

Medical Image Processing

Digital Imaging

A digital image is a numerical matrix of pixels where the values indicate the intensity of each pixel. Number of pixels N is the product of each dimension of the image. Number of possible intensity values v_i is 2^n , where n is the pixel depth (number of bits per pixel). Color images have three channels, one for each color property.

- **Size** in disk (bits): $N \cdot n \cdot \text{channels} + \text{additional file data}$ (name, metadata...)
- **Dynamic range**: $DR = v_{\max} - v_{\min} \rightarrow DR(dB) = 20 \log_{10} DR$

Color encodings

- **Grayscale**: intensity values (no color). Typical screens: 256 values. Hospital screens: 1024 values.
- **RGB**: channels are *red-green-blue*. Colors are a combination of these three. There are $(2^n)^3$ possible colors. (0,0,0) is black and (256,256,256) is white. Useful in color generation (hardware implementation to display on screens).
- **HIS**: channels are *hue-intensity-saturation*. Hue describes pure color, intensity indicates brightness, saturation measures color dilution with white. Useful in color description.
- **Pseudocolor**: a grayscale image can be visualized in “false” color using a lookup table that matches intensity values to RGB colors.

Image formats

- **JPEG**: lossy compression, 8-bit grayscale and 24-bit color images, generational degradation (progressive loss of quality after several compressions and decompressions).
- **TIFF, PNG, BMP**: lossless compression, good quality and true color (up to 16-bit). Images processed with ImageJ should be saved in TIFF format.
- **GIF**: 8-bit pixels (256 colors only), background transparency. Good for logos, diagrams.

Histogram statistics

- **Histogram**: Representation of pixel distribution along intensity values. The sum over the histogram yields total number of pixels: $\sum h(v_i) = N$.

$$h(v_i) = n_i, \text{ where } \begin{cases} v_i: \text{pixel value} \\ n_i: \text{number of pixels with value } v_i \end{cases}$$

- **Normalized histogram** (probability of a pixel value): $P(v_i) = \frac{h(v_i)}{N}$, so that $\sum P(v_i) = 1$.

- **Cumulative distribution function**: $CDF(v_k) = \sum_{i=1}^k P(v_i) = \sum_{i=1}^k \frac{n_i}{N}$

- **Moment**: $m_k = \sum P(v_i) \cdot v_i^k$

- First moment (**mean**): $m_1 = \sum P(v_i) \cdot v_i = \bar{v}$

- **Central moment**: $\mu_k = \sum P(v_i) \cdot (v_i - \bar{v})^k$

- Second central moment (**variance**): $\mu_2 = \sum P(v_i) \cdot (v_i - \bar{v})^2 = \sigma^2$

- Third central moment (**skewness**): $\mu_3 = \sum P(v_i) \cdot (v_i - \bar{v})^3$

- Fourth central moment (**kurtosis**): $\mu_4 = \sum P(v_i) \cdot (v_i - \bar{v})^4$

- **Information content** in one pixel value (bits): $I(v_i) = \log_2 \left(\frac{1}{P(v_i)} \right) = -\log_2 P(v_i)$

- **Entropy** (bits/pixel): $H = \sum P(v_i) I(v_i) = -\sum P(v_i) \log_2 P(v_i)$

Noise increases entropy and sharpening decreases it. Maximum entropy when the image is **equalized** and all values have same probability: $H_{\max} = -\sum_{i=1}^{2^n} \frac{1}{2^n} \log_2 \left(\frac{1}{2^n} \right) = \sum_{i=1}^{2^n} \frac{1}{2^n} n = n$

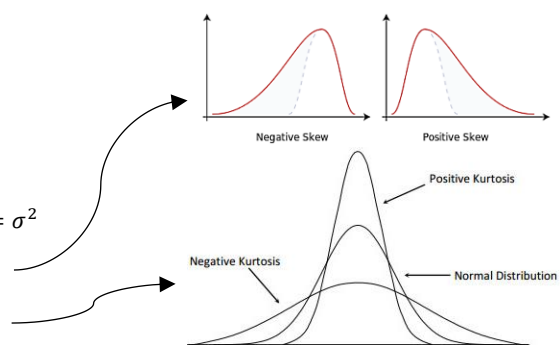
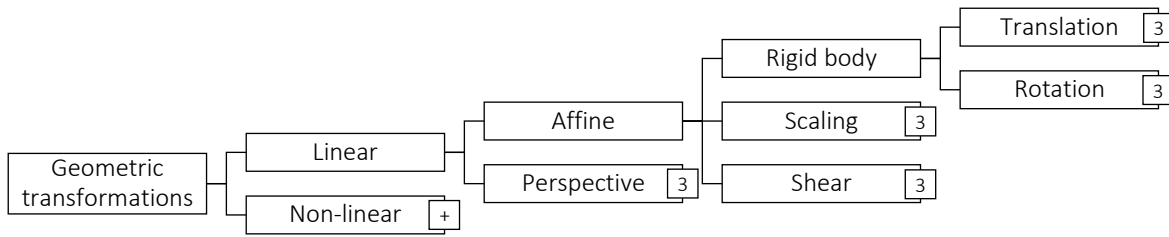


Image Enhancement I: Geometric Transformations



Each linear GT has 3 *degrees of freedom* → 15 DOF in linear GT. Homogeneous coordinates (additional row) are needed for translation.

- **Rigid:** shape is preserved. Used to compare images of same subject, brain, bone.
- **Affine:** straight parallel lines are maintained. Used to compare different subjects (sometimes) or specimens (small animals).
- **Perspective:** straight lines are maintained; also called projective.
- **Non-linear:** straight lines can be curved. Used for motion compensation, atlas-based analysis or inter-subject quantification.

AFFINE TRANSFORMATIONS

Affine in 2D

Translation matrix:

$$T = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation matrix:

$$R = \begin{bmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Scaling matrix:

$$Sc = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Shear matrix:

$$Sh = \begin{bmatrix} 1 & s_{xy} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Transformation formula:

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

Combine matrices:

$$I' = M_2 \cdot M_1 \cdot I$$

Inverse transformation:

$$I = M^{-1} \cdot I'$$

Affine in 3D

Translation matrix:

$$T = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scaling matrix:

$$Sc = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Shear matrix:

$$Sh = \begin{bmatrix} 1 & s_{xy} & s_{xz} & 0 \\ 0 & 1 & s_{yz} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformation formula:

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

Rotation matrix: $R = R_z \cdot R_y \cdot R_x$

Rotation around x axis:

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \alpha_x & -\sin \alpha_x & 0 \\ 0 & \sin \alpha_x & \cos \alpha_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation around y axis:

$$R_y = \begin{bmatrix} \cos \alpha_y & 0 & \sin \alpha_y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \alpha_y & 0 & \cos \alpha_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation around z axis:

$$R_z = \begin{bmatrix} \cos \alpha_z & -\sin \alpha_z & 0 & 0 \\ \sin \alpha_z & \cos \alpha_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Combine matrices:

$$I' = M_2 \cdot M_1 \cdot I$$

Inverse transformation:

$$I = M^{-1} \cdot I'$$

NON-LINEAR TRANSFORMATIONS

See [NON-LINEAR REGISTRATION](#).

INTERPOLATION

After transforming the image, new values are obtained through interpolation in the original image.

- **Nearest neighbor interpolation:** fast and simple but produces blocking artifact.

$$P'(x', y', z') = P(\text{round}\{f^{-1}(x', y', z')\})$$

- **Bilinear interpolation:** 8 pixels in 3D, produces smoothing.

$$P'(x', y') = P(x_1, y_1) \cdot \text{green} + P(x_2, y_1) \cdot \text{red} + P(x_1, y_2) \cdot \text{yellow} + P(x_2, y_2) \cdot \text{blue}$$

- **Higher order** (cubic, splines): better approximation but more surrounding pixels and calculations, produces ringing artifact (oscillations).

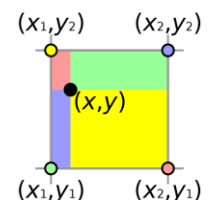
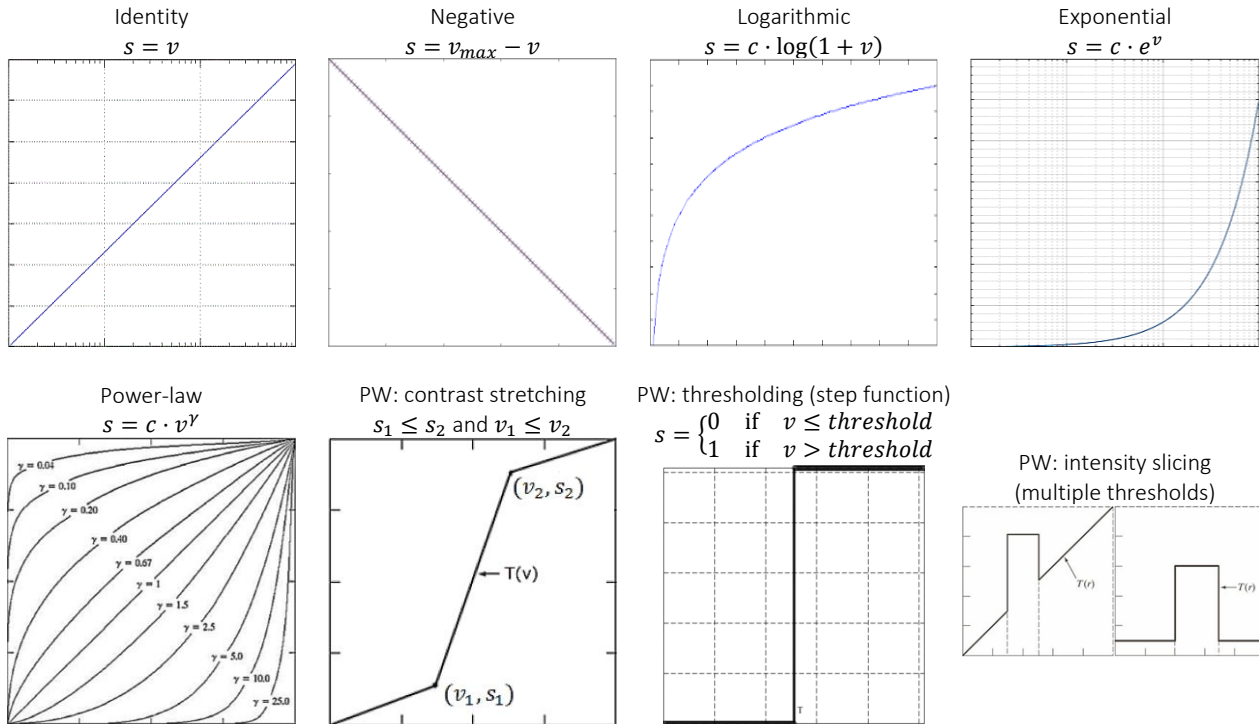


Image Enhancement II: Spatial Filtering

POINT PROCESSING

$s = T(v)$, where v is the input pixel value and s is the output pixel value. PW means *piecewise*.



Equalization

Automatically maximizes information (entropy), achieves a flat **histogram**. The maximum value of the resulting histogram is $2^{n'} - 1$, which may or may not be the original maximum value $2^n - 1$.

$$s_k = T(v_k) = (2^{n'} - 1) \cdot CDF(v_k) = (2^{n'} - 1) \sum_{i=0}^k \frac{n_i}{N}, \text{ where } \begin{cases} k = 0, 1, \dots, 2^n - 1 \\ N: \text{total number of pixels} \\ n: \text{original bit-depth} \\ n': \text{bit-depth achieved} \end{cases}$$

Algebraic operations

Pixel-by-pixel operations between images of same size.

- **Subtraction:** preserve contrast agent, remove shading patterns or stationary components $\rightarrow I_s = I_1 - I_2$
- **Multiplication:** segmentation, multiply region of interest by 1, the rest by 0 $\rightarrow I_s = I_1 \times I_2$
- **Division:** correct non-homogeneous response $\rightarrow I_s = I_1 / I_2$
- **Averaging:** average several noisy images to reduce noise $\rightarrow I_s = \frac{1}{K} \sum_{i=1}^K I_i$, where K is number of images taken.

