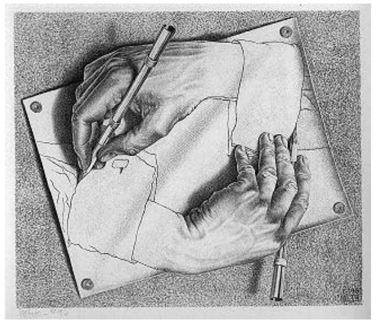
INF 4300 – Digital Image Analysis

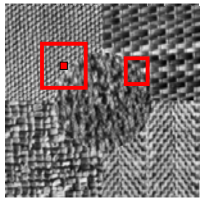
SUMMARY – PART I



Fritz Albregtsen 14.11.2016

”Texture” – description of regions

- Remember: we estimate local properties (features) to be able to isolate regions which are similar in an image (segmentation), and possibly later identify these regions (classification), usually with the final goal of object description
- One can describe the ”texture” of a region by:
  - smoothness, roughness, regularity, orientation...
- Problem: we want the local properties to be as ”local” as possible
- Large region or window
  - Precise estimate of features
  - Imprecise estimate of location
- Small window
  - Precise estimate of location
  - Imprecise estimate of feature values



Using variance estimates

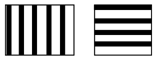
- Variance, σ^2, is directly a measure of ”roughness”
- A bounded measure of ”smoothness” is

$$R = 1 - \frac{1}{1+\sigma^2}$$

  - R is close to 0 for homogenous areas
  - R tends to 1 as σ^2, ”roughness”, increase

1. order statistics discussion

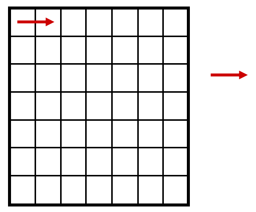
- 1. order statistics can separate two regions even if $\mu_1 = \mu_2$, as long as $\sigma^2_1 \neq \sigma^2_2$
- The statistics of a pixel (x, y) is found in a local window
- Problems:
  - Edges around objects are exaggerated
    - Solution: use adaptive windows
  - 1. order statistics does not describe geometry or context
    - Cannot discriminate between



    - Solution:
      - Calculate 1. order statistics with different resolutions, and obtain indirect information about 2. and higher order statistics.
      - Simply use 2. or higher order statistics.

GLCM

- Matrix element *P(i,j|d,θ)* in a GLCM is 2. order probability of changing from graylevel *i* to *j* when moving distance *d* in direction *θ* .
- Dimension of co-occurrence matrix is *GxG* (*G* = gray-levels in image)
- Reduce the number of gray-levels by re-quantization
- Choose a distance *d* and a direction *θ*

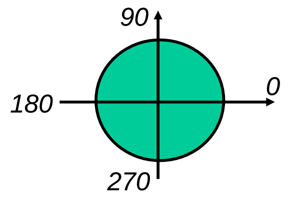


In this example, d=1 and θ=0

- Check all pixel pairs with distance *d* and direction *θ* inside the window. *Q(i,j|d,θ)* is the number of pixel pairs where pixel *1* in the pair has pixel value *i* and pixel *2* has pixel value *j*.

GLCM

- Usually a good idea to reduce the number of *(d,θ)* variations evaluated
- Simple pairwise relations:
  - *P(d,0^0) = P^t (d,180^0)*
  - *P(d,45^0) = P^t (d,225^0)*
  - *P(d,90^0) = P^t (d,270^0)*
  - *P(d,135^0) = P^t (d,315^0)*



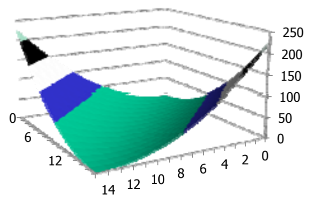
- Isotropic matrix by averaging *P(θ), θ ∈ {0^o, 45^o, 90^o, 135^o}*
  - *Beware of differences in effective window size!*
- An isotropic texture is equal in all directions
- If the texture has a clear orientation, we select *θ* according to this.

