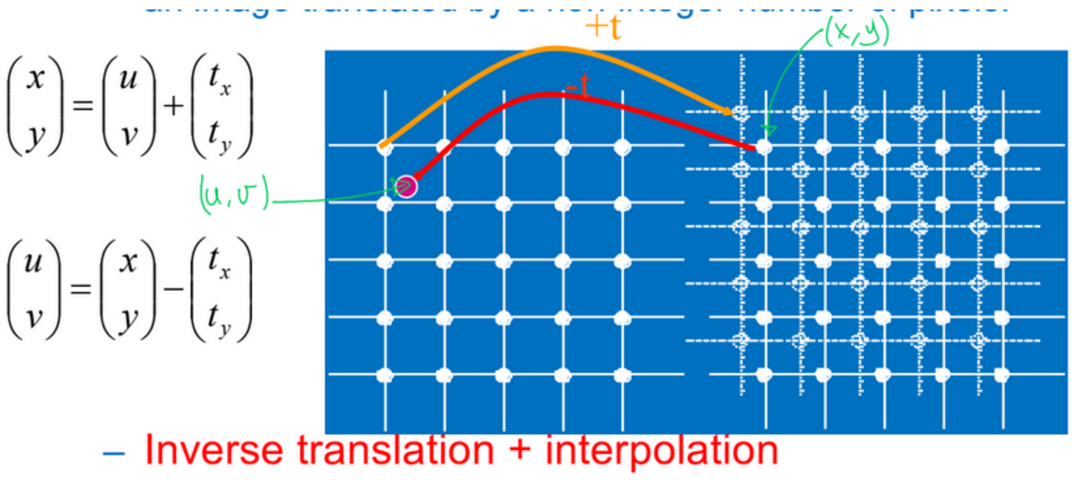
IAPR interview questions

Long questions

Present how to perform geometrical transformations of a digital image? Take the example of a translation of a non-integer number of pixels. (C01_s12-17)

- Preprocessing step :
  - we used it in the lab 2 to align and center the numbers in the image to compute some region based descriptors that were not invariant to rotation or translation such as the descriptors derived from the horizontal and vertical projections.
  - Also, used those geometrical transformations in the same lab to find the minimal distance of an object to a certain distance map.
  - medical imaging: composentation of the difference of position between two patients f.ex

- example of translation by non-integer c01_s12



$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

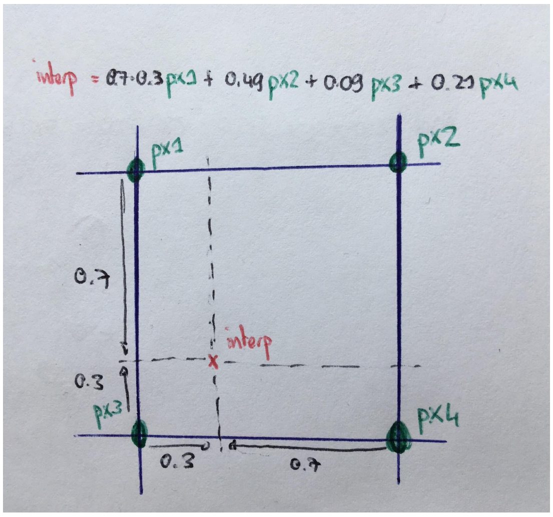
$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

– Inverse translation + interpolation

- we want to move the pixels not the screen, but the grid of pixels is fixed on the screen. For non-integer translation, we have to calculate new value instead of just translate already known ones

- we see that the value of the pixel (x,y) at the base of the red arrow on the figure (right part) correspond to the value of the pixel (u,v) located at the red dot on the original grid.

- We can find the value of (u,v) by interpolation.
  By giving weights dependent to the relative pixel positions to the one of interest, we can obtain a simple and pretty accurate linear interpolation of the pixel value.
  the more points we use to interpolate, the more accurate the result



- The general approach to do a geometrical operation is to do the inverse transformation than the one wanted on the original image and compute the values obtained by interpolation on a set of (neighbors) pixels. Eventually, compute the values at the right places on the final image.

- generalizabe for all linear transformations:
  - translation
  - rotation c01_s13
  - scaling

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} a_1 & a_2 & a_0 \\ b_1 & b_2 & b_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$\text{Translation}: \mathrm{a}_0 = -t_x, a_1 = 1, a_2 = 0, b_0 = -t_y, b_1 = 0, b_2 = 1$$

$$\text{Rotation}: \mathrm{a}_0 = 0, a_1 = \cos\theta, a_2 = \sin\theta, b_0 = 0, b_1 = -\sin\theta, b_2 = \cos\theta$$

$$\text{Zoom}: \mathrm{a}_0 = 0, a_1 = 1/s, a_2 = 0, b_0 = 0, b_1 = 0, b_2 = 1/s$$