Local enhancement

Apply transformation only to pixels with specific characteristics (standard deviation, mean value, location, local histogram...).

MASK PROCESSING

$g(x, y) = T[f(x, y)]$, where T applies a convolution with a $k \times k$ mask or kernel: flip mask horizontally, center it on one pixel, multiply each mask coefficient by the corresponding pixel of the original image, and sum the products to obtain the new value of the pixel centered. The same mask is used for every pixel in the image. Issues: needs normalization and quantization, border of the image (padding), choosing the center element in an even-sized mask.

Identity

Does not change the image $\rightarrow \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$



Smoothing

Linear, used to blur edges, remove small ($size \leq 2k$) or low-contrast ($size < 2FWHM$) details, reduce noise, bridge small gaps, detect big objects, correct false contouring. 2D mask separable into two 1D masks: less computation time (1D mask needs $2kN$ and 2D mask needs k^2N).

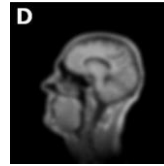
- Box mask $\rightarrow \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

- Weighted average mask: reduces blurring $\rightarrow \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$

- Gaussian filter: uses a Gaussian distribution (circularly isotropic) $\rightarrow h(x, y) = Ae^{-\frac{x^2+y^2}{2\sigma^2}}$

- Pascal triangle: approximates a Gaussian distribution $\rightarrow H_k = \frac{1}{4^{k-1}} (r_k^T \cdot r_k)$, where r_k is the k^{th} row.

$$\begin{array}{ccccccc}
 & & & & 1 & & \\
 & & & 1 & 1 & & \\
 & & 1 & 2 & 1 & & \\
 & 1 & 3 & 3 & 1 & & \\
 & 1 & 4 & 6 & 4 & 1 & \\
 1 & 5 & 10 & 10 & 5 & 1 & \\
 & 1 & 6 & 15 & 20 & 15 & 6 & 1 \\
 & & 1 & 7 & 21 & 35 & 35 & 21 & 7 & 1
 \end{array}$$



Edge detection

Linear (convolution), used to extract edges, detect objects. Makes use of first (edge extraction) or second (edge enhancement) differentiation (from left to right and from top to bottom).

- Horizontal gradient mask: detects vertical edges $\rightarrow \frac{\partial f}{\partial x} = f(x+1, y) - f(x, y) \rightarrow \begin{bmatrix} -1 & 1 \\ 0 & 0 \end{bmatrix}$

- Vertical gradient mask: detects horizontal edges $\rightarrow \frac{\partial f}{\partial y} = f(x, y+1) - f(x, y) \rightarrow \begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix}$

- Magnitude gradient mask: nonlinear, non-negative $\rightarrow \text{mag}(\nabla f) = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$ = apply H and V masks, then operate.

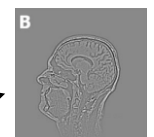
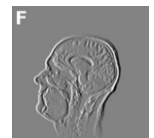
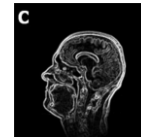
- Roberts cross operator: even, small, sensitive to noise, detect sharp edges in all directions $\rightarrow \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$ and $\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$

- Prewitt operator: uneven, unweighted, detect smooth orthogonal edges $\rightarrow \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$ and $\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$

- Sobel operator: uneven, weighted, detect smooth orthogonal edges $\rightarrow \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$ and $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$

- Kirsch operator: 8 first-derivative convolution masks, one for each direction.

- Laplacian mask: second derivative, double edges, sensitive to noise $\rightarrow \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$



Sharpening

Used to highlight fine details, deemphasize slowly varying areas.

- Unsharp masking: $f_{\text{sharp}} = f + \overbrace{(f - f_{\text{smooth}})}^{\text{edges}} = 2f - f_{\text{smooth}} \rightarrow 2 \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$

- High-boost filtering: $f_{\text{sharp}} = cf + f_{\text{edge}} \rightarrow c \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$

c increases contrast and intensity. Use $c = 1$ to preserve background intensity.



NON-LINEAR PROCESSING

Adaptive and diffusion filters facilitate image segmentation by reducing noise while preserving edges. Instead of convolving the whole image with a fixed kernel, these filters are able to adapt to local properties (contrast, texture, edges...) by adjusting their mask coefficients or using non-linear relationships.

Adaptive filtering

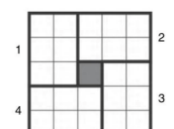
The kernel moves along the image like a convolution but, instead of acting equally on all the image, the filter changes its behavior according to properties of the local neighborhood. It may use some *if*, *for*, *max* or other non-linear algorithms.

- Median filter: chooses the median value, used to remove impulse (salt and pepper) noise ($\text{size} < k/2$) while preserving edges.

- Center-weighted median filter: gives more importance to the value of the central pixel, which is repeated w times before computing the median of all the values.

- Kuwahara filter: 4 non-overlapping regions are defined, the central pixel value is replaced by the average value of the pixels in the region with lowest variance. Bigger kernel size \rightarrow smoother.

- Linear Kuwahara (Fiji) filter: uses a straight (linear) kernel instead of rectangular. Can be applied to circular neighborhood which shows less directionality than a square one.



- **Adaptive bilateral filter:** convolution with a kernel h obtained by multiplication of two terms: spatial Gaussian (domain, affected by distance to center) and gradient Gaussian (range, affected by pixel values). $\xi(x,y)$ is an adaptive term based on non-linear algorithms ($f_{max/min/mean}$). On edges, the range causes a directional distortion of the domain: it elongates along the edge, causing smoothing along edge but not across it. Disadvantage: too many parameters.



$$h(x,y,m,n) = \underbrace{e^{-\frac{(x-m)^2+(y-n)^2}{2\sigma_d^2}}}_{\text{domain}} \underbrace{e^{-\frac{[f(x,y)-f(m,n)-\xi(x,y)]^2}{2\sigma_r^2}}}_{\text{range}}, \quad \xi(x,y) = \begin{cases} f_{max} - f(x,y) & \text{if } f(x,y) > f_{mean} \\ f_{min} - f(x,y) & \text{if } f(x,y) < f_{mean} \\ 0 & \text{if } f(x,y) = f_{mean} \end{cases}$$

The values of the kernel depend on:

- distance to the central pixel $x - m$ and $y - n$
- domain and range standard deviations σ_d and σ_r
- value of original image f
- maximum, minimum and mean values in the region f_{max} , f_{min} and f_{mean}

Diffusion filtering

Iterative process where the values of the original image f are modified by the image gradient ∇f of the previous iteration and a diffusion equation $D = g(\nabla f)$:

$$f_{new} = f + \partial_t f \quad \partial_t f = \text{div}(D \nabla f)$$

Classification of diffusion filters (DF):

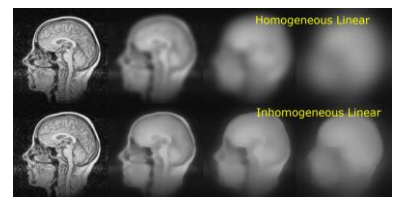
- Linear DF uses the original image to compute D , non-linear DF uses the image filtered in the previous iteration.
 - Linear requires more iterations to reach same result.
 - Non-linear requires more computation power per iteration.
- Homogeneous DF acts equally in all the image, inhomogeneous DF adapts to local features.
- Isotropic DF acts equally in all spatial directions, anisotropic DF adapts its properties to edge direction.
- $\uparrow D \rightarrow$ increases diffusion speed \rightarrow more smoothing.
- $\uparrow \Delta t \rightarrow$ increases the effect of diffusion \rightarrow more smoothing.

Types of diffusion filters, depending on the above characteristics:

- **Linear homogeneous isotropic:** equivalent to a Gaussian smoothing filter: blurs noise and edges equally everywhere and in all directions (non-adaptive).

$$D = 1 \rightarrow \partial_t f = \text{div}(\nabla f)$$

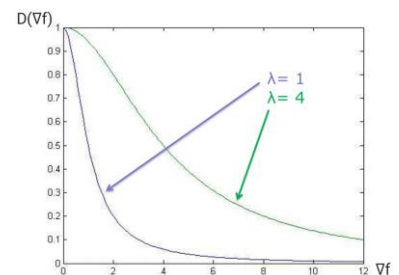
- **Linear inhomogeneous isotropic:** blurring is reduced where important features (edges) are detected (spatial adaptation). D is computed with the gradient of the original image ∇f_0 , and λ is a contrast parameter acting as a threshold for the gradient: edges are preserved if $|\nabla f| > \lambda$. Therefore, $\uparrow \lambda \rightarrow \uparrow D \rightarrow$ more smoothing.



$$D(\nabla f_0) = \frac{1}{\sqrt{1 + \frac{|\nabla f_0|^2}{\lambda^2}}} \rightarrow \partial_t f = \text{div}(D(\nabla f_0) \nabla f)$$

- **Non-linear inhomogeneous isotropic:** same as above, but in each iteration D is computed with the image filtered in the previous iteration (spatial and temporal adaptation). Disadvantage: lack of intra-edge smoothing. Ex.: Perona-Malik filter.

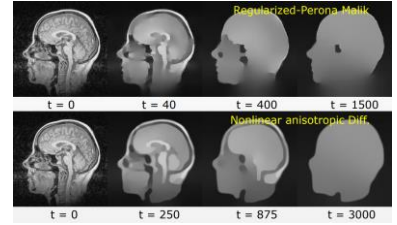
$$D(\nabla f) = \frac{1}{1 + \frac{|\nabla f|^2}{\lambda^2}} \rightarrow \partial_t f = \text{div}(D(\nabla f) \nabla f)$$



- **Regularized ~:** smooths the gradient before adaptation to avoid instability of the gradient.

$$f_{\sigma} = K_{\sigma} * f \rightarrow \partial_t f = \text{div}(D(\nabla f_{\sigma})\nabla f)$$

- **Non-linear inhomogeneous anisotropic:** diffusion is applied as a tensor (instead of a scalar) so it is defined differently in each direction (spatial, temporal and directional adaptation), which allows to preserve/enhance local features. Diffusion over edges does not occur, but diffusion parallel to edges is enabled. Intraregional instead of interregional smoothing.



- Eigenvectors (diffusion direction): $\begin{cases} \text{perpendicular to edge: } v_1 \parallel \nabla f \rightarrow v_1 = \frac{\nabla f}{|\nabla f|} \\ \text{tangential to edge: } v_2 \perp \nabla f \rightarrow v_2 = \begin{pmatrix} v_{1y} \\ -v_{1x} \end{pmatrix} \end{cases}$
- Eigenvalues (diffusion strength): $\begin{cases} \text{no diffusion (non-linearity): } \lambda_1 = D(\nabla f) \\ \text{strong diffusion: } \lambda_2 = 1 \end{cases}$

$$D = \begin{bmatrix} v_1 & v_2 \end{bmatrix} \cdot \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

Implementation in code of a 2D isotropic diffusion filter. D is computed for each orthogonal direction, multiplied by the gradient in that direction, divided by the squared distance ($\sqrt{\Delta x^2 + \Delta y^2}$ if diagonal directions) and summed. It is considered isotropic because D is ultimately applied as a scalar, not a tensor.

$$f_{new} = f + \Delta t \left(D_N \frac{\nabla f_N}{\Delta y^2} + D_S \frac{\nabla f_S}{\Delta y^2} + D_E \frac{\nabla f_E}{\Delta x^2} + D_W \frac{\nabla f_W}{\Delta x^2} \right)$$

Image Enhancement III: Frequency Filtering

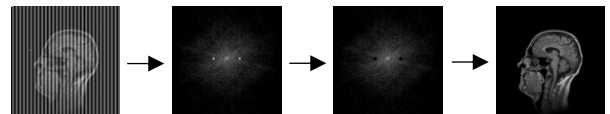
$H(u, v) = T[F(u, v)]$, where $F(u, v)$ is the Fourier transformation of the image (cycles/mm):

$$F(u, v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \cdot e^{-j2\pi(\frac{ux}{M} + \frac{vy}{N})} = |F(u, v)| e^{-j\theta(u, v)}$$

- **Magnitude** $|F(u, v)| = \sqrt{\text{Re}\{F(u, v)\}^2 + \text{Im}\{F(u, v)\}^2}$ and **phase** $\theta(u, v) = \tan^{-1} \frac{\text{Im}\{F(u, v)\}}{\text{Re}\{F(u, v)\}}$
- **Resolution** in Fourier domain: $\Delta u = \frac{1}{N\Delta x}$
- **Zero padding:** adding zeros to the spatial image (increasing N) so that resolution in Fourier domain Δu is improved.
- **Zero frequency:** frequency at the center of the Fourier image (mean value of $f(x, y)$): $F_0 = \frac{1}{MN} \sum f(x, y) \cdot e^{-j2\pi 0} = \frac{\sum f(x, y)}{MN} = \bar{f(x, y)}$

Notch filter

Remove one frequency only: $H(u, v) = \begin{cases} 0 & (u, v) = (M/2, N/2) \\ 1 & \text{otherwise} \end{cases}$



Low-pass filter (smoothing)

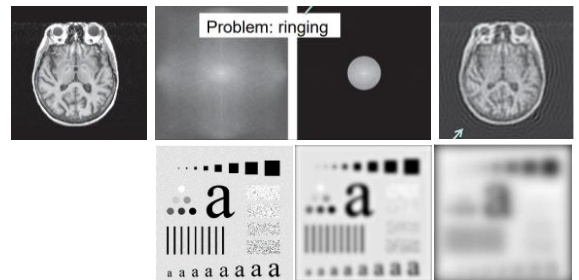
Frequencies near center are preserved, frequencies far from the center are removed. $D(u, v) = \sqrt{\left(u - \frac{M}{2}\right)^2 + \left(v - \frac{N}{2}\right)^2}$ is distance from center to (u, v) and D_0 is cut-off frequency.