GLCM features

- A number of features are available (Haralick et al., …)
- May be seen as weight functions applied to probability matrix:
  - Weighting based on value of GLCM element
    - Example: **Entropy**

W(i,j) = -log{P(i,j)}

  - Weighting based on position in GLCM
    - Example:
      - **Inertia**
      - W(i,j) = $(i-j)^2$



- You should be able to discuss which feature will discriminate between two different textures, based on the distribution of their GLCM's.

Learning goals - texture

- Understand what texture is, and the difference between first order and second order measures
- Understand the GLCM matrix, and be able to describe algorithm
- Understand how we go from an image to a GLCM feature image
  - Preprocessing, choosing d and θ, selecting some features that are not too correlated
- **There is no optimal texture features, it depends on the problem**

Edge-based segmentation

Two steps are needed:

1. Edge detection (to identify "edgels" - edge pixels)
   - (Gradient, Laplacian, LoG, Canny filtering)

2. Edge linking – linking adjacent "edgels" into edges
   - *Local Processing*
     - *magnitude* of the gradient
     - *direction* of the gradient vector
     - edges in a predefined neighborhood are linked if both magnitude and direction criteria is satisfied
   - *Global Processing via Hough Transform*

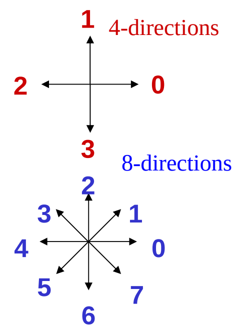
Thinning of edge in 3x3 neighborhood

1 Quantize the edge direction into four (or eight) directions.

2 If gradient magnitude G(x,y) > 0, inspect the two neighboring pixels in the given direction.

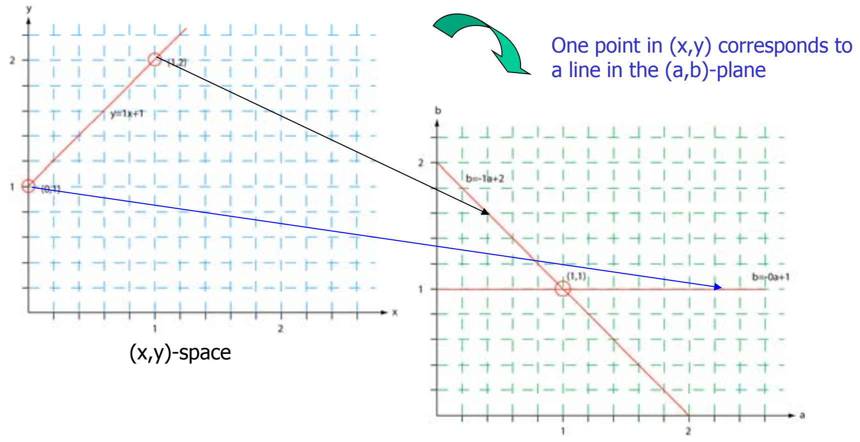
3 If gradient magnitude of any of these neighbors is higher than G(x,y), mark the pixel.

4 When all pixels have been scanned, delete or suppress the marked pixels.



Used iteratively in nonmaxima suppression.

Hough transform – basic idea

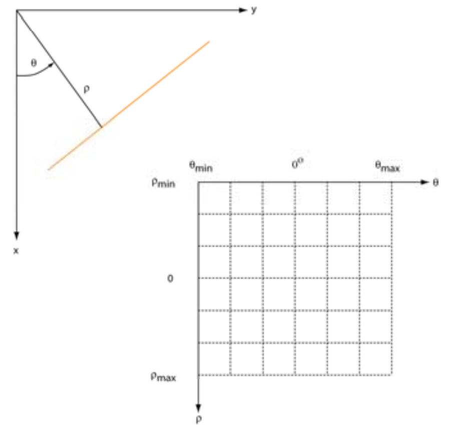


One point in (x,y) corresponds to a line in the (a,b)-plane

(x,y)-space

(a,b)-space

HT and polar representation of lines



Polar representation of lines

HT using the full gradient information

- Given a gradient magnitude image g(x,y) containing a line.
- Simple algorithm:

```
for all $g(x_i,y_i)>T$ do
    for all $\theta$ do
        $\rho= x_i \cos\theta + y_i \sin\theta$
        find indexes (m,n) corresponding to $(\rho, \theta)$ and increment A(m,n);
```