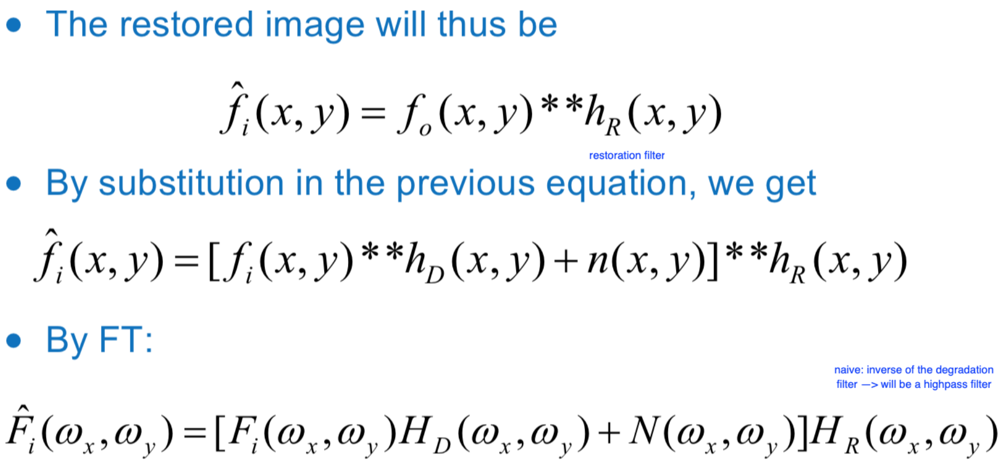
- as well as non-linear transformation like polynomial transformations

What is image restoration? On this context what is inverse filtering and what is a Wiener filter? (C01_s36-49)

- invert non-wanted effect on an image
  - degradation by undesired low-pass filtering effect due to the acquisition system -> *degradation filter*
  - additive noise

$$\underbrace{f_o(x,y)}_{\text{observed image}} = \underbrace{f_i(x,y)}_{\text{ideal image}} ** \underbrace{h_D(x,y)}_{\text{degradation filter}} + \underbrace{n(x,y)}_{\text{noise}}$$

- **inverse filtering**: c01_s37
  find the best *restoration filter h_r* that will restore the image



- The restored image will thus be

$$\hat{f}_i(x,y) = f_o(x,y) ** \underbrace{h_R(x,y)}_{\text{restoration filter}}$$

- By substitution in the previous equation, we get

$$\hat{f}_i(x,y) = [f_i(x,y) ** h_D(x,y) + n(x,y)] ** h_R(x,y)$$

- By FT:

naive: inverse of the degradation filter —> will be a highpass filter

$$\hat{F}_i(\omega_x,\omega_y) = [F_i(\omega_x,\omega_y)H_D(\omega_x,\omega_y) + N(\omega_x,\omega_y)]H_R(\omega_x,\omega_y)$$

- Naive solution: take a filter h_r with an inverse frequency response to h_D

$$\boxed{H_R(\omega_x,\omega_y) = \frac{1}{H_D(\omega_x,\omega_y)}}$$

- h_D lowpass -> h_r highpass
  convolution of the degraded image with h_r will get rid of the effect of h_D but will only amplify the effect of the noise which is generally located in the high frequency spectrum

- New solution: **Wiener filtering**
  - **hypothesis**: images: 2D random variables with zero mean
  - **Goal**: find filter that minimize the quadratic error (expectation of the squared difference of the ideal image and its estimate)

$$\varepsilon = E\left\{\left[f_i(x,y)-\hat{f}_i(x,y)\right]^2\right\}$$

- After some computations, it can be determined that the frequency response of the filter minimizing this error is the following.

$$\boxed{H_R(\omega_x,\omega_y)=\frac{H_D^*(\omega_x,\omega_y)}{\left|H_D^*(\omega_x,\omega_y)\right|^2+\dfrac{P_N(\omega_x,\omega_y)}{P_{f_i}(\omega_x,\omega_y)}}}$$

Noise Power Spectrum

Ideal image Power spectrum

frequency response:

- if noise power is low (low freq) -> inverse filter
- if noise power is high (high freq) -> 0

-> adaptive band-pass filter :

behaves like inverse filter at low frequencies
behaves like a low-pass filter at high frequencies

Need to estimate h_D and noise power spectrum

- h_D impulse response of the system: can be deducted from a fragment of the image that contain a punctual object
- noise power spectrum: uniform region of an image show the noise (remove the mean). FT of its autocorrelation function -> estimation of the noise power spectrum

Explain what object labeling is and the algorithm to implement it. (c02_s03-18)

- **object:** part of an image that ils semantically coherent
    - connex
    - coherent color
    - surrounded by sharp contours
    - coherent texture
    - prior knowledge

- **object labelling:** separation of the object(s) from the background in an image by assigning it defined pixel value

- part of **segmentation** process**:** partition of an image in a finite number of regions (objects) so that those regions does not intersect.
    - region-based methods
    - contour-based methods

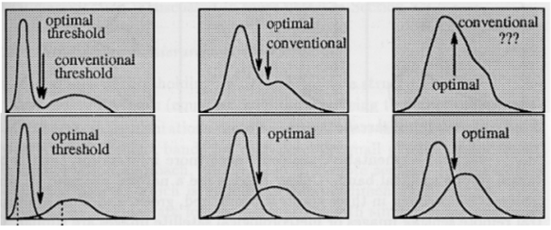
1. Thresholding

    separation of the image by setting one or several thresholds on the gray levels
    problem: **find the optimal threshold**

