INF 4300 – Digital Image Analysis

SUMMARY – PART I

Fritz Albregtsen 24.11.2010

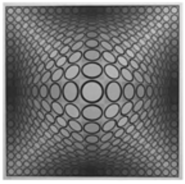
What is texture?

> "Spatially extended patterns of more or less accurate repetitions of some basic texture element, called texels."

- Intuitively obvious, but no precise definition exists
  - ”fine, coarse, grained, smooth” etc
- Texture consists of texture primitives, texels,
  - a contiguous set of pixels with some tonal and/or regional property
- Texture can be characterized by
  - tone, intensity properties of texels
  - structure, spatial relationships of texels
- A texel is the characteristic object that the texture consists of (the ”brick in the wall”)
- Textures are highly scale dependent.

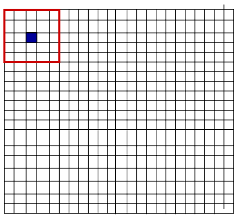
Uses for texture analysis

- Segment an image into regions with the same texture, i.e. as a complement to graylevel or color
- Recognize or classify objects in images based on their texture
- Find edges in an image, i.e. where the texture changes
- ”shape from texture”
- object detection, compression, synthesis
- Industrial inspection:
  - find defects in materials



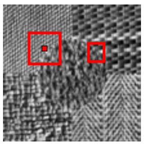
Computing texture images

- Select a window size and a texture measure
- For each pixel (i,j) in the image:
  - Center the window at pixel (i,j)
  - Compute the texture measure
    - One value is computed based on the gray-level variations of pixels inside the image
  - Assign the computed value to the center pixel (i,j) in a new image of the same size
- This is similar to filtering
- Pixels close to the image border can be handled in the same manner as for filtering/convolution



”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 measure of ”smoothness” is

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

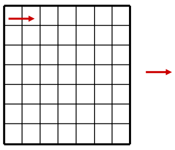
  - R is close to 0 for homogenous areas
  - R tends to 1 as σ^2, ”roughness”, increase

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)
- Choose a distance *d* and a direction *θ*

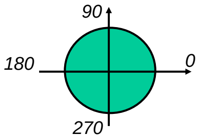


In this example, d=1 and θ=0

- 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*.

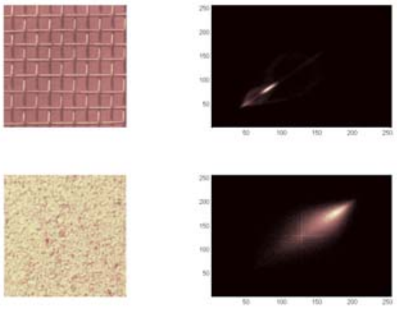
GLCM

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

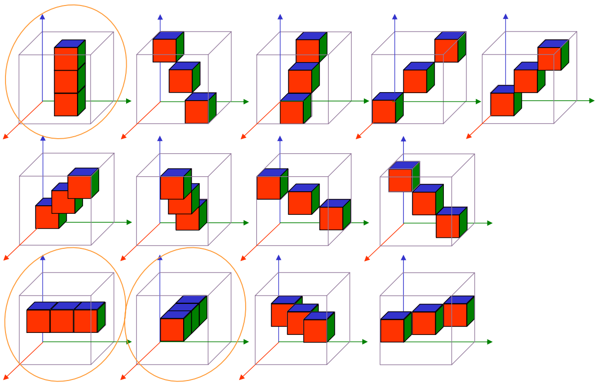


- Isotropic matrix by averaging *P(θ)*, $\theta \in \{0^o, 45^o, 90^o, 135^o\}$
  - *Beware of differences in effective window size!*
- An isotropic texture is equal in all directions
- If the texture has a clear orientation, we select *θ* according to this.

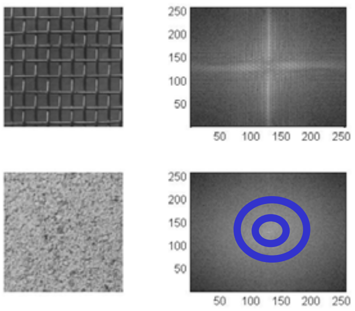
Isotropic GLCM example



3 of 13 symmetric 3D NN

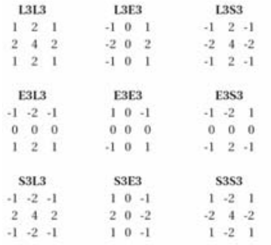


Fourier analysis example



Higher order statistics

- Higher order methods include
  - Gray level runlength matrices - ”histograms” of graylevel run lengths in different directions (INF 5300)
  - Laws’ texture masks – masks resulting from covolutions of line (L), edge (E), spot (S), and ripple (R)



L3L3	L3E3	L3S3
1 2 1	-1 0 1	-1 2 -1
2 4 2	-2 0 2	-2 4 -2
1 2 1	-1 0 1	-1 2 -1
E3L3	**E3E3**	**E3S3**
-1 -2 -1	1 0 -1	-1 -2 1
0 0 0	0 0 0	0 0 0
1 2 1	-1 0 1	-1 2 -1
S3L3	**S3E3**	**S3S3**
-1 -2 -1	1 0 -1	1 -2 1
2 4 2	2 0 -2	-2 4 -2
-1 -2 -1	1 0 -1	1 -2 1

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
- Understand Law’s texture measures and how they are built based on basic filtering operations
- **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*

Improved edge detection

- What if we assume the following:
  - All gradient magnitudes above a strict threshold are assumed to belong to a bona fide edge.
  - All gradient magnitudes above an unstrict threshold and connected to a pixel resulting from the strict threshold are also assumed to belong to real edges.
- Hysteresis thresholding – Canny’s edge detection (see INF 2310).