- Better algorithm if we have both
  - The gradient magnitude g(x,y)
  - And the gradient components g_x and g_y
    - So we can compute gradient direction

$$\phi_g(x,y) = \arctan\left(\frac{g_y}{g_x}\right)$$

- The new algorithm:

```
for all $g(x_i,y_i)>T$ do
        $\rho= x_i \cos(\phi_g(x,y)) + y_i \sin(\phi_g(x,y))$
        find indexes (m, n) corresponding to $(\rho, \phi_g(x,y))$, increment A(m,n);
```

Hough transform for circles

- A circle in the xy-plane is given by

$$(x-x_c)^2+(y-y_c)^2=r^2$$

- So we have a 3D parameter space.

- **Simple 3D accumulation procedure:**

```
set all $A[x_c,y_c,r]=0$;
for every (x,y) where g(x,y)>T
    for all $x_c$ and $y_c$
        $r=\mathrm{sqrt}((x-x_c)^2+(y-y_c)^2)$;
        $A[x_c,y_c,r] = A[x_c,y_c,r]+1$;
```

- **Better procedure … ?**

Hough transform for ellipses

- A general ellipse in the xy-plane has 5 parameters:
  - Position of centre (x_c, y_c), semi-axes (a,b), and orientation (θ).
- Thus, we have a 5D parameter space.
- For large images and full parameter resolution,
  straight forward HT may easily overwhelm your computer!

- **Reducing accumulator dimensionality:**
  - Pick pixel pairs with opposite gradient directions
  - Midpoint of pair ”votes” for centre of ellipse!
  - Reduces HT-accumulator from 5D to 3D.
  - Min and max distance of pixel pairs => 2b & 2a

- **Reducing accumulator extent & resolution**
- **Using other tricks …**

Learning goals - HT

- Understand the basic Hough Transform, and its limitations
- Understand that the normal representation is more general

- Be able to implement line detecting HT with accumulator matrix
  - Understand how this may be improved by gradient direction information.

- Be able to implement simple HT for circles of given size,
  in order to find position of circular objects in image.
  - Understand how this may be improved.

- Understand simple HT to detect ellipses
  - Understand how this may be improved.

- Understand the basic random Hough transform.

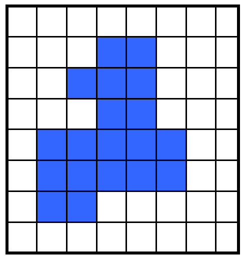
- Do the exercises!

Shape representations vs. descriptors

- After the segmentation of an image,
  its regions or edges are represented and described in a manner appropriate for further processing.
- Shape representation:
  the ways we store and represent the objects
  - Perimeter (and all the methods based on perimeter)
  - Interior (and all the methods …)
- Shape descriptors:
  recipes for features characterizing object shape.
  - The resulting feature values should be useful for discrimination between different object types.

Chains

- Chains represent objects within the image or borders between an object and the background

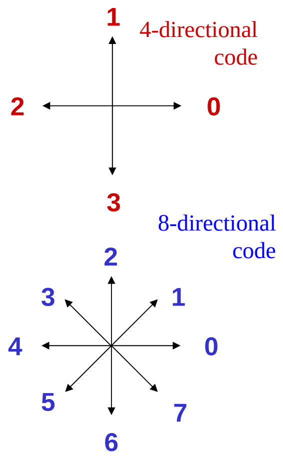


🟦 Object pixel

- How do we define border pixels?
  - 4-neighbors
  - 8-neighbors
- In which order do we check the neighbors?
- Chains represent the object pixel, but not their spatial relationship directly.