    - a priori knowledge of the value
    - known properties of the objet
    - histogram study

    on histogram:
    choice of the threshold value that **minimize the segmentation error**
    optimal thresholding:

    - identifying the gray level distribution of each class -> generally fitting a Gaussian model



advantages:

- very simple and works well on certain cases

limitations:

- limited number of cases where it works
- does not define objects, only separates foreground from background

2. Region growing
    - fix a **starting point** (seed) in the desired region
    - define a **homogeneity criterion** used to define the region
        - intensity difference between two neighbors pixels lower than a fixed threshold
        - local variance is lower than a threshold
    - by a **recursive procedure,** we include in the region all the neighboring pixels of the current pixels that satisfy the homogeneity criterion
    - region will grow until it contains all the points connected to the seed, forming a connex region

Region merging:

- consider each pixels (or 2x2, 4x4,... blocks) as a region
- fuse adjacent regions if homogeneity criterion of the union is respected
- result in a very rough segmentation

Region splitting:

- dual approach
- start with the entire image
- separate recursively (for example in 4 parts) until we get small regions that are homogeneous enough
- often result in over-segmentation

Split and Merge:

- one start by splitting the image into to many small regions and merge them afterward

What are the main principles of edge detection, and the two main families of methods to do edge detection? Present typical methods for each family. (c02_s20-25)

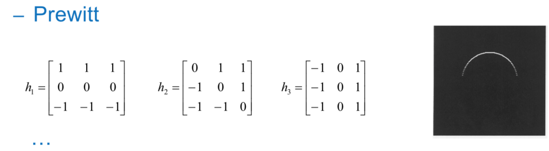
- way to define region

- kind of **segmentation** process**:** partition of an image in a finite number of regions (objects) so that those regions does not intersect.
  - region-based methods
  - contour-based methods

- objects often defined by sharp transitions

- **edge**
  - sharp transition of intensity in an image
  - intensity profile is like a step function
  - **1st derivative** has a **maximum**
  - **2st derivative** has **zero-crossing**

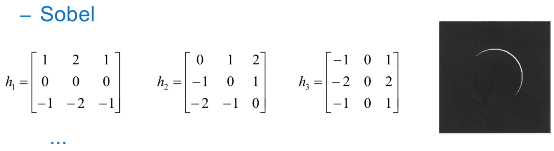
1. maximum of first derivative
   finite approximation of the first derivative by convolution with high-pass filter
   orientation-sensitive -> use multiple fiters for horizontal, vertical and diagonal detection
   high pass filters amplify the noise -> good idea to lowpass the original image beforehand
   a. Robert operator 2x2
   b. Prewitt filter 3x3
   c. Sobel filter 3x3



– Prewitt

$$h_1 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \quad h_2 = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix} \quad h_3 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

...



– Sobel

$$h_1 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad h_2 = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix} \quad h_3 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

...

2. zero-crossing of the 2nd derivative
   Laplacian of Gaussian
   i. noise induce false detection -> gaussian blur to remove the noise
   ii. calculate the second derivative of the filtered image (Laplacian)
   iii. zero-crossing points of the resulting image are the edges

   iv. trick use property of the convolution: we simply need to convolve the image with the second derivative of a Gaussian (hence Laplacian of Gaussian, LoG)

$$\nabla^2\left(G(x,y)**f(x,y)\right)=\left(\nabla^2 G(x,y)\right)**f(x,y)$$

advantages

- Gaussian and Laplacian kernel much smaller than the image -> fewer arithmetic operations
- LoG kernel can be precalculated -> only one convolution performed on the image

What are the 4 main operators of binary mathematical morphology? Explain each of them (c02_s26-39)

- noisy images can create many problems in segmentation
  - irregular borders
  - holes

  solution:
  - smooth the borders
  - fill the holes

- **mathematical morphology:** compare the objects present in an image with a reference object (structuring element), of given size and shape such as a cross, a square or a disk

- **binary mathematical morphology**: compare the foreground of the image (that is binary) to a binary structuring element.

- **four main operators**
  - change the size of the objects
    - dilation
    - erosion
  - fill the holes
    - opening (holes inside the object)
    - closing (holes external to the object)

1. Dilation
   a. expansion of the object by the structuring element
   b. fill the holes inside the object that are smaller than the structuring element

2. Erosion
   a. remove all the pixels all the pixels of the object where the structuring element can't be fitted entirely
   b. object will shrink as all the part smaller than the structuring element will be removed

   Properties:
   - performing multiple steps of dilation/erosion with a small structuring element is the same thing as doing one pass with a larger element
   - The erosion and dilation are linked together by a duality property. Indeed, erosion of an object is the dilation of the background by the (reflection, if no symmetry, of the) structuring element.