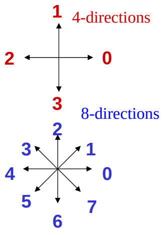
Thinning of edges

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

2 For all nonzero gradient magnitude pixels, inspect the two neighboring pixels in the four (or eight) directions.

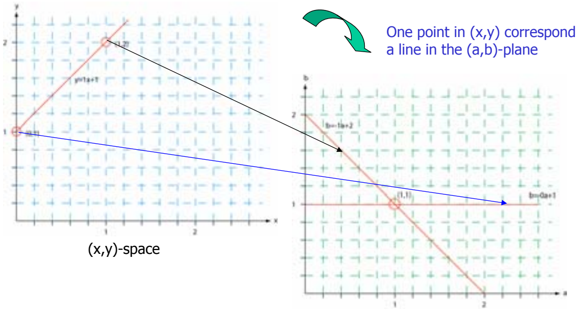
3 If the edge magnitude of any of these neighbors is higher than that under consideration, mark the pixel.

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

Σ: Edge based segmentation

- Advantages
  - An approach similar to how humans segment images.
  - Works well in images with good contrast between object and background
- Disadvantages
  - Does not work well on images with smooth transitions and low contrast
  - Sensitive to noise
  - Robust edge linking is not trivial

Region growing

- Starts with a set of seeds (starting pixels)
  - Predefined seeds
  - All pixels as seeds
  - Randomly chosen seeds
- Region growing steps (bottom-up method)
  - Find starting points
  - Include neighboring pixels with similar features (graylevel, texture, color) A similarity measure must be selected.

Two variants:

1. Select seeds from the whole range of grey levels in the image. Grow regions until all pixels in image belong to a region.
2. Select seed only from objects of interest (e.g. bright structures). Grow regions only as long as the similarity criterion is fulfilled.

- Problems:
  - Not trivial to find good starting points
  - Need good criteria for similarity

Similarity measures

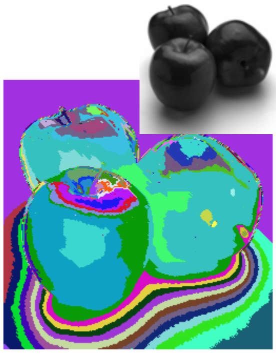
- Graylevel or color?
- Graylevel and spatial properties, e.g., texture, shape, …
- Intensity difference within a region
  (from pixel to seed or to region average so far)
- Within a value range (min, max)
- Distance between mean value of the regions
  (specially for region merging or splitting)
- Variance or standard deviation within a region.

Region merging techniques

- Initialize by giving all the pixels a unique label
  - All pixels in the image are assigned to a region.
- The rest of the algorithm is as follows:
  - In some predefined order, examine the neighbor regions of all regions and decide if the predicate evaluates to true for all pairs of neighboring regions.
  - If the predicate evaluates to true for a given pair of neighboring regions then give these neighbors the same label.
  - The predicate is the similarity measure (can be defined based on e.g. region mean values or region min/max etc.).
  - Continue until no more mergings are possible.
  - Upon termination all region criteria will be satisfied.

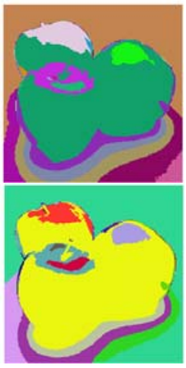
Region merging techniques

- We run a standard region merging procedure where all pixels initially are given a unique label.
- If neighboring regions have mean values within 10 gray levels they are fused.
- Regions are considered neighbors in 8-connectivity.



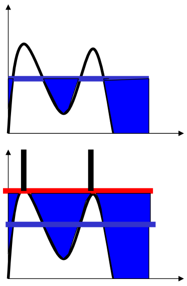
Region merging techniques

- Initialization is critical, results will in general depend on the initialization.
- The order in which the regions are treated will also influence the result:
  - The top image was flipped upside down before it was fed to the merging algorithm.
- Notice the differences!



Watershed segmentation (G&W:10.5)

- Look at the image as a 3D topographic surface, (x,y,intensity), with both valleys and mountains.
- Assume that there is a hole at each minimum, and that the surface is immersed into a lake.
- The water will enter through the holes at the minima and flood the surface.
- To avoid two different basins to merge, a dam is built.
- Final step: the only thing visible would be the dams.
- The connected dam boundaries correspond to the watershed lines.