- **Ideal LP filter:** sinc in SD, pulse in FD; produces ringing.

$$H(u, v) = \begin{cases} 1 & D(u, v) \leq D_0 \\ 0 & D(u, v) > D_0 \end{cases}$$

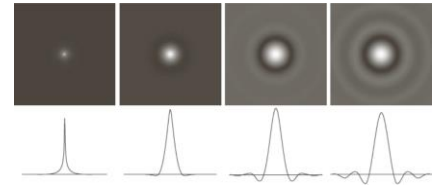
- **Gaussian LP filter:** Gaussian in SD and FD, no ringing.

$$H(u, v) = e^{-\frac{D^2(u, v)}{2D_0^2}}$$



- **Butterworth LP filter:** allows to control transition and ringing, n is the order.

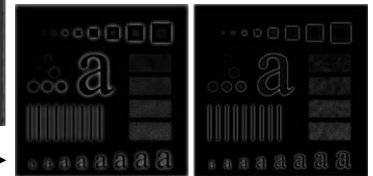
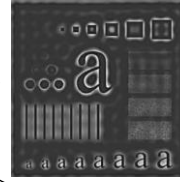
$$H(u, v) = \frac{1}{\sqrt{1 + \left(\frac{D(u, v)}{D_0}\right)^{2n}}}$$



High-pass filter (edge detection)

Frequencies near center are removed, frequencies far from the center are preserved. $H_{HP}(u, v) = 1 - H_{LP}(u, v)$.

- **Ideal HP filter** (ringing) $\rightarrow H(u, v) = \begin{cases} 0 & D(u, v) \leq D_0 \\ 1 & D(u, v) > D_0 \end{cases}$
- **Gaussian HP filter** $\rightarrow H(u, v) = 1 - e^{-\frac{D^2(u, v)}{2D_0^2}}$
- **Butterworth HP filter** $\rightarrow H(u, v) = \frac{1}{\sqrt{1 + \left(\frac{D_0}{D(u, v)}\right)^{2n}}}$
- **Laplacian filter** $\rightarrow H(u, v) = -D^2(u, v) = -\left[\left(u - \frac{M}{2}\right)^2 + \left(v - \frac{N}{2}\right)^2\right]$. Only edges, $H(0, 0) = 0$.



High-frequency emphasis filter (sharpening)

Edges are emphasized. Analogous to **sharpening** techniques in spatial filtering.

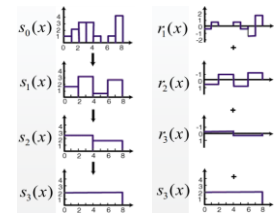
- **Unsharp masking** $\rightarrow H(u, v) = 2 - H_{LP}(u, v)$
- **High-boost filtering** $\rightarrow H(u, v) = A + H_{HP}(u, v)$

Image Enhancement IV: Multiresolution Wavelets

Haar wavelet transform

A signal is successively decomposed into scaling (s_i) and wavelet (r_i) functions, which can then be used to reconstruct original signal (s_0). Somehow similar to Fourier: decompose a temporal/spatial signal into sines and cosines in *frequency* domain.

- **Haar scaling function:** down-sampled version of the signal containing overall features. Obtained with low-pass filter or by averaging each pair of data points $\rightarrow s_i = \text{downsample}(s_{i-1})$
- **Haar wavelet function:** residual version of the signal containing details. Obtained with high-pass filter or by subtracting scaling function to the original signal $\rightarrow r_i = s_{i-1} - s_i$

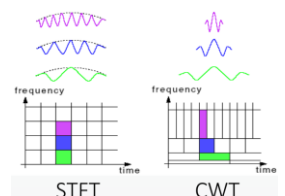
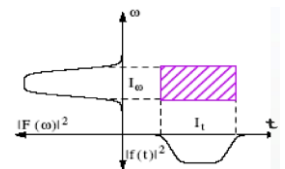


Each scaling function has half the resolution of the previous one (Haar pyramid), which occupies half the storage, until it becomes a single data point (average value of the original signal). Each wavelet function is made of *basis functions* (square-shaped sequence called wavelet) that can be linearly combined to represent any function. Original signal can be reconstructed by summing all residuals and the last scaling function. N is the number of iterations.

$$s_0 = s_N + \sum_{i=1}^N r_i$$

Time vs. frequency

- **Heisenberg principle:** short function in time domain corresponds to wide band of frequencies in frequency domain, while a narrow band of frequencies in Fourier domain leads to a function spread out in time domain. Therefore, a signal can't be concentrated in both time and frequency domains (there is a lower bound on time-frequency product).
- **Time-frequency atom:** extension of a wavelet in time and frequency domains. A signal is usually sampled in short time domain (of infinite frequencies) and its Fourier transform is localized in a narrow band of frequencies (of infinite time).
- **Time and frequency tiling:** time-frequency atoms are represented as windows (boxes) tiled on top of each other. Different transforms lead to different representations.
 - **Short-time Fourier transform (STFT):** all frequencies (high and low) have same box size.
 - **Continuous wavelet transform (CWT):** lower frequencies have wider windows and high frequencies have narrower ones. Due to Heisenberg principle, narrow windows are also taller.

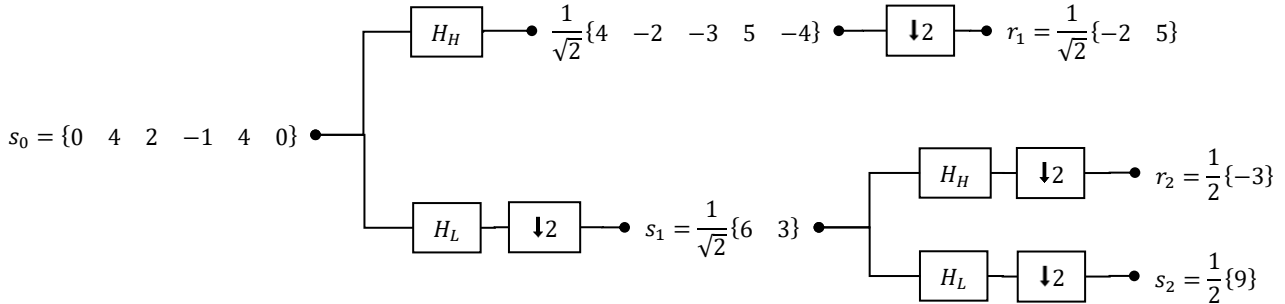


Wavelet transform in 1D

Can be achieved following Haar scaling and wavelet functions definitions, or by convolving with filters (filterbank).

- **Haar filterbank:** implementation of wavelet transform through a set of low-pass (returns approximation coefficients) and high-pass (details coefficients) filter convolutions. Input signal (s_i) needs to be zero-padded and filters (H) need to be flipped before convolving them. Downsampling ($\downarrow 2$) is done by choosing even positions.

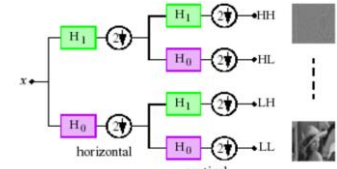
Original signal: $s_0 = \{4 \ 2 \ -1 \ 4\}$ High-pass filter: $H_H = \left\{ \frac{1}{\sqrt{2}} \ -\frac{1}{\sqrt{2}} \right\}$ Low-pass filter: $H_L = \left\{ \frac{1}{\sqrt{2}} \ \frac{1}{\sqrt{2}} \right\}$



Reconstruction of a signal is made with *inverse* Haar filterbank from its residuals and last scaling function, which are upsampled (with zeros), zero padded, convolved by the inverse LP and HP filters and summed, following the inverse path.

Wavelet transform in 2D

Wavelet transform is separated into two levels (horizontal and vertical filtering) where a low-pass (L) and high-pass (H) filters are applied, followed by downsampling. Original image is decomposed in one approximation (scaled) image (LL) and three residual images containing horizontal (HL), vertical (LH) and diagonal (HH) details. Filters can be implemented through pyramids:



- **Approximation pyramid:** a 2x2 block average filter (averaging each four pixels) is used to downscale the image, until one single value is obtained. Dimensions of original image need to be multiple of 2.

$$s_0 = \begin{bmatrix} 5 & 5 & 3 & 4 \\ 5 & 5 & 2 & 1 \\ 2 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 \end{bmatrix} \rightarrow s_1 = \begin{bmatrix} 5 & 2.5 \\ 1 & 0.5 \end{bmatrix} \rightarrow s_2 = [2.25]$$

- **Residual pyramid:** computed by subtracting scaled image from the previous one. Scaled image needs to be upsampled for the operation.

$$r_1 = s_0 - s_1 = \begin{bmatrix} 5 & 5 & 3 & 4 \\ 5 & 5 & 2 & 1 \\ 2 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 5 & 5 & 2.5 & 2.5 \\ 5 & 5 & 2.5 & 2.5 \\ 1 & 1 & 0.5 & 0.5 \\ 1 & 1 & 0.5 & 0.5 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0.5 & 1.5 \\ 0 & 0 & -0.5 & -1.5 \\ 1 & -1 & 0.5 & -0.5 \\ -1 & 1 & -0.5 & 0.5 \end{bmatrix}$$

$$r_2 = s_1 - s_2 = \begin{bmatrix} 5 & 2.5 \\ 1 & 0.5 \end{bmatrix} - \begin{bmatrix} 2.25 & 2.25 \\ 2.25 & 2.25 \end{bmatrix} = \begin{bmatrix} 2.75 & 0.25 \\ -1.25 & -1.75 \end{bmatrix}$$

Reconstruction of original image is made summing residuals and last scaling image, upsampled to original size. Applications:

- **Multiscale registration:** registration done with downsampled images, improves robustness (from global to local) and computational speed (see [multiresolution scheme](#)).
- **Donoho algorithm** (noise reduction): before reconstruction, wavelet coefficients in residual images (which contain noise) are set to zero according to some threshold (*wavelet coefficient shrinking*).
- **Wavelet-based edge detector:** edges (and their directionality) are easily detected in residual images.
- **Transform-based codification:** wavelets improve quality in image compression (JPEG vs. JPEG 200).
- **Convolutional neural networks:** wavelet decomposition provides useful features that can be used in deep learning algorithms.

Image Segmentation

To separate a region of interest (ROI) from background. Quantify area or volume, compare, detect objects or tissues, localize activity.


MANUAL SEGMENTATION

Precise, used to validate automatic methods, but slow, expensive and low reproducibility.

INTENSITY-BASED SEGMENTATION

Separation of ROI (foreground) from background based on pixel intensity.