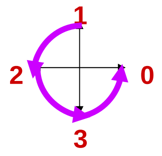
Chain codes

- Chain codes represent the boundary of a region
- Chain codes are formed by following the boundary in a given direction (e.g. clockwise) with 4-neighbors or 8-neighbors
- A code is associated with each direction
- A code is based on a starting point, often the upper leftmost point of the object



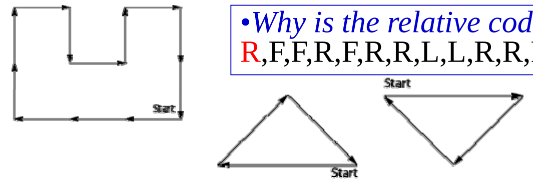
Start point & rotation

- The chain code depends on the starting point.
- It can be made invariant of start point by treating it as a circular/periodic sequence, and redefining the start point so that the resulting number is of minimum magnitude.
- We can also normalize for rotation by using the first difference of the chain code:
  (i.e., direction changes between code elements)
  - Code: 10103322
  - First difference (counterclockwise): 33133030
  - To find first difference, look at the code and count counterclockwise directions.
  - Treating the curve as circular, add the 3 for the first point.
  - Minimum circular shift of first difference: 03033133
- This invariance is only valid if the boundary itself is invariant to rotation and scale.

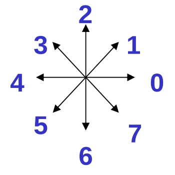


Relative chain code

- Here, directions are defined **in relation to** a moving perspective.
- Example: Orders given to a blind driver ("F", "B", "L", "R").
- *The directional code representing any section of the contour is relative to the directional code of the preceding section.*



•Why is the relative code:
R,F,F,R,F,R,R,L,L,R,R,F ?



Note: rotate the code table so that 2 is forward from your position

- The absolute chain code for the triangles are 4,1,7 and 0, 5, 3.
- The relative codes are 7, 7, 0. (Remember ”Forward” is 2)
- Invariant to rotation, as long as starting point remains the same.
- Start-point invariance by ”Minimum circular shift”.
- Note: To find the first R, we look back to the end of the contour.

Recursive boundary splitting

- Draw straight line between contour points that are farthest apart. These two points are the initial breakpoints.

1. For each intermediate point:
2. Compute the point-to-line distance
3. Find the point with the greatest distance from the line.
4. If this distance is greater than given threshold, we have a new breakpoint.
5. The previous line segment is replaced by two, and 1-4 above is repeated for each of them.

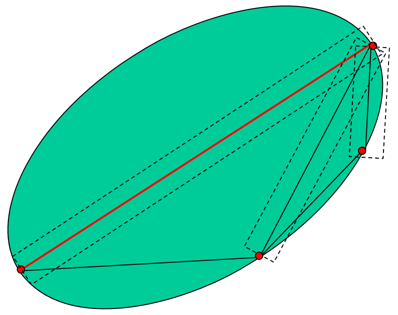
- The procedure is repeated until all contour points are within the threshold distance from a corresponding line segment.
- The resulting ordered set of breakpoints is then the set of vertices of a polygon approximating the original contour
- This is probably the most frequently used polygonization.
- Since it is recursive, the procedure is fairly slow.

Recursive boundary splitting

- Draw line between initial breakpoints; boundary points farthest apart.

1. For each intermediate point:
2. If greatest distance from the line is lager than a given threshold, we have a new breakpoint.
3. The previous line segment is replaced by two line segments, and 1-2 above is repeated for each of them.

- The procedure is repeated until all contour points are within the threshold distance from a corresponding line segment.

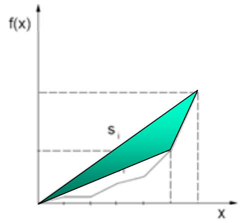


- The resulting ordered set of breakpoints is then the set of vertices of a polygon approximating the original contour.
- The set of breakpoints is a subset of the set of boundary points.