3. Opening
    a. erosion of an object with a structuring element followed by a dilation using the same element
    b. union of all the structuring elements included in the object
    c. get rid of the external element of the object smaller than structuring element
    d. opened object is a subset of the initial object, can only be smaller than the original

4. Closing
    a. dilation of the object with a structuring element followed by an erosion using the same element
    b. fill the holes smaller than the structuring element inside the object
    c. closed object is a superset of the initial object, can only be larger or the same size as the original object

Properties:

- duality
- indempotence: making multiple opening / closing with the same structuring element has the same effect as doing it only once

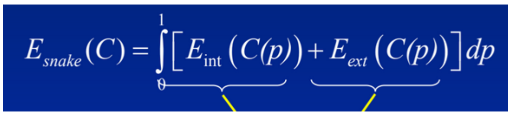
What is the main principle of an active contour? (c03)

- segmentation model popular for theoretical and numerical reasons

- **segmentation:** partition of an image in a finite number of regions (objects) so that those regions does not intersect.

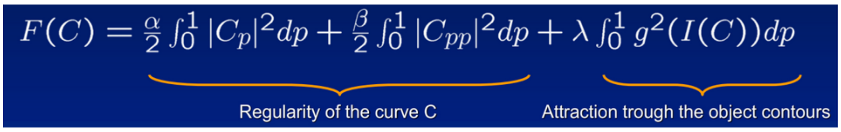
- active contours model perform segmentation by **extracting the contours** of an image.

- **1st model** (Kass-Witking_Terzopoulos)
  - deform a parametrical curve *C* on the object contour by minimizing an **energy function**
    - *C(p) = [x(p), y(p)] with p in [0; 1]*



$$E_{snake}(C) = \int_0^1 \left[\underbrace{E_{\text{int}}(C(p))} + \underbrace{E_{ext}(C(p))} \right] dp$$

  - The **energy term** will be composed of two main terms that are opposite components.
    - **internal energy**: regularity of the curve -> low when the curve is nicely smooth
    - **external energy**: attraction to the object contours -> low when the curve fit nicely on the edges of the object

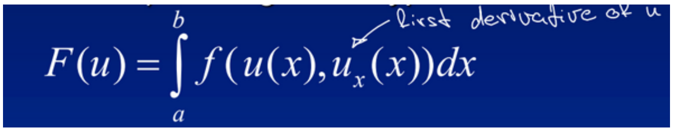


$$F(C) = \underbrace{\tfrac{\alpha}{2} \textstyle\int_0^1 |C_p|^2 dp + \tfrac{\beta}{2} \textstyle\int_0^1 |C_{pp}|^2 dp}_{\text{Regularity of the curve C}} + \underbrace{\lambda \textstyle\int_0^1 g^2(I(C)) dp}_{\text{Attraction trough the object contours}}$$

  - to minimize the energy functional: **variational calculus**:

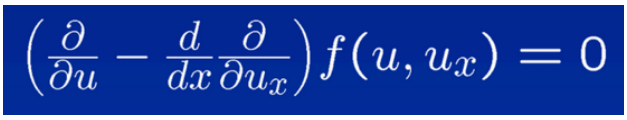
    generalize the computation of the derivative

    allows to transform the **extremization** problem for an equation of this type



$$F(u) = \int_a^b f(u(x), \overset{\text{first derivative of u}}{u_x}(x)) dx$$

    into a problem of resolution of a differential equation (**Euler-Lagrange equation**)



$$\left(\frac{\partial}{\partial u} - \frac{d}{dx} \frac{\partial}{\partial u_x} \right) f(u, u_x) = 0$$

    - using variational calculus on the energy function we end up with a simple system of equations (s17) that can be resolved iteratively

    - **advantage:**
        - fast numerical computation

    - **drawbacks:**
        - final segmentation depends or the parameterization of the curve
        - approach does not allow to generate topological changes

- **2nd model: Geodesic active contour**
    - invariant with the changing of the curve parameterization

    - **energy functional**

$$F(C) = \int_0^1 g(I(C))|C_p|dp$$

    - variation calculus -> Euler-Lagrange equation
      solve the equation iteratively via **Gradient descent**

    - **level set method**
        - solve the topology problem of the 1st model
        - two mathematical representation of a curve:
            - **explicit:** *C = C(p) = (x(p), y(p))*
            - **implicit:** *c = {(x, y): f(x, y) = 0}*
        - implicit form allows to add a level dimension to the curve (useful for non-connex objects)

    - **Advantages:**
        - independent of the curve parameterization
        - automatic topology change thanks to level-set method
        - solutions are mathematically well defined
    - **Drawbacks:**
        - slower numerical computational cost
        - final solution depends of the initial condition (local minim problem)

problem with the general principle of active contours: fail when (parts of) object of interest are hidden or when noise level to high
-> introduction of **prior informations** on the object of interest

What are the Fourier descriptors ? (c04_s22-30)

- **Description step:** describe a 2D shape resulting from a segmentation task with set of adequates numerical features.

  feature used later for object classification, ideally:
  - small intra-class variance
  - large inter-class variance
  - small number of feature
  - independance in translation-rotation

  two main types of representations:
  - **external:** based on shape contours
    - point to point
    - global
  - **internal:** based on the inner part of the shape

- **Fourier descriptors:** contour-based description, global approach.