Thresholding

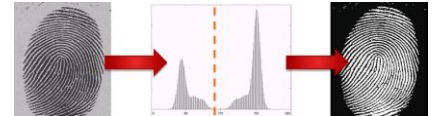
Thresholded image $g(x, y) = \begin{cases} 1 & f(x, y) > T \\ 0 & f(x, y) \leq T \end{cases}$ 

Different methods to find T . Segmentation is affected by noise if it is added to the ROI (solved by pre-smoothing the image) and uneven illumination (solved with local thresholding).

- **Basic global thresholding**: compute overall mean value $\mu_0 = T_0$ which separates histogram in two groups ($G_1 \leq T_0$ and $G_2 > T_0$), compute mean of each group μ_1 and μ_2 and compute threshold: $T_1 = \frac{\mu_1 + \mu_2}{2}$. Process is repeated until $|T_{i+1} - T_i| < T_\infty$ (predefined limit). Good for suitable histograms with two separate regions (bimodal).

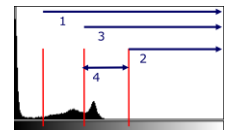
- **Otsu's method**: advanced global thresholding. Steps to compute the within-class variance for a certain threshold t (the value at t is included in the first class):

1. Weight of each class (w_1 and w_2): $w = \sum p(i)$
2. Marginal mean of each class (μ_1 and μ_2): $\mu = \frac{\sum p(i)x(i)}{w}$
3. Variance of each class (σ_1^2 and σ_2^2): $\sigma^2 = \sum x(i)^2 p(i) - \mu^2$
4. Within-class variance: $\sigma_w^2(t) = w_1 \sigma_1^2 + w_2 \sigma_2^2$
5. After computing σ_w^2 for all possible thresholds t , we choose the smallest: $T = t | \sigma_{w_{min}}^2$



Steps to compute the between-class variance for a certain threshold t (faster than σ_w^2):

1. Compute weights and marginal means of each class.
2. Between-class variance: $\sigma_b^2(t) = w_1 w_2 (\mu_1 - \mu_2)^2$
3. After computing σ_b^2 for all possible thresholds t , we choose the largest: $T = t | \sigma_{b_{max}}^2$



- **Multiple global thresholding**: obtain several ROIs (apply Otsu's method iteratively).
- **Multivariable thresholding**: several variables in histogram (intensity, RGB, multimodality images...).
- **Basic adaptive thresholding**: divide image in sub-images and segment each one individually (obtaining Otsu's threshold in each one). Solves uneven illumination problem. Example: moving averages.



Region growing

One or several pixels (seeds) are manually or automatically selected and neighboring pixels are examined and added to the growing region if they are sufficiently similar (i.e., similar intensity). Conditions can be changed to update the region. Generates connected regions, easy and fast, but does not work if intensity varies within the ROI (noises, uneven illumination).

EDGE-BASED SEGMENTATION

ROIs are separated by edges, which can be detected by discontinuity in some property (depth, orientation, illumination...) → gradient of intensity (edges).

Basic edge extraction

Fast and automatic, but sensitive to noise, needs post-processing (edge linking to close objects) and choosing adequate thresholds is difficult. Steps: image smoothing (noise reduction), **edge points** detection (potential edges) and **true edge** localization.

- **Marr-Hildreth** or Laplacian of a Gaussian (Mexican hat): use Gaussian to smooth and Laplacian to detect **edge points** (double edges).
- **Gradient of a Gaussian**: use Gaussian to smooth, gradient to detect **edge points**.

$$S = \nabla(g * I) = (\nabla g) * I = \begin{bmatrix} g_x \\ g_y \end{bmatrix} * I, \quad g = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- **Thresholded gradient of a Gaussian**: use Gaussian to smooth, gradient to detect **edge points** and threshold to localize **true edges** (thick edges).



- **Canny edge detector:** an optimal edge detector should find all edges (low error rate), as close to the real ones as possible and return a single point per edge (no thick or double edges, and no isolated pixels). Steps:

1. Gaussian to smooth image.
2. Gradient magnitude (find **edge points**) and gradient angle (find edge direction, perpendicular to the actual edge, approximated to 0°, 45°, 90° or 135°).



$$G = \sqrt{G_x^2 + G_y^2} \quad , \quad \theta = \arctan\left(\frac{G_y}{G_x}\right)_{rounded}$$

3. *Non-maxima suppression* to each **edge point**: it is **true peak** only if higher than the two neighbor pixels in edge direction (so that the resulting edge is one pixel wide).
4. *Hysteresis* (double) thresholding to each **true peak**: it is **true edge** only if at least one of the below conditions is met. Connected to **true edge** can be either contiguously or through other **true peaks**.

$$\begin{cases} \textcircled{1} & value > T_H \\ \textcircled{2} & value > T_L \text{ \& connected to true edge} \end{cases}$$

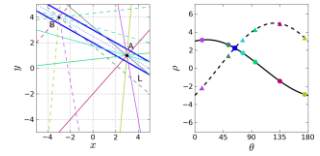
Hough transform

Detects and quantifies shapes (straight lines, circles, ellipses, polygons...) in a binary edge image (Canny output), even in the presence of noise or incomplete outlines. The (x, y) space is transformed into a new parametric space. Disadvantage: only works with predefined shapes.

- **Line detection (line):** each point (x_i, y_i) defines a *straight line* in the new (m, b) space. Points lying on the same line in the (x, y) space define lines in the new space which all intersect at the same point. Problem: for vertical lines, $m = \infty$.

$$y(x) = mx + b \rightarrow b(m) = -xm + y$$

- **Line detection (sine):** each point (x_i, y_i) defines a *sinusoidal curve* in the new (θ, ρ) space. Points lying on the same straight line in the (x, y) space define sinusoidal lines in the new space which all intersect at the same point. In the original space, straight lines are referenced by their point closest to the origin, so that ρ is the shortest distance from the origin to the line (normal to the line) and θ is the angle of that normal with respect to the x-axis.



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

The new space (θ, ρ) is quantized and each edge pixel is represented there. Edges are detected by finding local maxima in the new sinusoidal space, which can be converted back to the (x, y) space: $m = -\frac{\cos \theta}{\sin \theta}$ and $b = \frac{\rho}{\sin \theta}$. *Finer* quantization improves accuracy but increases computation time, while *coarser* quantization is more tolerant to noise.

- **Circle detection:** each point (x_i, y_i) defines a *cone* in the new (a, b, r) 3D space. Points lying on the same circumference in the (x, y) space define 3D cones in the new space which all intersect at the same point. r can be fixed so that a (a, b) 2D space is obtained, where only a and b are searched for.

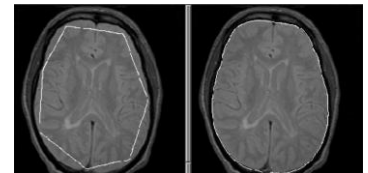


$$r^2 = (x - a)^2 + (y - b)^2$$

Active contours

Active contours or snakes are continuous, deformable contours that enclose a target object by locking on to its edges. Numerical concept of an elastic rubber band subjected to image-dependent forces. Used to delineate features interactively (contour adapts when image changes), maintains a closed shape even if features are disturbed by noise.

Snake is represented by control points, which are manually initialized and then processed: snake is subjected to constraints defined as energies, which make it evolve to reduce (minimize) its energy. An appropriate energy function makes the snake detect objects in the image and have particular properties. Different initial conditions lead to different results. Affected by gradient (solved with previous smoothing).



$$E_{snake} = E_{int} + E_{img} + E_{con}$$

Internal energy (E_{int}) represents smoothness along the snake, has two components: resisting to stretching and bending; image energy (E_{img}) guides the snake towards the desired image property (following gradient to edges); constraint energy (E_{con}) accounts for user-defined limitations (properties we know about the target object).

- **Balloon energy:** constant constraint normal to the curve, forces the snake to “blow up” and stop when reaching an edge. No need to initialize close to the final solution.

MATHEMATICAL MORPHOLOGY

Use a structuring element to extract or modify information about the shape of the object. Useful to post-process binary images resulting from segmentation and to obtain topological descriptors (used in AI, see Feature extraction). Also available for grayscale images.

Basic morphology

A kernel shaped as structuring element (square, disk...) is superimposed centered on top of each pixel of the input image (similar to convolution) but the operation is *logical* rather than using weights and multiplying.

- **Dilation:** increases the size of foreground object: the centered pixel becomes foreground if any pixel within the kernel coincides with a foreground pixel. Object is dilated by k pixels (disk kernel of radius k) or $(k - 1)/2$ pixels (square kernel of size k). Disk is the most used structure element since objects are usually circular in real life. Used to round or modify corners (with circle or cross), get edges (edges = dilated – original) and add some extra pixels around a bone or tumor mask.
- **Erosion:** decreases the size of foreground object (equivalent to dilating background). Pixels are set to background when kernel coincides with background. Used to remove thin lines or details (smaller than structuring element) and also get edges (edges = original – erosion).
- **Opening:** erosion + dilation with same element: contours are smoothed, narrow isthmuses broken and thin protrusions eliminated. Objects are removed if dissimilar shape or smaller than structuring element.
- **Closing:** dilation + erosion with same element: contours are smoothed, narrow gaps connected and holes in foreground are eliminated if dissimilar or smaller than structuring element.



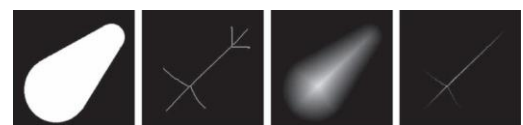
Advanced morphology

- **Hit-or-miss transform:** structuring element with 0 (background), 1 (foreground) and X (any), the centered pixel becomes true (foreground) if the kernel pattern fully coincides with the image. Used to find specific features (corners, edges...) and in the algorithms below.

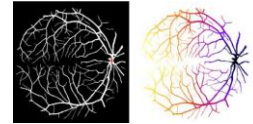
Bottom-left corner		
X	1	X
0	1	1
0	0	X

Top edge		
0	0	0
X	1	X
1	1	1

Top-right branch		
0	0	0
0	1	0
X	X	0
- **Thinning:** successively erodes foreground boundary while preserving boundary end points. Uses hit-or-miss transform to detect edges and subtract them from the original image $\rightarrow I_{thin} = I - I_{hit-or-miss}$
- **Thickening:** dual of thinning, equivalent to thinning background.
- **Skeletonization:** repeated thinning until *center line* is obtained (corners are also subtracted to obtain single-pixel wide skeleton of the object). Two steps: ① pixels are marked as candidates for removal (conditional thinning) and ② candidates are removed if it does not destroy connectivity. Skeleton provides a simple and compact representation with information on object length (measuring maximally separated pair of end points) and shape (counting junction points, where at least 3 branches meet). Used for angiogram, handwriting recognition.
 - **Pruning:** skeleton is post-processed by removing spurs (residual branches caused by irregularities in original boundary). Limited number of iterations (convergence removes all open-loop pixels).
 - Noise and small changes in the boundary greatly affect the output skeleton. Opening or closing prior to skeletonization can fix it.
- **Distance transform:** grayscale shape (original object's shape) where intensity of each pixel corresponds to its distance to nearest background pixel. Obtained with multiple, repeated erosions of increasing size until all foreground pixels disappear and labeling each pixel with number of erosions performed before it disappeared. Represents topology of the object, can be used in registration and AI algorithms.
- **Medial axis transform:** grayscale skeleton where intensity of each pixel represents its distance to a boundary in the original object. Can be obtained by thresholding or finding *ridges* (local maxima) in distance transform. Can be used to reconstruct original object.

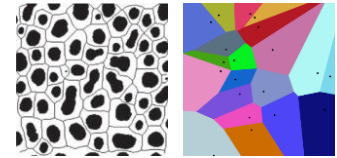


- **Geodesic distance transform:** grayscale mask where intensity of each pixel represents the distance to some marker in the mask, computed by constraining propagation of distance within the mask. Used to find distance between two points in vasculature network.



- **Skiz** (skeleton of influence zone): skeletonization of background, creates regions around each feature so that all points in the region are *closest* to that object.

- **Voronoi diagram:** skiz made around seeds (points), regions are polygons. Dual of **Delaunay triangulation**: the set of centroid points in Voronoi diagram are the vertices in Delaunay triangulation, and the circumcenters of Delaunay triangles are the vertices in Voronoi diagram.