Sequential polygonization

- Start with any contour point as first “breakpoint”.
- Step through ordered sequence of contour points.
- Using previous breakpoint as the current origin, area between contour and approximating line is:

$$A_i = A_{i-1} + \frac{1}{2}(y_i\, x_{i-1} - x_i\, y_{i-1}), \quad s_i = \sqrt{x_i^2 + y_i^2}$$

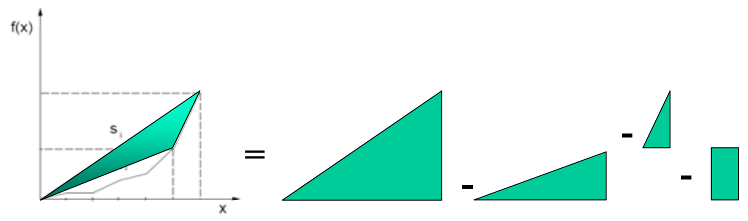


- Let previous point be new breakpoint if
  - area deviation ***A*** per unit length ***s*** of approximating line segment exceeds a specified tolerance, ***T***.

- If **$|A_i| / s_i$ < T, i** is incremented and (**A_i, s_i**) is recomputed.
- Otherwise, the previous point is stored as a new breakpoint, and the origin is moved to new breakpoint.

- This method is purely sequential and very fast.
- Reference: K. Wall and P.E. Danielsson, A Fast Sequential Method for Polygonal Approximation of Digital Curves, Computer Vision, Graphics, and Image Processing, vol. 28, 1984, pp. 220-227.

Sequential polygonization



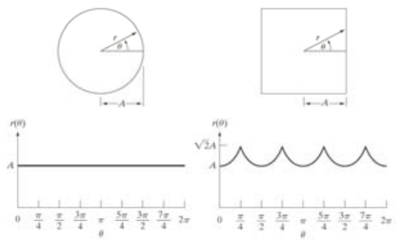
$$\Delta A_i = \frac{1}{2}(x_i y_i) - \frac{1}{2}(x_{i-1} y_{i-1}) - \frac{1}{2}(x_i - x_{i-1})(y_i - y_{i-1}) - (x_i - x_{i-1}) y_{i-1}$$

$$= \frac{1}{2} x_i y_i - \frac{1}{2} x_{i-1} y_{i-1} - \frac{1}{2} x_i y_i + \frac{1}{2} x_{i-1} y_i + \frac{1}{2} x_i y_{i-1} - \frac{1}{2} x_{i-1} y_{i-1} - x_i y_{i-1} - x_{i-1} y_{i-1}$$

$$= \frac{1}{2} x_{i-1} y_i - \frac{1}{2} x_i y_{i-1} = \underline{\underline{\frac{1}{2}(x_{i-1} y_i - x_i y_{i-1})}}$$

Signature representations

- A signature is a 1D functional representation of a 2D boundary.
- It can be represented in several ways.
- Simple choise: radius vs. angle:



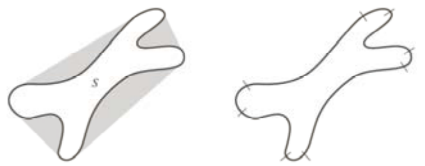
a b

FIGURE 11.10 Distance-versus-angle signatures. In (a) $r(\theta)$ is constant. In (b), the signature consists of repetitions of the pattern $r(\theta) = A \sec\theta$ for $0 \le \theta \le \pi/4$ and $r(\theta) = A \csc\theta$ for $\pi/4 < \theta \le \pi/2$.

- Invariant to translation.
- Not invariant to starting point, rotation or scaling.

Boundary segments from convex hull

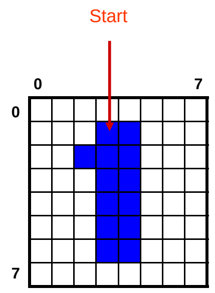
- The boundary can be decomposed into segments.
  - Useful to extract information from concave parts of the objects.
- Convex hull H of set S is the smallest convex set containing S.
- The set difference H-S is called the convex deficiency D.
- If we trace the boundary and identify the points where we go in and out of the convex deficiency, these points can represent important border points charaterizing the shape of the border.
- Border points are often noisy, and smoothing can be applied first.
  - Smooth the border by moving average of *k* boundary points.
  - Use polygonal approximation to boundary.
  - Simple algorithm to get convex hull from polygons.