- **Idea:** fourier transform of the contours: decomposition in “frequencies” bands.

1. *(x_k, y_k) k=0, … N-1* coordinates of the N **successive points** of a contour.
2. define complex number $u_k = x_k + j\, y_k$
3. calculate the DFT of those complex points

$$f_l = \sum_{k=0}^{N-1} u_k e^{-\frac{j2\pi kl}{N}}$$

   complex numbers f_l are the **Fourier descriptors** of the contours

   original contour can be retrieved using inverse DFT

   firsts descriptors contain the majority of the shape information of the object (“lowest frequency” components)

- **Invariances**
  - **Translation**
    translation only affect the f_0 -> just discard it

  - **Rotation**
    rotation only affect the phase of the descriptors by the same amount, does not modify their amplitude -> use feature based on the amplitude

- - **Scaling**
    scaling only multiply the descriptor by a scaling factor -> use the ratio between the descriptors as it gets rid of the scaling factor

  - **choice of u_0**
    the choice of the starting point of the contour affect the phase of the descriptors -> feature based on amplitude are robust

- Energy concentrated in the low frequencies -> Fourier descriptors are powerful for classification: we can use the norm of the firsts descriptors as features.

What is a Freeman code ? (c04_s11-15)

- **Description step:** describe a 2D shape resulting from a segmentation task with set of adequates numerical features.

  feature used later for object classification, ideally:
  - small intra-class variance
  - large inter-class variance
  - small number of feature
  - independance in translation-rotation

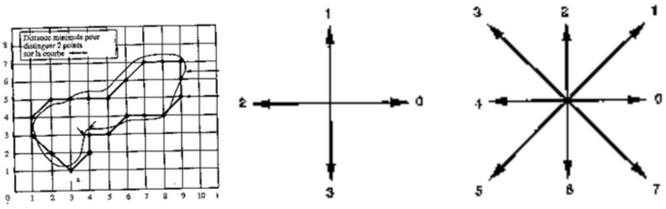
  two main types of representations:
  - **external:** based on shape contours
    - point to point (local)
    - global
  - **internal:** based on the inner part of the shape

- **Freeman code:** contour-based description, point-to-point approach (more sensitive to noise, smoothing is often needed beforehand)

- **Goal:** find a way to represent contour.
  then use this contour for classification by defining **similarity between contours**
  classification feature becomes for example **distance between the contour of the shape and the one of a reference shape**

Freeman code:

1. select a starting point
2. position of the next point is defined with respect to position of the previous in a given connectivity definition (4 or 8) using a **code** defined on 2 or 3 bits.
3. contour encoded as a **character chain** (ex, 120100122445544445677), compact representation



- **Invariant with respect to translation**
- Possibility to deal with rotation of 45 degrees by looking at the matching of the difference between two successive elements rather that the matching between all the elements

- **Attention:** sensitive to the starting point: the same object described twice from a different starting point will have two different codes.
  one code should be fixed and compared with the other code starting from any point

- **Edition distance:** notion of distance between two chains of characters. Used here to measure the similarity between two contours.
  - reflect the **difference in length** between two chain
  - reflect the **number of different characters** between two chains apprearing in corresponding positions.

  -> **Edidition distance** between two chains x and y = **minimum number of elementary operation necessary to transform x into y**

- multiple ways of computing this distance:
  - **Combinatory search:** prohibitive cost
  - **Fisher-Wagner:** recursive algorithm:
    start by calculating distance between little chains and add characters

What is a morphological skeleton ? (c04_s18)

- **Description step:** describe a 2D shape resulting from a segmentation task with set of adequates numerical features.

  feature used later for object classification, ideally:
  - small intra-class variance
  - large inter-class variance
  - small number of feature
  - independance in translation-rotation

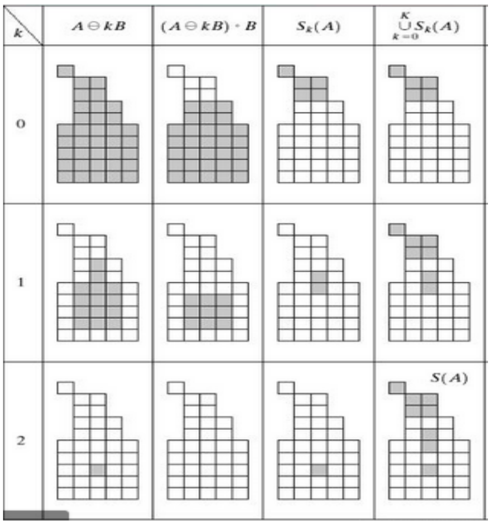
  two main types of representations:
  - **external:** based on shape contours
    - point to point (local)
    - global
  - **internal:** based on the inner part of the shape

- **Morphological skeletons** can be used instead of contours for ex. distance between two skeletons

- Computed recursively with the help of **Morphological operations:**
  - **erosion**
  - **opening**
    see question on the morphological operators

$$S(X) = \bigcup_{r=0}^{N} S_r(X) \quad \text{with} \quad S_r(X) = (X \ominus rB) - (X \ominus rB) \circ B$$

skeleton composed of the union of **partial skeletons** generated by the erosion of the object by a structuring element minus the same eroded image opened with the same structuring element



- opening remove sharp part of the object
  -> subtracting opening = only keeping the sharp parts
  union of those sharp parts -> skeleton

- The objects are then represented by **medial axis** and those representations can be compared between objects.

