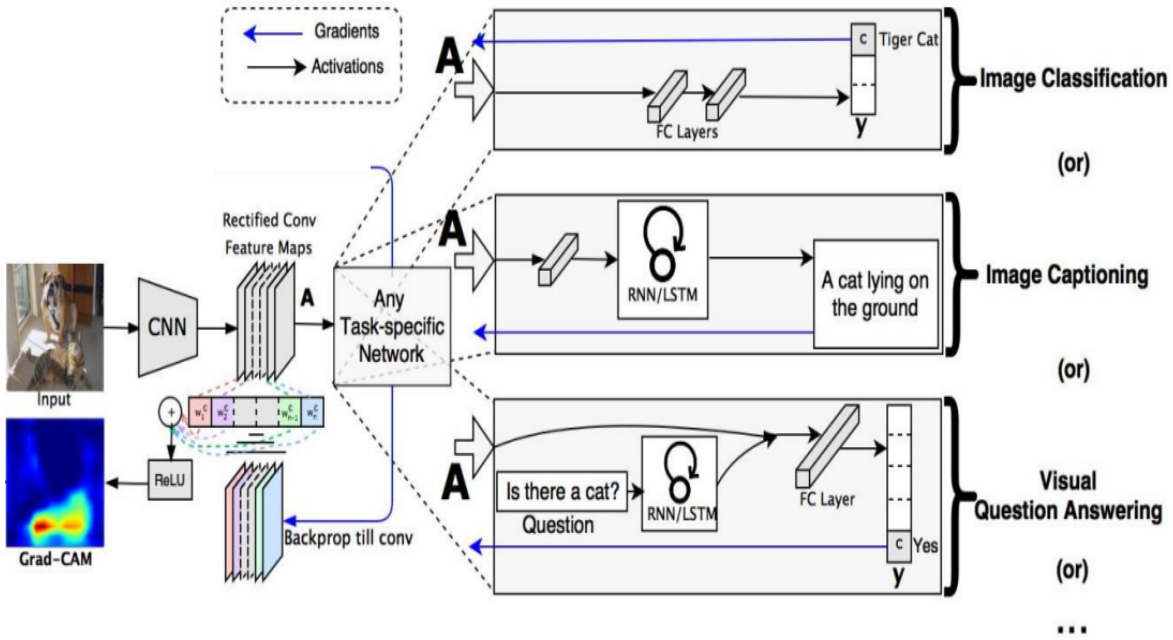
- **Grad-CAM**: generalizes CAM to arbitrary tasks and architectures. Backpropagates output of interest to last convolutional layer. Combines activation maps with weights from GAP on gradients.
  - **Neuron importance weights** α_k^c for activation map k and output score y^c computed via GAP:
  - Grad-CAM is computed as the corresponding weighted sum of activation maps, rectified to disregard activations with a negative influence on the desired output: $M_{GC}^c = \text{ReLU}\left(\sum_k \alpha_k^c A^k\right)$

$$\alpha_k^c = \frac{1}{Z} \sum_{i=1}^{H} \sum_{j=1}^{W} \frac{\delta y^c}{\delta A_{ij}^k}$$

- α_k: weight for the $k-th$ feature map.
- H and W are width and height of the feature map.
- $\frac{\delta y^c}{\delta A_{ij}^k}$: gradient of the output score y^c (e.g., "dog" score) with respect to the feature map activation A_{ij}^k at spatial location (i, j) in the $k-th$ filter.
- $Z = H \times W$: total number of spatial locations.

Algorithm:

1. The input image is passed through a CNN. The CNN extracts feature maps from the image.
2. Rectified convolutional feature maps refer to the outputs of the last convolutional layer after applying a ReLU activation function.
3. The rectified feature maps are then fed into a task-specific network.
4. Once the forward pass is complete, Grad-CAM focuses on the output of interest (e.g., the probability of the "dog" class). Gradients are computed by backpropagating the error from this specific output to the last convolutional layer.
5. The gradients are spatially averaged (i.e., global average pooling) over width and height for each feature map (if feature map is of shape 10×10, the gradients will also have the same spatial shape, 10×10, for that feature map). This gives a single weight (importance score) for each feature map. These weights represent the importance of each feature map in the context of the output of interest (e.g., "dog").
6. A weighted sum of the rectified convolutional feature maps is computed using these importance weights. The result is a class activation map that highlights regions in the image that most influence the decision for the output of interest



- **Attention networks**: some NNs include an attention mechanism that determines how much certain video frames or image regions should contribute towards the final decision. The primary goal is often to increase accuracy, but attention also provides intrinsic **interpretability**. We can use these attention weights to see which parts of the image contribute to the output.