Contour representation using 1D Fourier transform

- The coordinates (x,y) of these M points are then put into a complex vector s :

  $s(k)=x(k)+iy(k), \quad k \in [0,M-1]$
- We choose a direction (e.g. anti-clockwise)
- We view the x-axis as the real axis and the y-axis as the imaginary one for a sequence of complex numbers.
- The representation of the object contour is changed, but all the information is preserved.
- We have transformed the contour problem from 2D to 1D.



$$s(1) = 3 + 1i$$
$$s(2) = 2 + 2i$$
$$s(3) = 3 + 3i$$
$$s(4) = 3 + 4i$$
$$\vdots$$

x = 3 2 3 3 3 3 4 4 4 4 4 4 3
y = 1 2 3 4 5 6 6 5 4 3 2 1 1

Fourier-coefficients from *f(k)*

- We perform a 1D forward Fourier transform

$$a(u)=\frac{1}{M}\sum_{k=0}^{M-1}s(k)\exp\left(\frac{-2\pi iuk}{M}\right)=\frac{1}{M}\sum_{k=0}^{M-1}s(k)\left(\cos\left(\frac{2\pi uk}{M}\right)-i\sin\left(\frac{2\pi uk}{M}\right)\right),\qquad u\in[0,M-1]$$

- Complex coefficients *a(u)* are the Fourier representation of boundary.
- *a(0)* contains the center of mass of the object.
- Exclude *a(0)* as a feature for object recognition.
- a*(1),* a*(2),....,a(M-1)* will describe the object in increasing detail.
- These depend on rotation, scaling and starting point of the contour.

- For object recognitions, use only the N first coefficients (a*(N), N<M*)
- This corresponds to setting a*(k)=0, k>N-1*

Fourier Symbol reconstruction

- Inverse Fourier transform gives an approximation to the original contour

$$\hat{s}(k)=\sum_{k=0}^{N-1}a(u)\exp\left(\frac{2\pi iuk}{M}\right),\qquad k\in[0,M-1]$$

- We have only used N features to reconstruct each component of $\hat{s}(k)$.
- The number of points in the approximation is the same (*M*), but the number of coefficients (features) used to reconstruct each point is smaller (*N<M*).

- Use an even number of descriptors.
- The first 10-16 descriptors are found to be sufficient for character description. They can be used as features for classification.
- The Fourier descriptors can be invariant to translation and rotation if the coordinate system is appropriately chosen.
- All properties of 1D Fourier transform pairs (scaling, translation, rotation) can be applied.

Examples from Trier et al. 1996

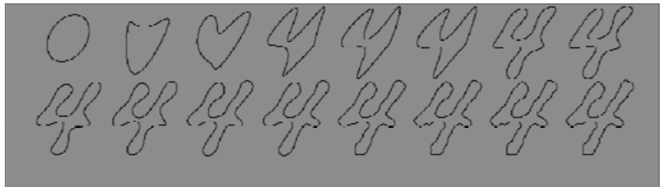


Figure 18: Character '4' reconstructed by elliptic Fourier descriptors of orders up to 1, 2, ...10; 15, 20, 30, 40, 50, and 100, respectively.

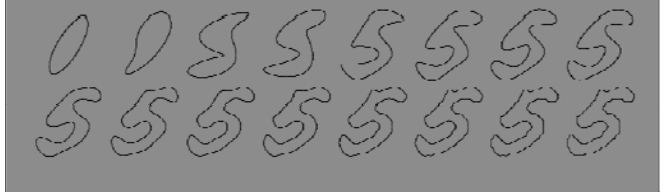


Figure 19: Character '5' reconstructed by elliptic Fourier descriptors of orders up to 1, 2, ...10; 15, 20, 30, 40, 50, and 100, respectively.