- In addition, we get an information on what are the small and the large parts of the object as the morphological skeleton is a multi-scale approach. Generally, the main parts of objects have to be similar, therefore, by looking only at the skeleton for the large parts of the object we can compare only those main parts.

- skeleton obtained **depend on the choice of the structuring element**
  -> make this process **invariant to rotation**: **using a combination of structuring element** like using for one step a cross, the other step a square, then the cross again and so on.

What is a Bayesian classifier ? (principle, advantages & limitations, application to Gaussian cases) (c05_s07-18)

- **Classification:** given feature of an object as input, use classification rule to determine if the object is part of a given class
  feature vector + decision surface -> class label

- **two types** of classifiers:
  - **Supervised**
    - training set: set of feature vectors with corresponding class labels
    - classifier exploit prior information
  - **Unsupervised**
    - group similar vectors to create clusters and identify those clusters

- **Bayesian classifier**: supervised one

  Probabilistic approach:
  - feature vectors *x* are assumed to come from a known probability distribution
  - design a classifier that assign a feature vector to the most probable class w_i
  - *x* belongs to class *i* if $P(w_i \mid x) > P(w_j \mid x)$

  - **known prior probabilities***:* $P(w_i), P(w_j)$
    (can be evaluated from the proportion of each class in the training set)
  - **known conditional pdfs:** $p(x \mid w_i)$
    (if not, have to be identified from the training set)
  - **Bayes rule** $$P(w_i \mid x) = \frac{p(x \mid w_i)\ P(w_i)}{p(x)}$$
  - maximize the **a posteiori probability:** $p(x \mid w_i)P(w_i)$

  - **decision surfaces:** *area where* $P(w_i \mid x) = P(w_j \mid x)$
    use a **discrimination function** *f(.)* (monotonically increasing) and write
    $g_i(x) = f(P(w_i \mid x))$
    decision surface -> $g_{ij} = g_i(x) - g_j(x) = 0$

- **Advantage:**
  - Classification errors when proba distributions overlap but Bayesian classifier **minimises the classification errors** -> ideal classifier

- **Drawback:**
  - Assume that probabilities are known which is never truly the case (we work with estimations)

- **Bayesian classification for normal laws:**
  - gaussian pdfs on slide 11

  - **Discrimination function:** use log because it gets rid of the exponentials s12

  - **Special case Σ_i identical for all classes**
    - quadratic terms disappear from the equations of the decision curves as well as constant
    - discriminant functions are linear -> decision surfaces = **hyperplanes**

    - **sub particular case $\Sigma = \sigma^2 I$** (diagonal)
      - decision hyperplanes pass right between the mean of the class and is orthogonal to the line between those means

    - **Σ not diagonal**
      - decision line still pass in the midlle but not orthogonal

  - **Minimal distance classifier**

    without constants: $g_i(x) = - \frac{1}{2}(x-\mu)^T \Sigma_i^{-1}(x-\mu)$

    - **if Σ diagonal** most probable class is class that maximize $g_i(x)$ -> minimize **euclidian distance**
    - **if Σ not diagonal** maximize $g_i(x)$ -> minimize **Mahalanobis distance**

-

Short questions

What is a Median filter ?

- Pre-processing step -> denoising

- **noises**
    - source of noises
        - electrical, mechanical interference on the measurements
        - signal quantifications
    - signal to noise ratio
    - random signal
    - additive signal

- **Non-linear filter**
    - good compromise between filtering power and respect of the image details

1. consider the pixels on a neighborhood of size nxn
2. **sort the value** in ascending order
3. set the **median value** of the list as the value of the central pixel

- **Advantages:**
    - suppression of small variations
    - very effective on “salt & pepper” noise
    - keeps the contours

What is the Laplacian of Gaussian (LoG) method for edge detection?

- **edge**
  - sharp transition of intensity in an image
  - intensity profile is like a step function
  - **1st derivative** has a **maximum**
  - **2st derivative** has **zero-crossing**

Laplacian of Gaussian

zero-crossing of the 2nd derivative

1. noise induce false detection -> gaussian blur to remove the noise
2. calculate the second derivative of the filtered image (Laplacian)
3. zero-crossing points of the resulting image are the edges

4. trick use property of the convolution: we simply need to convolve the image with the second derivative of a Gaussian (hence Laplacian of Gaussian, LoG)

$$\nabla^2\left(G(x,y)**f(x,y)\right)=\left(\nabla^2 G(x,y)\right)**f(x,y)$$

advantages

- Gaussian and Laplacian kernel much smaller than the image -> fewer arithmetic operations
- LoG kernel can be precalculated -> only one convolution performed on the image

How do we calculate the axes of inertia of a binary object? (c04_s44-46)