- **Morphological** (or geodesic) **reconstruction:** a marker image with seeds is processed with *conditional* dilations or erosions, combined with a mask image that restricts markers' propagation using logical operations. Process repeated until *idempotence*: marker image is not further changed, meaning the markers completely filled the chosen region. Seeds can be set at the image border to:



- Select (and remove) objects touching the borders by using original image as mask.
 - Select (and fill) holes within particles by using complement image as mask, then combining the result with the mask and inverting.



- **Chamfer distance:** way of calculating distances by approximating Euclidean (decimal) distances with integer weights, simpler to compute. Examples: chessboard, city-block, Borgefors, chess knight.

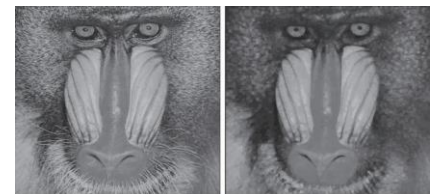


Grayscale morphology

The algorithm operates on a grayscale image, instead of a binary one. Output is also grayscale image.

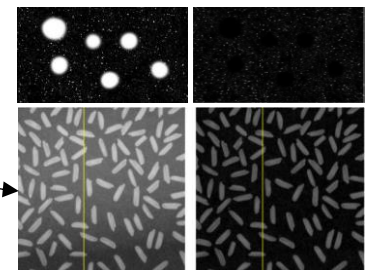
- **Grayscale dilation (erosion):** structuring element contains a pattern of 1s and 0s, is superimposed on input image, and the output value is the maximum (minimum) of pixel values coinciding with a 1. Dilation brightens and expands bright areas, while erosion darkens and expands dark areas.
- **Median filter:** it chooses the median value from the pixels selected (see in [Adaptive filtering](#)).

- **Grayscale opening (closing):** grayscale erosion (dilation) followed by grayscale dilation (erosion). Bright (dark) features smaller than structuring element are greatly reduced (increased) in intensity, while larger features remain unchanged.



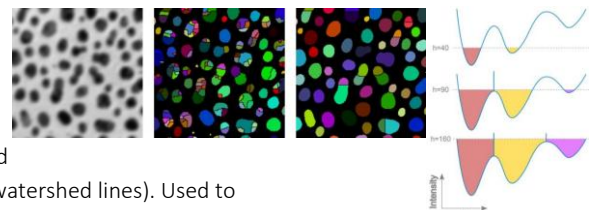
- **Grayscale morphological reconstruction:** used to select specific structures (conditional grayscale dilation) or remove objects touching the border (by planting the marker seed at border).

- **Top-hat transform:** subtracts the result of grayscale opening to the original image. Removes big, bright objects while keeping bright objects smaller than structuring element. When used with a large structuring element (bigger than target objects), it homogenizes the background making it easier to segment bright objects.



- **Regional maxima:** region of pixels with same value whose neighbors all have smaller value. Can be extended to neighboring pixels that have also high values (extended maxima). Analogously with regional minima.

- **Watershed segmentation:** image is turned into topographic surface, where pixel intensities represent elevation information with dark regions corresponding to "deep" catchment basins and bright regions to "high" watershed lines. Water sources are placed in regional minima (bottom of each basin), basins are flooded and "dams" are built where different water sources meet (top of watershed lines). Used to separate and label adjacent objects with different intensities. Noisy images with many regional minima will lead to over-segmentation. Solution: blur original image (Gaussian filter) or merge similar regions in output image.



SEGMENTATION EVALUATION

The contours of two binary segmentations S_1 and S_2 are compared. Manual segmentation is usually the gold standard.

- **Dice coefficient:** $D = \frac{2|S_1 \cap S_2|}{|S_1| + |S_2|}$
- **Jaccard index** or **intersection over union** (IoU): $J = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$

In code, $|S|$ is implemented as $\text{sum}(S)$. Indicators D and J are similar: if $S_1 = S_2 \rightarrow D = J = 1$, and if S_1 and S_2 do not coincide at all, then $D = J = 0$. Both can be related:

$$J = \frac{D}{2 - D} \quad D = \frac{2J}{1 + J}$$

- **Maximum Hausdorff distance:** maximum of all distances from a point in one set to the closest point in the other set.
 1. Compute distances from each point in S_1 to the closest point in $S_2 \rightarrow d_1 = d(S_1, S_2)$
 2. Compute distances from each point in S_2 to the closest point in $S_1 \rightarrow d_2 = d(S_2, S_1) \neq d(S_1, S_2)$
 3. Find the maximum of all those distances $\rightarrow d_H(S_1, S_2) = \max(d_1, d_2)$

Image Registration

Combining different images into a single one may be useful to see all complementary information (anatomical + functional) or to generate quantitative data (subtraction, quantification...). However, 3D volumes may not match due to different patient position, time, acquisition device or even modality. First, images need to be *aligned* (registered). To register A (fixed image) and B (moving image), a geometric transformation T and interpolation are applied to B in order to obtain B^T , which is then compared to A . Applications:

- **Neurosurgery:** in deep brain stimulation, MR is used to locate target tissue and CT with a Leksell or stereotactic frame attached to skull is used to provide the fiducials. After registration, the target tissue can be localized in reference to the fiducials.
- **Multimodal quantification:** images from different modalities, which provide different types of information, are integrated to obtain a more complete understanding of the patient's situation.
- **Treatment-resistant epilepsy:** MR (anatomy) + PET (activity) used to locate epileptic tissue.
- **Liver hemangiomas:** MR allows tissue visualization while SPECT indicates if tumor is benign or not.
- **Small animal imaging:** CT/MR or CT/PET + time.

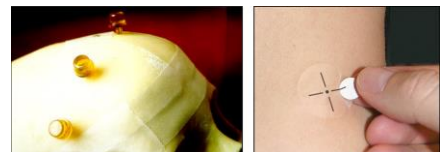
FEATURE-BASED REGISTRATION

Features extracted from the images are used to compute the rigid transformation matrix T that best aligns B to A .

Homologous points

The same numbered list of points is used in both images, so that each point p_i in image B has its homologous point p'_i in image A . Points can be of two types:

- **Anatomical landmarks:** points in the anatomy of the patient (center of mass, bifurcations, corners...) are used as reference. Not useful in non-anatomical modalities (PET, SPECT).
- **External fiducials:** artificial markers attached to the patient are used as reference. The smaller the marker, the more accurate but also difficult to find. Can contain something to make them easily identifiable in the image. Two types of markers:
 - Skin adhesive: non-invasive, easy to place and remove, numerous, but may detach and move with the skin.
 - Bone mounted: precise, fixed during surgery, but intrusive and can only be placed when surgery is required.



The *source* point cloud p_i is transformed into p''_i , which is as close as possible to the *target* point cloud p'_i . The solution is guaranteed to converge, independently of the starting point.

- **Least squares formulation:** finds the best transform for the abovementioned task by minimizing the error. Three steps:
 1. Translation to origin: points in each cloud are referenced to the centroid C_p of their respective clouds.
 2. Rotation: inertial axes of both clouds are found and aligned with rotation matrix R . Since there is no exact solution, the problem is minimized to obtain \hat{R} .
 3. Translation to target centroid: the rotated source cloud is moved with translation matrix T to make it coincide with the target cloud. Since there is no exact solution, \hat{T} is obtained from \hat{R} .

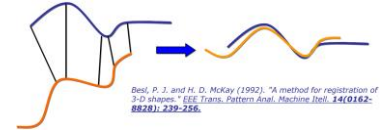
$$\left. \begin{array}{l} \text{source} \rightarrow C_p = \frac{1}{N} \sum_{i=1}^N p_i \rightarrow q_i = p_i - C_p \\ \text{target} \rightarrow C_{p'} = \frac{1}{N} \sum_{i=1}^N p'_i \rightarrow q'_i = p'_i - C_{p'} \end{array} \right\} \begin{array}{l} \textcircled{2} \quad q'_i = Rq_i \rightarrow \text{minimize } \sum |q'_i - Rq_i|^2 \rightarrow \hat{R} \\ \textcircled{3} \quad p'_i = Rp_i + T \rightarrow \hat{T} = C_{p'} - \hat{R}C_p \end{array} \rightarrow p''_i = \hat{R}p_i + \hat{T}$$

Final solution could be further minimized, but alignment between transformed points p''_i and target points p'_i is never perfect due to errors when selecting the landmarks (fiducial localization error), which leads to:

- **Fiducial registration error:** constant error of each point, defined as $\varepsilon = p''_i - p'_i$.
- **Target registration error:** depends on the distance from target object to each landmark (further to the landmark \rightarrow bigger error). Fiducials should surround the ROI homogeneously to reduce TRE.

Non-homologous points

Image A contains the target model and image B either a set of points (points-to-model) or another model (model-to-model). Since points are not homologous, there may be several solutions that depend on initialization (initial shape, position...).



- **Iterative closest point:** every point in B is associated to the closest one in A . The rigid transform is found and the error (sum of squared differences) between A and B^T is calculated. The process is repeated until the error is minimized. No a priori knowledge required.

INTENSITY-BASED REGISTRATION

Transformation matrix T is computed iteratively. In each iteration, B is transformed and interpolated into B^T , and pixel values in A and B^T are used to compute and evaluate a *cost function*, which measures similarity between both images and is optimum (maximum or minimum) when images are best aligned. Optimization made with the gradient, which tells how much and in which direction to tune parameters. A stopping criterion determines when the cost function is sufficiently optimized. No pre-segmentation needed.

Cost function depends on as many parameters as T has, and it should: have max (or min) value when images are registered, not have local max (or min), not be affected by noise, and be computed fast (voxels \times evaluations \times parameters).

Intramodality registration

To compare images taken at different times, to check anatomy or pathology changes (tumor growth). Easy to implement.

- **Sum of absolute differences:** optimized at minimum.

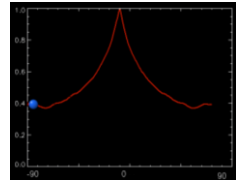
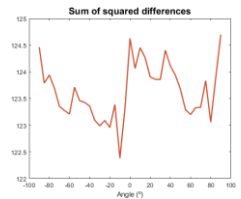
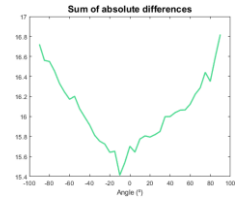
$$SAD = \frac{1}{N} \sum |A(x_A) - B^T(x_A)|$$

- **Sum of squared differences:** works best if images are very similar but is affected by high differences (tumor). Needs normalization. Optimized at minimum.

$$SSD = \frac{1}{N} \sum |A(x_A) - B^T(x_A)|^2$$

- **Normalized cross correlation:** assumes linear relation between intensity values. Doesn't need normalization. Optimized at maximum.

$$C = \frac{1}{N-1} \sum \frac{(A(x_A) - \mu_A) \cdot (B^T(x_A) - \mu_{B^T})}{\sigma_A \cdot \sigma_B}$$



Intermodality registration

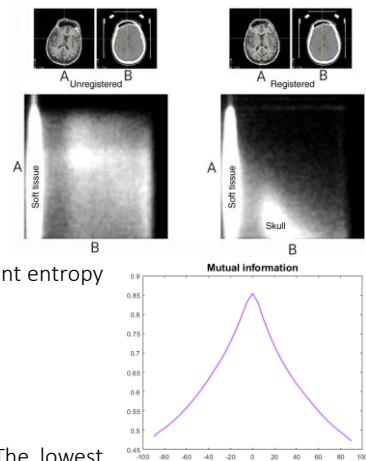
Not so easy to implement because there is no simple relation between intensities of different modalities.