Run Length Encoding of Objects

- Sequences of adjacent pixels are represented as ”runs”.
- Absolute notation of foreground in binary images:
  - Run_i = …;<row_i, column_i, runlength_i>; …
- Relative notation in graylevel images:
  - …;(graylevel_i, runlength_i); …
- This is used as a lossless compression transform.
- Relative notation in binary images:

  Start value, length_1, length_2, …, eol,

  …

  Start value, length_1, length_2, …, eol,eol.
- This is also useful for representation of image bit planes.
- RLE is found in TIFF, GIF, JPEG, …, and in fax machines.

Topologic features

- This is a group of invariant integer features
  - Invariant to position, rotation, scaling, warping
- Features based on the object skeleton
  - Number of terminations (one line from a point)
  - Number of breakpoints or corners (two lines from a point)
  - Number of branching points (three lines from a point)
  - Number of crossings (> three lines from a point)
- Region features:
  - Number of holes in the object (H)
  - Number of components (C)
  - Euler number, E = C - H
    - Number of connected components – number of holes
  - Symmetry

Region with two holes

Regions with three connected components

Projections

- 1D horizontal projection of the region:

$$p_h(x) = \sum_y f(x,y)$$

- 1D vertical projection of the region:

$$p_v(y) = \sum_x f(x,y)$$

- Can be made scale independent by using a fixed number of bins and normalizing the histograms.
- Radial projection in reference to centroid -> ”signature”.

Moments

- Borrows ideas from physics and statistics.
- For a given continuous intensity distribution *g(x, y)* we define moments m_{pq} by

$$m_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q g(x,y)\,dx\,dy$$

- For sampled (and bounded) intensity distributions f (x, y)

$$m_{pq} = \sum_x \sum_y x^p y^q f(x,y)$$

- A moment m_{pq} is of *order* p + q.
- This moment is NOT invariant to position of object !

Central moments

- These are position invariant moments :

$$\mu_{p,q} = \sum_x \sum_y (x-\bar{x})^p (y-\bar{y})^q f(x,y)$$

- where

$$\bar{x} = \frac{m_{10}}{m_{00}}, \qquad \bar{y} = \frac{m_{01}}{m_{00}}$$

- The total mass and the center of mass are given by

$$\mu_{00} = \sum_x \sum_y f(x,y), \qquad \mu_{10} = \mu_{01} = 0$$

- This corresponds to computing ordinary moments after having translated the object so that center of mass is in origo.
- Central moments are independent of position, **but are not scaling or rotation invariant.**
- What is μ_{00} for a binary object?

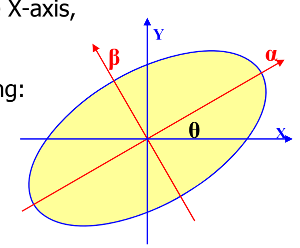
Object orientation - I

- Orientation is defined as the angle, relative to the X-axis, of an axis through the centre of mass that gives the lowest moment of inertia.
- Orientation θ relative to X-axis found by minimizing:

$$I(\theta)=\sum_{\alpha}\sum_{\beta}\beta^2\ f(\alpha,\beta)$$

where the rotated coordinates are given by

$$\alpha = x\cos\theta + y\sin\theta, \quad \beta = -x\sin\theta + y\cos\theta$$



- The second order central moment of the object around the α-axis, expressed in terms of x, y, and the orientation angle θ of the object:

$$I(\theta)=\sum_{x}\sum_{y}\left[y\cos\theta - x\sin\theta\right]^2\ f(x,y)$$

- We take the derivative of this expression with respect to the angle θ
- Set derivative equal to zero, and find a simple expression for θ :