- **Description step:** describe a 2D shape resulting from a segmentation task with set of adequates numerical features.

  feature used later for object classification, ideally:
  - small intra-class variance
  - large inter-class variance
  - small number of feature
  - independance in translation-rotation

  two main types of representations:
  - **external:** based on shape contours
    - point to point
    - global
  - **internal:** based on the inner part of the shape

- **axis of inertia:**
  - region-based descriptor (internal)
  - system of axes that minimize the variance of the shape projected on the axes
  - **useful** to define the orientation of the object
    align objects
  - two **eigenvectors** of the **covariance matrix**

    $$\begin{pmatrix} \sum_{i=1}^{N}(x_i-\bar{x})^2 & \sum_{i=1}^{N}(x_i-\bar{x})(y_i-\bar{y}) \\ \sum_{i=1}^{N}(x_i-\bar{x})(y_i-\bar{y}) & \sum_{i=1}^{N}(y_i-\bar{y})^2 \end{pmatrix}$$

  - **eigenvalues** express the variance of shape projected on the axes of inertia -> proportional to the length and with of the object can be used to compute the elongation of the object (square root of the ratio of the eigenvalues)

What is a Euclidean distance classifier ?

- **Special case of the Bayesian classifier for normal laws**

- **Classification:** given feature of an object as input, use classification rule to determine if the object is part of a given class
  feature vector + decision surface -> class label

- **two types** of classifiers:
  - **Supervised**
    - training set: set of feature vectors with corresponding class labels
    - classifier exploit prior information
  - **Unsupervised**
    - group similar vectors to create clusters and identify those clusters

- **Bayesian classifier**: supervised one

  Probabilistic approach:
  - feature vectors *x* are assumed to come from a known probability distribution
  - design a classifier that assign a feature vector to the most probable class w_i
  - *x* belongs to class *i* if $P(w_i \mid x) > P(w_j \mid x)$

  - **known prior probabilities***:* $P(w_i), P(w_j)$
    (can be evaluated from the proportion of each class in the training set)
  - **known conditional pdfs:** $p(x \mid w_i)$
    (if not, have to be identified from the training set)

  - **Bayes rule** $P(w_i \mid x) = \dfrac{p(x \mid w_i)\, P(w_i)}{p(x)}$
  - maximize the **a posteiori probability:** $p(x \mid w_i)P(w_i)$

  - **decision surfaces:** *area where* $P(w_i \mid x) = P(w_j \mid x)$
    use a **discrimination function** *f(.)* (monotonically increasing) and write
    $g_i(x) = f(P(w_i \mid x))$
    decision surface -> $g_{ij} = g_i(x) - g_j(x) = 0$

- **Bayesian classification for normal laws:**
  discrimination functionon s12
  if we neglect constants discrimination function becomes $g_i(x) = -\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)$

- **special case** $\boldsymbol{\Sigma_i = \sigma^2 I}$ (diagonal) same for all classes
  - maximizing $g_i(x)$ is the same as minimizing euclidean distance between *x* and *mu*

$$d_e = \|x - \mu_i\|$$

What is a Mahalanobis distance classifier ?

- **Special case of the Bayesian classifier for normal laws**

- **Classification:** given feature of an object as input, use classification rule to determine if the object is part of a given class
  feature vector + decision surface -> class label

- **two types** of classifiers:
  - **Supervised**
    - training set: set of feature vectors with corresponding class labels
    - classifier exploit prior information
  - **Unsupervised**
    - group similar vectors to create clusters and identify those clusters

- **Bayesian classifier**: supervised one

  Probabilistic approach:
  - feature vectors *x* are assumed to come from a known probability distribution
  - design a classifier that assign a feature vector to the most probable class w_i
  - *x* belongs to class *i* if $P(w_i \mid x) > P(w_j \mid x)$

  - **known prior probabilities***:* $P(w_i)$, $P(w_j)$
    (can be evaluated from the proportion of each class in the training set)
  - **known conditional pdfs:** $p(x \mid w_i)$
    (if not, have to be identified from the training set)
  - **Bayes rule** $P(w_i \mid x) = \dfrac{p(x \mid w_i)\, P(w_i)}{p(x)}$
  - maximize the **a posteiori probability:** $p(x \mid w_i)P(w_i)$

  - **decision surfaces:** *area where* $P(w_i \mid x) = P(w_j \mid x)$
    use a **discrimination function** *f(.)* (monotonically increasing) and write
    $g_i(x) = f(P(w_i \mid x))$
    decision surface -> $g_{ij} = g_i(x) - g_j(x) = 0$

- **Bayesian classification for normal laws:**
  discrimination functionon s12
  if we neglect constants discrimination function becomes $g_i(x) = -\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)$

- **Σ not diagonal** same for all classes
  - maximizing $g_i(x)$ is the same as minimizing mahalanobis distance between *x* and *mu*

$$d_m = \left((x-\mu_i)\Sigma^{-1}(x-\mu_i)\right)^{1/2}$$

What is a k-NN classifier ? (c05_s19-20)