- **Intermodality similarity measures:** estimates a mapping function to compare intensities of different modalities. Modality dependent.
- **Registration of intensity ridges:** extracts features (edges) in both images and matches them. Modality dependent.
- **Information theory:** maximizes the amount of shared information in two images. Independent of image modality and intensity correlation. Amount of information is measured by joint entropy $H(A, B)$. The less independent (more similar) the images are, the lower their joint entropy. $P_A(v_i)$ and $P_{AB}(v_i, v_j)$ are the normalized *histogram* and *joint histogram*.

$$H(A) = - \sum_i P_A(v_i) \cdot \log_2 P_A(v_i) \quad H(A, B) = - \sum_i \sum_j P_{AB}(v_i, v_j) \cdot \log_2 P_{AB}(v_i, v_j)$$

- **Joint histogram:** 2D matrix combining the distribution of pixel intensities across two images A and B . Each axis contains the intensity range of one image (similar to a regular **histogram**), and the bid in position (i, j) of the matrix contains the amount of pixel pairs (that share same position in both images) with value v_i in image A and value v_j in image B . If the two images are identical, then the matrix only contains values on main diagonal.
- **Mutual information:** stable cost function for intermodal registration. Measures how much of one image is explained by the other. Information is maximum (and joint entropy is minimum) when images are registered.

$$MI = H(A) + H(B) - H(A, B) = H(A) - H(A|B)$$



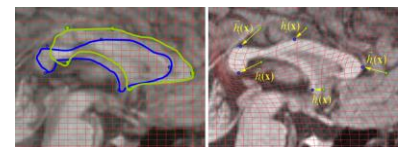
Optimization methods: Powell, Simplex.... Ways to improve the optimization process:

- **Multiresolution scheme:** images are downsampled to different levels of resolution. The lowest resolution images (first level) are registered and the output initializes (serves as starting point of) the next level. Registration is done from global to local and the process is faster (uses less voxels in the first iterations) and more robust (avoids local minima).
- **Capture range:** interval of parameter values that can be used in the transformation, where solution can converge to. It may avoid converging to local minima and instead focuses the algorithm on the range around the correct solution.

NON-LINEAR REGISTRATION

Uses non-linear transformations to further register images, usually after linear registration. Local differences can be adjusted by using different transformations for every pixel (hundreds of parameters, a lot of computations). Necessary in motion compensation (breathing), atlas-based analysis (compare subject to atlas) or inter-subject quantification (in research).

- **Free form deformations:** a mesh of *control points* is defined, which can be displaced through the transformation. Manipulating the mesh allows the deformation of the underlying image. Transformation of the pixels between control points is obtained by B-spline interpolation. The more control points, the more deformation.



Regularization avoids wrong deformations by imposing constraints on the transformation.

- **Bayesian formulation:** favors smooth deformations by minimizing the *bending energy*. The images need to be pre-registered with linear methods to reduce the need for strong deformations.

$$E_{bend} = \underbrace{\gamma C(A, B^T)}_{\text{similarity criteria}} - \alpha \cdot \text{Deformation}$$

REGISTRATION EVALUATION

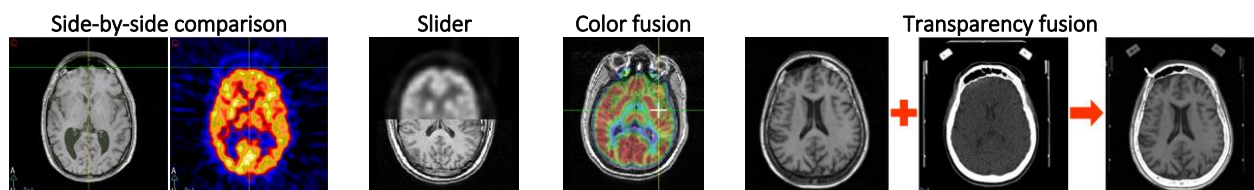
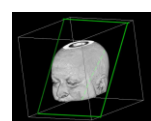
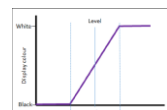


Image Visualization

2D VISUALIZATION

- **Window level and width:** used to increase contrast on specific intensities when visualizing on a screen, without modifying original values. Intensity values outside of the window take the maximum/minimum value of the window. Level determines the center value of the window: \uparrow level \rightarrow focuses contrast on high (white) intensities, low values are all black. \uparrow Width \rightarrow more intensities included \rightarrow less contrast.
- **Multiplanar reconstruction (MPR):** 3D volumes are reconstructed from 2D images by stacking the slices. After that, any view from any angle (axial, sagittal, coronal, oblique...) can be reconstructed through MPR.



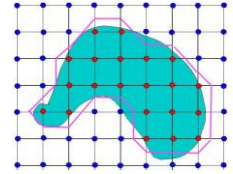
3D VISUALIZATION

3D data is rendered and projected in 2D so that we can see the 3D volume in a flat screen.

Surface rendering

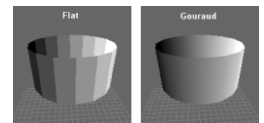
Geometrical surface of the object to be rendered is created with a triangular mesh. Fast, the result is hollow (no voxels). Always needs previous segmentation (intensity- or edge-based) that gives a list of coordinate points used to create the mesh. Tiling problem: triangles (tiles) need to be joint (tiled) to form a surface.

- **Delaunay triangulation:** the set of points are used as vertices of the triangles, which are computed in a way that no vertex lies inside the circumcircle of any triangle in the set. Dual of [Voronoi diagram](#).
- **Marching cubes algorithm:** the set of points define an *iso-surface* separating the segmented object from the rest. The volume is divided into a grid of cubes, and vertices of the grid correspond to voxels that either belong or not to the segmentation. For cubes having vertices both in and out of the segmentation, the iso-surface must intersect their edges between inside and outside vertices. In 3D, there are 2^8 possible configurations, but only 14 unique triangulations (considering rotations and mirroring). Smaller cubes → better resolution.



The resulting triangular mesh is projected into 2D, additional processing is needed to remove hidden surfaces and for shading.

- **Backface culling:** backfacing surfaces (not facing the viewer) are hidden. A tile is backfacing if its normal \vec{N} is facing away from the viewing direction $\vec{V} \rightarrow \vec{N} \cdot \vec{V} = \cos(\alpha) > 0$
- **Flat shading:** shading value depends on angle between \vec{N} and \vec{V} , and the distance to viewer. Every tile is colored and then illuminated, so that each tile presents only one shading value.
- **Gouraud shading:** lightning at the vertices of each tile are computed, and then the illumination (color) of each pixel in the tile surface is linearly interpolated. Achieves continuous lightning on the object.

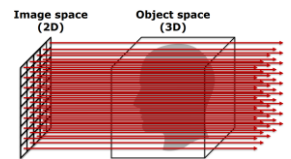


Further improvements can be made to the triangular mesh: smoothing, sharpening... Example of application: virtual endoscopy.

Volume rendering

Entire 3D volume is projected into a 2D plane, so that every voxel partially contributes to the final result. Pre-segmentation not needed but it is often used. Realistic results, but slow since visualization needs to be re-rendered every time the view plane changes.

- **Ray casting:** a 2D array of rays is “casted” from the viewer and through the object, returning a value if they interact with it (or black if they don’t hit the object). Each ray interacts with many voxels, which are weighted differently to achieve the desired rendering. Output value V_{out} will be a function of each voxel value v_i and opacity α_i . Segmentation can be used before to assign specific opacity values to different tissues (pre-configured tables available).



$$V_{out} = V_{in}(1 - \alpha_{in}) + \sum_{i=1}^N v_i \alpha_i$$

- **Maximum intensity projection (MIP):** output value is the maximum of all voxels. Useful to detect implants (CT) or contrast agents (PET/SPECT).

$$V_{MIP} = \max(v_1, v_2, \dots, v_N)$$

- **Digitally reconstructed radiography (DRR):** output value is the sum of all voxels. Used with CT, it is equivalent to taking an X-ray since Hounsfield units are related to X-ray attenuation.

$$V_{DRR} = v_1 + v_2 + \dots + v_N$$



Image Compression

COMPRESSION PROPERTIES

- **Compression ratio:** $C_R = \frac{n_1}{n_2}$, where n_i is the number of information-carrying units. Data is maximally compressed when $C_R = 1$.
- Relative data **redundancy:** $R_D = 1 - \frac{1}{C_R}$. There is no redundancy when $C_R = 1$.
- **Coding redundancy:** related to the codes used to express information. For an image with L intensity levels (different values), each with probability $p(v_i)$ and represented with $l(v_i)$ pixels: $L_{avg} = \sum p(v_i)l(v_i)$ is the average number of bits used per pixel. Usually, $L_{avg} = n$ (all values use n bits) but we can code some values with $> n$ bits and others with $< n$ bits. If we assign high-probable values (appear more) to $< n$ and less-probable values (appear less) to $> n$, we can reduce the number of bits used: $L_{avg} < n$.

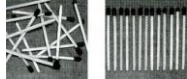
- **Interpixel (spatial) redundancy:** due to the correlation between pixel values that are repeated (ordered) along the spatial image. Some values could be predicted through statistics, rather than storing them. 
- **Psychovisual redundancy:** due to irrelevant details that are not important, but they occupy memory. They could be ignored.
- **Fidelity criteria:** used to quantify the nature of the information loss. It can be objective (SAD between original and compressed images, SNR) or subjective (by visual inspection).

Image compression model includes:

- **Mapper:** transforms image to another space designed to reduce redundancy (reversible, may reduce or not data).
- **Quantizer:** reduces accuracy of mapper's output in accordance with a fidelity criterion (irreversible).
- **Symbol coder:** minimizes code redundancy (reversible).

COMPRESSION TYPES

Lossless compression

No quantizer, reversible, no distortion and low C_R . Recommended for medical images.

- **Shannon's first theorem:** an image can be represented with as few as H (entropy) bits/pixel (taking into account code redundancy).
- **Huffman coding:** implementation of Shannon's first theorem.
 1. Order probabilities of symbols (values) and combine (sum) the two lowest probabilities into one in the next source reduction.
 2. Code each reduced source with 0 or 1, so that each time one of them is the symbol expanded (alternating).
 3. When decoding the bits back to the symbols, only one possible decodification is possible (uniquely decodable blocks).

Symbol	Probability	Source reduction				Source expansion						Code			
a_1	0.4	0.4	0.4	0.4	0.6	0	0.4	1	0.4	1	0.4	1	1		
a_2	0.3	0.3	0.3	0.3	0.4	0.4	1	0.3	00	0.3	00	0.3	00	00	
a_3	0.1	0.1	0.2	0.3				0.3	01	0.2	010	0.1	011	011	
a_4	0.1	0.1	0.1							0.1	011		0.1	010	0100
a_5	0.06												0.1	0101	01010
a_6	0.04													01011	

- **Run Length Encoding (RLE):** takes into account repeating intensity strings in the image. Can result in an expansion instead of compression if there are no repeating intensities.
- **LZW Coding:** takes advantage of repeated strings of symbols in the data, a dictionary is created during the coding process.

Lossy compression

Includes quantizer, irreversible, distortion and high C_R .

- **Joint Photographic Experts Group (JPEG):** high compression, user decides trade-off between fidelity and compression ($\hat{Q} \rightarrow \downarrow C_R$), valid for any type of digital image.
 1. Image is decomposed in sub-images (8×8) which are processed independently with a discrete cosine transform (reversible).

$$f(x, y) \xrightarrow{DCT} F(u, v)$$

2. Quantization of the transformed coefficients (irreversible, some pixels are changed) and inverse transform.

$$G(u, v) = \text{round} \left(\frac{F(u, v)}{Q(u, v)} \right) \xrightarrow{DCT^{-1}} g(x, y) \neq f(x, y)$$

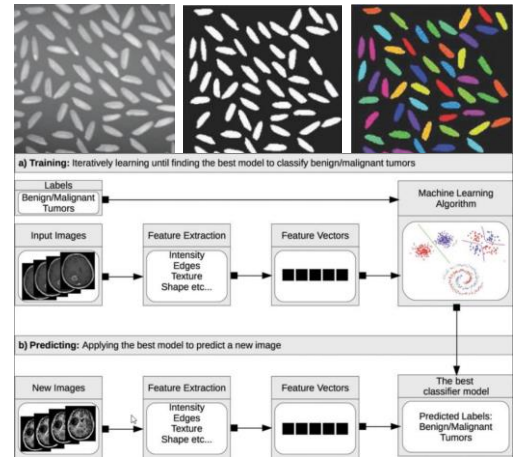
3. Any standard symbol encoder is applied.

Blocking artifact because image is processed independently in the different sub-images and *color distortion* because RGB color is transformed into a different space and processed.