Object orientation - II

- Second order central moment around the α-axis:

$$I(\theta)=\sum_{x}\sum_{y}\left[y\cos\theta - x\sin\theta\right]^2\ f(x,y)$$

- Derivative w.r.t. Θ = 0 =>

$$\frac{\partial}{\partial\theta}I(\theta)=\sum_{x}\sum_{y}2f(x,y)\left[y\cos\theta - x\sin\theta\right]\left[-y\sin\theta - x\cos\theta\right]=0$$

$$\Downarrow$$

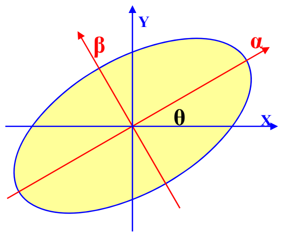
$$\sum_{x}\sum_{y}2f(x,y)\left[xy\left(\cos^2\theta-\sin^2\theta\right)\right]=\sum_{x}\sum_{y}2f(x,y)\left[x^2-y^2\right]\sin\theta\ \cos\theta$$

$$\Downarrow$$

$$2\mu_{11}\left(\cos^2\theta-\sin^2\theta\right)=2\left(\mu_{20}-\mu_{02}\right)\sin\theta\cos\theta$$

$$\Downarrow$$

$$\frac{2\mu_{11}}{\left(\mu_{20}-\mu_{02}\right)}=\frac{2\sin\theta\cos\theta}{\left(\cos^2\theta-\sin^2\theta\right)}=\frac{2\tan\theta}{1-\tan^2\theta}=\tan(2\theta)$$



- So the object orientation is given by:

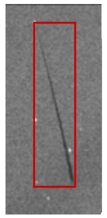
$$\underline{\underline{\theta=\frac{1}{2}\tan^{-1}\left[\frac{2\mu_{11}}{\left(\mu_{20}-\mu_{02}\right)}\right]}}, \quad \textit{where}\ \ \theta\in\left[0,\pi/2\right] \textit{if}\ \mu_{11}>0,\ \ \theta\in\left[\pi/2,\pi\right] \textit{if}\ \mu_{11}<0$$

Bounding box - again

- Image-oriented bounding box:
  - The smallest rectangle around the object, having sides parallell to the edges of the image.
  - Found by searching for min and max x and y within the object (*xmin, ymin, xmax, ymax*)
- Object-oriented bounding box:
  - Smalles rectangle around the object, having one side parallell to the orientation of the object (θ).
  - The transformation

$$\alpha = x\cos\theta + y\sin\theta, \quad \beta = y\cos\theta - x\sin\theta$$

    is applied to all pixels in the object (or its boundary).
  - Then search for α_{min}, β_{min}, α_{max}, β_{max}



What if we want scale-invariance?

- Changing the scale of *f(x,y)* by (α,β) gives a new image:

$$f'(x,y)=f\left(x/\alpha, y/\beta\right)$$

- The transformed central moments

$$\mu'_{pq}=\alpha^{1+p}\beta^{1+q}\mu_{pq}$$

- If α=β, scale-invariant central moments are given by the normalization:

$$\eta_{pq}=\frac{\mu_{pq}}{\left(\mu_{00}\right)^{\gamma}}, \quad \gamma=\frac{p+q}{2}+1, \quad p+q\geq 2$$

Moments as shape features

- The central moments are seldom used directly as shape descriptors.
- Major and minor axis are useful shape descriptors.
- Object orientation is normally not used directly, but to estimate rotation.
- The set of 7 Hu moments can be used as shape features. (Start with the first four, as the last half are often zero for simple objects).

The ”curse-of-dimensionality”

- Also called ”peaking phenomenon”.
- For a finite training sample size, the correct classification rate initially increases when adding new features, attains a maximum and then begins to decrease.
- The implication is that:
- For a high measurement complexity, we will need large amounts of training data in order to attain the best classification performance.
- => 5-10 samples per feature per class.

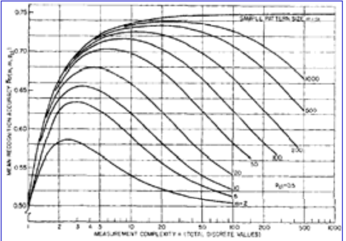


Fig. 3. Finite data set accuracy ($p_{c1} = \frac{1}{2}$).

Correct classification rate as function of feature dimensionality, for different amounts of training data. Equal prior probabilities of the two classes is assumed.

Illustration from G.F. Hughes (1968).