- **Classification:** given feature of an object as input, use classification rule to determine if the object is part of a given class
  feature vector + decision surface -> class label

- **two types** of classifiers:
  - **Supervised**
    - training set: set of feature vectors with corresponding class labels
    - classifier exploit prior information
  - **Unsupervised**
    - group similar vectors to create clusters and identify those clusters

- **k-nn** supervised method

- does not make any assumption on classes pdfs

- start from a training set of feature vector with their class label
- new unknown feature vector is assigned to the class the most represented among its **k nearest neighbors**
- error probability R **at least as large** as Bayesian one (Pe)
  - 1-NN : R < 2Pe
  - k-NN : R < (1 + 1/k) Pe

- k to low -> overfitting
- k to high -> underfitting (-> constant prediction to the most represented class)

What is a linear perceptron and how can we train it ? (c05_s21

- **Classification:** given feature of an object as input, use classification rule to determine if the object is part of a given class
  feature vector + decision surface -> class label

- **two types** of classifiers:
    - **Supervised**
        - training set: set of feature vectors with corresponding class labels
        - classifier exploit prior information
    - **Unsupervised**
        - group similar vectors to create clusters and identify those clusters

- **Linear perceptron:** supervised classifier
    - hypothesis: classes can be separated by straight lines (hyperplanes) simple model
    - decision surfaces: $g_{ij}(x) \equiv g_i(x) - g_j(x) = w^T x - w_0 = 0$
        - w: weight vector (orthogonal to the decision surface)
        - w_0: threshold (bias)

    - **goal:** tune w and w_0 so that $g_{ij}(x) > 0$ if x belongs to class *i and* < 0 if y belongs to class j

    - **search space** different values that w can takes

    - **loss function** that we try to minimize

$$J(w) = \sum_{x \text{ mal classifiés}} (\delta_x w^T x)$$

sum of all the absolute of the wrong values

$$\delta_x = -1 \text{ si } x \in \omega_1, \quad \delta_x = +1 \text{ si } x \in \omega_2$$

$$\text{On observe que } J(w) \geq 0$$

    - minimize loss function by **gradient descent**

$$w(t+1) = w(t) - \rho_t \left.\frac{\partial J(w)}{\partial w}\right|_{w=w(t)} \quad \text{with} \quad \frac{\partial J(w)}{\partial w} = \sum_{x \text{ mal classifiés}} \delta_x x$$

$$\boxed{w(t+1) = w(t) - \rho_t \sum_{\text{misclassified } x} \delta_x x}$$

    - ρ_t **gradient step** important parameter:
        - to small -> slow convergence
        - to large -> can induce divergence
        - large at the beginning and decreasing with time

What is a Multi-layer perceptron and how can we train it ? (c04_s24-29)

- **Classification:** given feature of an object as input, use classification rule to determine if the object is part of a given class
  feature vector + decision surface -> class label

- **two types** of classifiers:
  - **Supervised**
    - training set: set of feature vectors with corresponding class labels
    - classifier exploit prior information
  - **Unsupervised**
    - group similar vectors to create clusters and identify those clusters

- **Multi-layer perceptron:** supervised classifier

- multiple layers of linear perceptrons (cf above) -> allows non linear decision planes

- The Multi-layer perceptron allows to perform non-linear classification.
  -> take multiple decisions lines into account

- increasing number of layers allows to generate more complex decision surfaces

- **decision** (activation) **function** essential.
  derivation in the training phase (gradient descent)
  - step function unusable derivative = 0 everywhere
    (non-derivable in 0 too but so is Relu)
  - sigmoid
  - tanh

- decision based on the neuron with the highest output

What is supervised and non-supervised classification ? (c05_s06)

- **Classification:** given feature of an object as input, use classification rule to determine if the object is part of a given class
  feature vector + decision surface -> class label

- **two types** of classifiers:
  - **Supervised**
    - training set: set of feature vectors with corresponding class labels
    - classifier exploit prior information
    - decision rules learned with the help of the training set
    - Bayesian
    - Linear Classifiers
    - Non-linear classifiers (neural nets)

  - **Unsupervised**
    - group similar vectors to create clusters and identify those clusters
    - Clustering algorithms
    - k-means

What is non-supervised classification and describe the k-means algorithm ? (c05_s30-32)

- **Classification:** given feature of an object as input, use classification rule to determine if the object is part of a given class
  feature vector + decision surface -> class label

- **two types** of classifiers:
  - **Supervised**
    - training set: set of feature vectors with corresponding class labels
    - classifier exploit prior information
    - decision rules learned with the help of the training set
    - Bayesian
    - Linear Classifiers
    - Non-linear classifiers (neural nets)

  - **Unsupervised**
    - group similar vectors to create clusters and identify those clusters
    - samples in a defined feature space
    - try to identify region of high sample density, which model the sample probability distribution
    - Clustering algorithms
    - k-means

- **K-means**
  - try to identify m classes by means of their centers (centroids)
  - minimize intra-class variance

  1. choose m centroids randomly
  2. attach each vector x to the class of the closest centroid
  3. recalculate the centroid positions as the mean position of the vector of the class
  4. repeat until convergence.