Image Recognition and Classification

Object recognition or classification are a final step in a general diagnostic process:

- **Object segmentation:** used to separate ROIs (objects we want to classify) from background, returns a binary image or mask. If threshold is not properly defined, some objects may not be included.
- **Object detection:** foreground pixels that are connected are detected as a single object. Pixel connectivity in 2D is checked on the 8 surrounding pixels.
- **Object labeling:** each object is uniquely identified by a label (number).
- **Object recognition:** determines if an object belongs or not to a category (yes-or-no question).
- **Object classification:** step further than recognition, sorts the object into one of many classes or categories.



FEATURE EXTRACTION

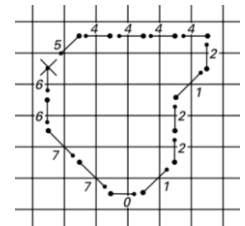
To help in object recognition and classification, *features* are extracted from detected objects. Features (or descriptors) are properties of an object that describe it in some aspect. They should be:

- **Robust:** invariant to translation, orientation, scale, illumination, noise or artifacts.
- **Discriminating:** features describing objects in different classes should be well-separated and non-overlapping.
- **Reliable:** objects of same class should have similar features.
- **Independent:** uncorrelated.
- **Perceptive:** it is helpful if features incorporate lessons from human perception.

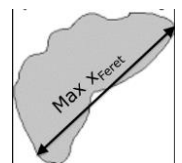
Shape features

Many of them can be taken from the segmentation, pixel values are not important but pixel distribution is.

- **Area (A):** number of pixels in the object.
- **Perimeter (P):** number of pixels in the object boundary. Obtained using the *chain code* of the boundary.
 1. Boundary pixels (having a neighbor equal to 0) are detected.
 2. Boundary pixels are connected anti-clockwise by horizontal, vertical or diagonal segments.
 3. Each direction is identified by a code (numbers from 0 to 7). The whole set of codes is the chain code.
 4. Codes corresponding to horizontal and vertical moves account for 1-pixel long segments, while diagonal moves account for $\sqrt{2}$ -long segments.
 5. $P = \#even\ codes + \sqrt{2} \cdot \#odd\ codes$



- **Maximum Feret's diameter:** line between two furthest points on the perimeter.
- **Eccentricity:** ratio of maximum Feret's diameter to maximum length perpendicular to it.
- **Circularity:** $4\pi A/P^2$, takes maximum value of 1 for a circle.
- **Euler number:** number of connected objects minus number of holes. Topological descriptor (unaffected by deformations).
- **Fourier descriptors:** used to describe overall object shape. Contour of the object is transformed into sines using Fourier transform. Sines with higher energy (amplitude) are more relevant for shape description, so they are selected to reconstruct the object. By the *Nyquist theorem*, frequency ranges from 0 to half the sampling frequency. The more Fourier coefficients taken, the higher resolution in that range and the more accurate the shape will be.

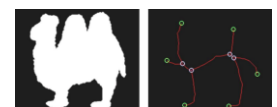


- **Skeleton:** used to get branch points (3 neighbors) and edge points (1 neighbor).

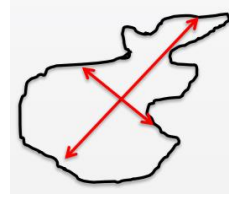
- **Statistical moment:** $m_{mn} = \sum_{x=1}^M \sum_{y=1}^N x^m y^n f(x, y)$

- (x, y) is pixel position, $f(x, y)$ is pixel value (two directions in 2D).

- In binary image: $f(x, y) = \begin{cases} 1 & \text{object} \\ 0 & \text{background} \end{cases} \rightarrow m_{00} = \sum f(x, y) = A$



- **Central moment:** $\mu_{mn} = \sum_{x=1}^M \sum_{y=1}^N (x - \mu_x)^m (y - \mu_y)^n f(x, y)$
 - **Variance** of x and y : μ_{20} and μ_{02}
 - **Covariance** between x and y : μ_{11}
 - **Covariance matrix:** $\Sigma = \begin{bmatrix} \mu_{20} & \mu_{11} \\ \mu_{11} & \mu_{02} \end{bmatrix}$
- **Principal axes:** correspond to eigenvectors of Σ . Used to reduce dimensionality.
 - **Center of gravity:** $(\mu_x, \mu_y) = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right)$
 - **Angle** for largest eigenvalue: $\theta = \frac{1}{2} \arctan \left(\frac{2\mu_{11}}{\mu_{20} - \mu_{02}} \right)$

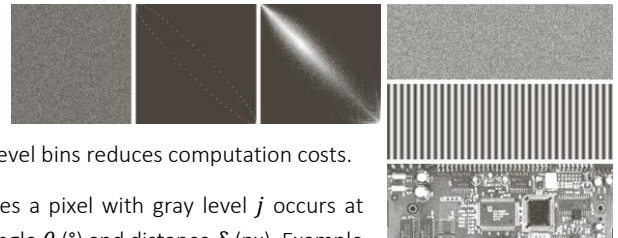


Content features

General intensity properties obtained from image *histogram*. Texture is related to intensity distribution and the relative position of pixels, so it is measured with *gray level matrices* (histogram is not enough).

- **Statistical descriptors:** computed from histogram. See *moment* and *central moment* in [Histogram statistics](#).
 - Normalized central moment (divided by σ^n) \rightarrow **skewness** ($\gamma_1 = \frac{\mu_3}{\sigma^3}$) and **kurtosis** ($Kurt[X] = \frac{\mu_4}{\sigma^4}$).

Gray level matrices (P) are created using a *position operator* p that indicates relative position from reference pixel to the one we want to compare (e.g., lower right direction). Several p should be tested. P is usually normalized to total number of pixels. Texture (entropy, uniformity...) can be described from P . Reducing the number of gray level bins reduces computation costs.



- **Gray level co-occurrence matrix (GLCM):** P_{ij} is number of times a pixel with gray level j occurs at position p from a pixel with gray level i . p usually expressed as angle θ (°) and distance δ (px). Example with p equal to “lower right” pixel, explaining the value at position (2, 3) (number of pixels of value 2 with a pixel of value 3 located one column down, one column to the right):

$$I = \begin{bmatrix} 5 & 2 & 5 & 4 & 4 \\ 3 & 3 & 3 & 1 & 3 \\ 2 & 1 & 1 & 1 & 3 \\ 4 & 2 & 2 & 2 & 3 \\ 3 & 5 & 3 & 3 & 2 \end{bmatrix} \rightarrow P = \begin{bmatrix} 0 & 2 & 2 & 0 & 0 \\ 0 & 2 & 3 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

reference pixel value
1
2
3
4
5

A symmetrical GLCM is obtained if distance is considered in two directions (e.g., pixels to left *and* right of reference pixel).

- **Gray level size zone matrix (GLSZM):** quantifies gray level zones (connected pixels sharing same intensity) in an image. 8-connectivity in 2D and 26-connectivity in 3D. P_{ij} is number of zones with gray level i and size j . Rotation independent.

$$I = \begin{bmatrix} 5 & 2 & 5 & 4 & 4 \\ 3 & 3 & 3 & 1 & 3 \\ 2 & 1 & 1 & 1 & 3 \\ 4 & 2 & 2 & 2 & 3 \\ 3 & 5 & 3 & 3 & 2 \end{bmatrix} \rightarrow P = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 \end{bmatrix}$$

zone size
1
2
3
4
5
gray level
1
2
3
4
5

- **Gray level run length matrix (GLRLM):** quantifies gray level runs (consecutive pixels sharing same intensity) in an image. $P_{ij,\theta}$ is the number of runs with gray level i and length j , along angle θ . For $\theta = 0^\circ$ (horizontal runs):

$$I = \begin{bmatrix} 5 & 2 & 5 & 4 & 4 \\ 3 & 3 & 3 & 1 & 3 \\ 2 & 1 & 1 & 1 & 3 \\ 4 & 2 & 2 & 2 & 3 \\ 3 & 5 & 3 & 3 & 2 \end{bmatrix} \rightarrow P = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 3 & 0 & 1 & 0 & 0 \\ 4 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 \end{bmatrix}$$

run length
1
2
3
4
5
gray level
1
2
3
4
5

Example of application: *PyRadiomics*, platform to extract radiomic features (tumor or tissue characteristics), including first-order statistics, shape descriptors and texture descriptors. The extracted features can be analyzed to find patterns differentiating types of tumor, which can then be used to classify new tumor samples.

FEATURE SELECTION

Depending on the task, some features are more relevant than others to classify objects and can be used to reduce amount of data. To find them, features are ranked by how *informative* they are (how well they can differentiate among different classes) and the most informative ones are selected. Features selected should have low in-class variability and large between-class variability. [Information theory](#) (entropy, mutual information) can quantify relevance of each feature.

- **Forward selection:** start with best single feature and add subsequent best features one by one. Used to select a few features.
- **Backward selection:** start with whole set and remove the worst features one by one. Used to select most of the features.

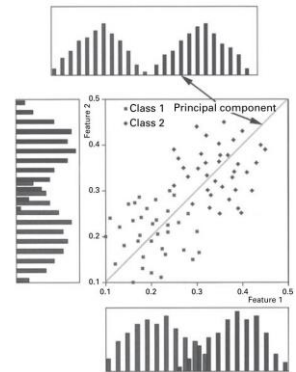
Dimension reduction

Additionally, features can be *combined* into a more informative set with new, uncorrelated features (feature extraction) that can express class variability in a more efficient manner. This allows to use less features than originally, reducing amount of data to process and improving computation costs.

- **Principal component analysis (PCA):** axes of original features are rotated to obtain principal components aligned in the directions of *maximum variability*. PC are linear combinations of original axes that define new, uncorrelated features, and they correspond to eigenvectors of covariance matrix of the feature vector \mathbf{x} . These new features are ranked by the amount of class variability they explain, and only the first ones are retained. PCA does not consider data classification, only distribution.

$$\Sigma = (\mathbf{x} - \mu)^T(\mathbf{x} - \mu) = \begin{bmatrix} \text{var}(x_1) & \cdots & \text{cov}(x_1, x_n) \\ \vdots & \ddots & \vdots \\ \text{cov}(x_n, x_1) & \cdots & \text{var}(x_n) \end{bmatrix} \rightarrow \text{eigenvectors}$$

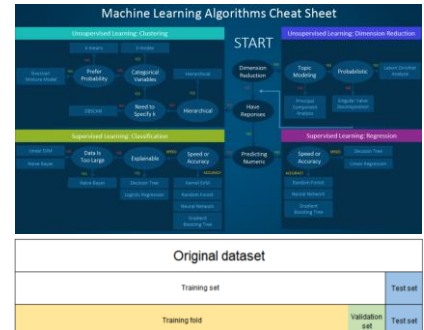
- Ex.: using original features (two dimensions) to sort a dataset into two classes may lead to overlapping of these classes, incorrectly classifying some samples (false positives). However, projecting data onto the principal component allows to optimally separate data into the two classes, reducing the required features to one (one dimension). In this case, the PC lies along the regression line.



OBJECT CLASSIFICATION

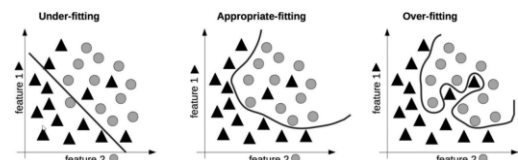
Based on a set of training images of different classes, a classifier (classification algorithm) should be able to correctly sort a new image (or object) into the defined classes. Classifiers are powered by machine learning models, which are developed in stages that use different subsets of the original dataset (dataset partition). Usually, the bigger the dataset available to train a model, the better the model is.

- **Training:** algorithm uses data to learn relationships between features and classes. Several images of each class should be included.
- **Validation:** uses data to improve model by tuning hyperparameters of the classifier (optimization of boundaries between classes).
- **Testing:** uses data unseen for the model during previous stages, provides an estimate of the model performance. Algorithm should be unchanged by these data (no tuning or improving).



Several models can be used in classifiers. An appropriate model minimizes classification error while avoiding the following:

- **Underfitting:** model doesn't adapt enough to nature of variables so it's not flexible.
- **Overfitting:** model adapts so much to training set that it becomes sensitive to small fluctuations and may not perform well with new data.



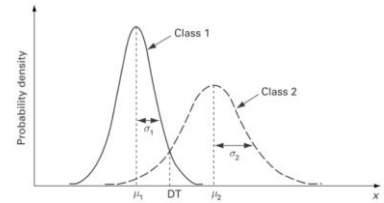
Parametric supervised methods

Training data is labeled with known classes represented by statistical descriptors (estimate parameters and probability distributions). Works better the more data there is.

- **Naive Bayes classifier:** features (x) are used to build probability distributions for each class (ω_i), which are probability density functions (pdf) with Gaussian shape. When a feature is extracted from a new sample, the classifier finds the probability of the sample belonging to each class. Answer is statistical, not definitive.

$$P(\omega_i|x) = P(x|\omega_i) \cdot \frac{P(\omega_i)}{P(x)} \approx P(x|\omega_i) \cdot P(\omega_i)$$

$$P(x|\omega_i) = f(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



- **Posterior probability** $P(\omega_i|x)$: probability of the sample belonging to class ω_i , knowing the feature measurement x .
- **Class-conditional probability** or **likelihood** $P(x|\omega_i)$: probability of measuring feature value x , knowing that the sample belongs to class ω_i .
- **Prior probability** $P(\omega_i)$: probability of sample belonging to ω_i , based on relative number of samples in each class in the whole population before taking the test. If there are more samples of class ω_i , then it is more likely that new samples belong to this class.
- **Evidence** $P(x)$: probability of new sample having feature value x . Same for all classes, it does not affect relative probabilities (scaling factor).

Bayes' decision rule for a two-class model is: new sample belongs to ω_1 if $P(\omega_1|x) > P(\omega_2|x)$, otherwise belongs to ω_2 . We therefore want to maximize $P(\omega_i|x)$.

- If posterior probability for each class do not overlap, then we can be 100% sure which class the sample belongs to (e.g., $P(\omega_1|x) = 1$ and $P(\omega_2|x) = 0$ so sample belongs to ω_1).
- If there is overlapping of the two classes, *optimal decision threshold* (feature value x that best separates two classes) corresponds to intersection of both Gaussian functions:

$$P(\omega_1|x) = P(\omega_2|x) \rightarrow P(x|\omega_1) \cdot P(\omega_1) = P(x|\omega_2) \cdot P(\omega_2) \rightarrow \frac{P(x|\omega_1)}{P(x|\omega_2)} = \frac{P(\omega_2)}{P(\omega_1)}$$

Ex.: suppose two classes with (μ_i, σ_i) equal to (4,1) and (10,1) have prior probabilities $P(\omega_1) = \frac{2}{3}$ and $P(\omega_2) = \frac{1}{3}$. Then optimal decision threshold is:

$$\frac{P(x|\omega_1)}{P(x|\omega_2)} = \frac{P(\omega_2)}{P(\omega_1)} \rightarrow \frac{\frac{1}{\sqrt{2\pi}} e^{-\frac{(x-4)^2}{2}}}{\frac{1}{\sqrt{2\pi}} e^{-\frac{(x-10)^2}{2}}} = \frac{1/3}{2/3} = \frac{1}{2} \rightarrow e^{\frac{-(x-4)^2 + (x-10)^2}{2}} = \frac{1}{2} \rightarrow \dots \rightarrow \boxed{x = 7.12}$$

- Multidimensional analysis: if there is more than one feature, a multidimensional feature space is built with each class characterized by a multidimensional Gaussian pdf.
- **Mahalanobis distance:** adapts the distance measurements to data distribution by considering the standard deviation and orientation of variables. More appropriate than Euclidean distance. Computed from measured features x , class centers of mass μ and **covariance matrix** Σ .

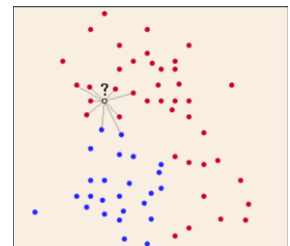
$$D_M(x) = \sqrt{(x - \mu)^T \Sigma^{-1} (x - \mu)}$$

- **Euclidean distance** obtained if Σ is identity matrix.
- **Standardized Euclidean distance** obtained if Σ is diagonal.

Non-parametric supervised methods

Training data is labeled with known classes, but the model doesn't assume anything about their data distribution and statistics.

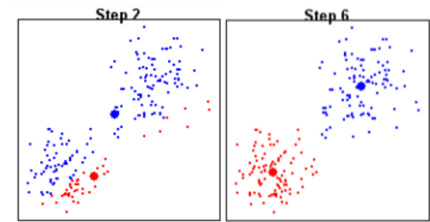
- **k-nearest neighbors classifier** (k-NN): new sample is assigned the most frequently occurring label in the k nearest neighbors (k is specified by user). To find the k nearest neighbors, distances from new sample to all labeled samples are computed and the minimum distances are used. In three-dimensional feature space, Euclidean distances result in spheres and **Mahalanobis** in ellipsoids. High-dimensional feature space and large k values lead to high computational costs.



Unsupervised methods

Classes of training data are unknown, so the model creates the classes by finding natural *clusters* (groups) of samples.

- **k-means clustering classifier:** k random classes are created and iteratively adjusted to fit the real data distribution (k is specified by user). It depends on *initialization* and may converge to a local minimum instead of the global minimum that best represents the ideal solution, so it should be run several times with different initial prototypes to increase probability of finding best solution.
 1. Randomly choose k objects from the training set (prototypes).
 2. Create k classes by assigning all other objects to nearest prototype.
 3. Compute new prototype as the center of mass of each class.
 4. Repeat the process until some stopping criteria (number of iteration, convergence or threshold) has been achieved.



CLASSIFICATION EVALUATION

There are metrics to evaluate performance of a classification algorithm.

- **Confusion matrix (CM):** rows correspond to real classes and columns to predicted ones. M_{ij} is the number of samples belonging to class i in the real world that are predicted to be in class j . A perfect classification CM would have all zeros except for the diagonal, meaning that all predicted classes correspond to the real ones.
- **Receiver operating characteristic (ROC) curve:** shows performance of a binary classifier for different decision thresholds. Assuming classes are positive (P) and negative (N), a sample can be predicted as true P/N (correct) or false P/N (incorrect). A curve is plotted with true positive rate (sensitivity) on y-axis and false positive rate (specificity) on x-axis, for several decision threshold values. Area under the curve is proportional to performance. Threshold that falls closer to upper left corner yields better results.

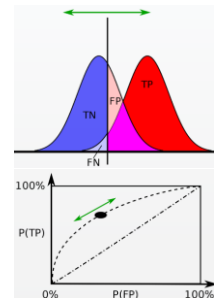


Image Software

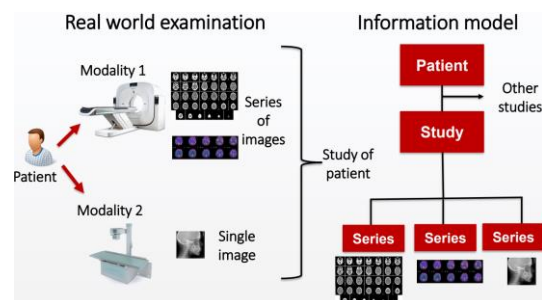
DICOM

DICOM⁷ (Digital Imaging and Communications in Medicine) is the international standard (since 1993) for communication of biomedical digital images (also audio, ECG, EEG...) and related data. Achieves compatibility, improves workflow efficiency, cooperative, incorporated into medical imaging devices.

DICOM information model

All-in-one digital format containing image, patient data, acquisition data... Data stored as *attributes* (fields), identified by name, tag and the value. Information is organized in a specific structure:

- **Information object:** file itself, containing all info, can be sent and received by DICOM-compatible devices.
- **Information object definition (IOD):** indicates the type of object or image (MRI, PET, X-ray...), as well as the structure, attributes included and the standard to manipulate it.
- **Information entity (IE):** collection of modules that refer to a same real-life object. Patient, study and series have unique identifiers (UID).
 - Image: anatomical/functional information obtained in a single acquisition event, is the bottom level in information model.
 - Series: collection of one or more images (usually same modality) representing same anatomical/functional object. Can be a 3D or temporal reconstruction.
 - Study: collection of one or more series (can be different modality) obtained from the same patient on the same appointment.
 - Patient: individual subject to one or more studies, is the top level in information model.
 - Other: frame of reference, equipment...
- **Module:** group of related attributes within an IE.
- **Attributes:** can be required (M), conditional (C) or optional (U). Each IOD will be defined by different attributes.
 - Patient name, study date, contrast, image size, bits allocated, rescale intercept and slope ($OU = m \cdot SV + b$, where OU is output units and SV is stored value)...



DICOM services

Common data interchange protocol that allows communication (image acquisition, archive, processing, visualization or printing) between different devices, manufacturers... as well as with the hospital information system (HIS). Follows a client-server model:

- **Service-object pair (SOP):** a service (store, query, retrieve, print...) is coupled to an object (image) to conform a request that can be handled by DICOM-compatible devices. Devices need to support each specific SOP in order to be able to handle it.
- **Service class user (SCU):** behaves as client, makes an SOP request.
- **Service class provider (SCP):** behaves as server, responds to an SOP request.

SCU (client)	Service	SCP (server)
Imaging device	DICOM Worklist	RIS in charge of scheduling
Imaging device	DICOM Store	PACS
Radiologist's workstation	DICOM Query/Retrieve	PACS
Clinician's workstation	DICOM Query/Retrieve (report)	PACS or radiologist's workstation
Radiologist's or clinician's workstation	DICOM Print	Printer

- **DICOM conformance statement:** list of SOPs supported by a product (device or software) and how to implement them.
- **Application entity title (AE Title):** unique name to identify a DICOM device or program within a PACS network.

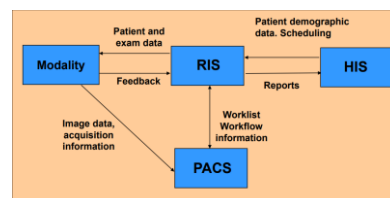
DICOM services run as an application layer protocol on top of TCP/IP (transmission control protocol/Internet protocol) layers, which are in charge of correctly sending data between devices over a network. IP locates the address of servers and clients (computers) and TCP locates ports (entries to a computer). One port can only be used for one DICOM service. Port used for DICOM communication is 104. Local host port is 127.0.0.1.

Hospital radiology

- **Radiology department:** combines different imaging modalities (MR, CT, NM, US...) and users (radiologist, clinicians, technicians, administrative...), high costs, works with digital images and interacts with all other departments.
- **Radiology workflow:** process to follow whenever a patient needs to get a radiology analysis. Everything must be compatible with DICOM, which is also compatible with Health Level 7 (HL7, the international standard for exchanging healthcare information).




1. Patient data is retrieved from HIS or radiology information system (RIS) and used to schedule an appointment.
2. Appointments are organized in a worklist (WKLST, in DICOM format), which distributes patients into the machines to be used (provides all patient information).
3. After image is taken, data is sent and stored in a picture archiving & communication system (PACS, very large database).
4. Image is sent from PACS to diagnostic workstation, where the radiologist evaluates and makes a report (and optionally prints the image). Radiologist also receives from RIS the list of images that need reporting.
5. Both image and report are sent together as results to the corresponding clinician's visualization computer through web access.



IMAGEJ

ImageJ⁷ used for visualization in 2D, good for microscopy.

- Open a format image: *File > Open...*
- Open an image from disk: *File > Import > RAW*
- Open a sample image: *File > Open Samples*
- Change LUT: *Image > Lookup Tables*
- Change type of image: *Image > Type*
- Zoom in/out: +/-
- Window settings: *Image > Adjust > Brightness/Contrast*
- Measure: *m*
- Set measurements: *Analyze > Set Measurements...*
- Thresholding: *Image > Adjust > Threshold...*
- Region growing:  *Wand tool*
- Create mask from selection: *Edit > Selection > Create Mask*
- Algebraic operations: *Process > Image Calculator...*
- Filter with kernel: *Process > Filters > Convolve...*
- Find skeleton: *Process > Binary > Skeletonize*
- Find distance map: *Process > Binary > Distance Map*

3DSLICER

3DSlicer[↗] used for visualization in 3D, good for radiology.

MATLAB

Many image processing techniques can be applied with MATLAB functions.

- `imshow(I, [])` : visualize a 2D image
- ...