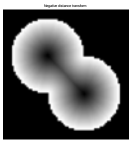


Watershed segmentation

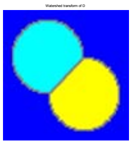
- Can be used on images derived from:
  - The intensity image
  - Edge enhanced image
  - Distance transformed image
    - Thresholded image. From each foreground pixel, compute the distance to a background pixel.
  - Gradient of the image
- Most common: gradient image

Splitting objects by watershed

- Example: Splitting touching or overlapping objects.
  - Given graylevel (or color) image
  - Perform first stage segmentation
    - (edge detection, thresholding, classification,…)
  - Now you have a labeled image, e.g. foreground / background
  - Obtain distance transform image
    - From each foreground pixel, compute distance to background.
  - Use watershed algorithm on inverse of distance image.



Negative distance transform

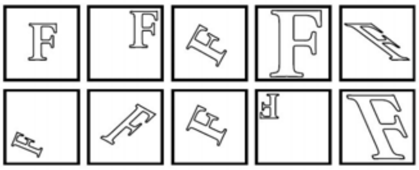


Watershed transform of D

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
  - Interior
- Shape descriptors: methods for characterizing object shapes.
  - The resulting feature values should be useful for discrimination between different object types.

Is invariance needed?



- Translation invariance
- Scale invariance
- Rotation invariance, but what about 6 and 9?
- Warp invariance

- For grey-level regions:
  - invariance to contrast and mean gray level

Shape invariants

- Shape descriptors depend on viewpoint,

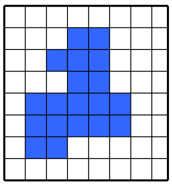
  => object recognition may often be impossible if object or observer changes position.
- Shape description invariance is important
  - shape invariants represent properties which remain unchanged under an appropriate class of transforms.
- Stability of invariants is a crucial property which affects their applicability.
- The robustness of invariants to image noise and errors introduced by image sensors is of prime importance.

Contour representation

- Goal of contour methods is to describe objects in images
- Hopefully, our contour detection method delivers a sequence of pixel coordinates in the image!
- The contour can be represented as:
  - Cartesian coordinates
  - Polar coordinates from a reference point (usually image origin)
  - Chain code and a starting point
    - Connectivity: 4- or 8-neighbors
    - Note: chain code is very sensitive to noise, image resolution and object rotation.
    - Not well suited for classification directly, but useful for computation of other features.

Chains

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

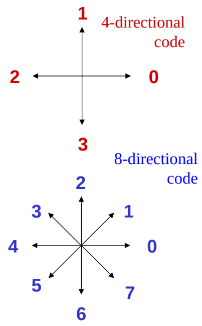


🟦 Object pixel

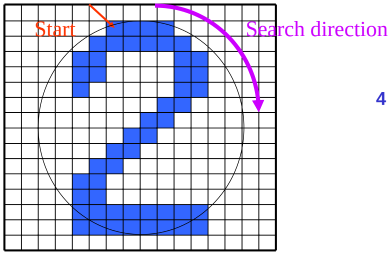
- How do we define border pixels?
  - 4-neighbors
  - 8-neighbors
- In which order do we check the neighbors?
- Chains represent the object pixel, but not their spatial relationship directly

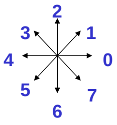
Chain codes

- 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



Chain codes

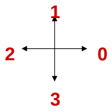




Chain code in clockwise direction:
00077665555556600000064444444222111111223444565221…

Comments about chain code

- The chain code depends on the starting point.
- It can be normalized for start point by treating it as a circular/periodic sequence, and redefine the starting 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: (direction changes between code elements)
  - Code: 10103322
  - First difference: 33133030
  - Minimum circular shift: 03033133
- This invariance is only valid if the boundary itself is invariant to rotation and scale.



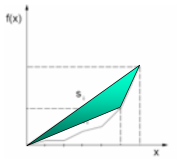
Recursive boundary splitting

- 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.

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:

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

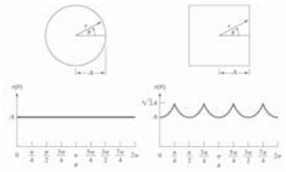
- Let previous point be new breakpoint if
  - area deviation ***A*** per unit length ***s*** of approximating line segment exceeds a specified tolerance, ***T***.

- 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.

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:



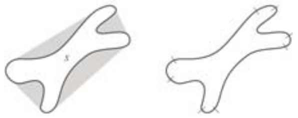
a b

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.



Descriptors extracted from the CH

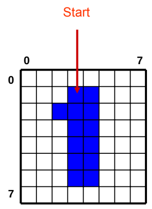
Useful features for shape characterization can be e.g.:

- Area of object and area of convex hull (CH)
  - CH ”solidity” aka ”convexity” = (object area)/(CH area)
    = The proportion of pixels in CH also in the object
  - Better than ”extent” = (object area)/(area of bounding box)
- Number of components of convex deficiency
  - Distribution of component areas
- Relative location of
  - points where we go in and out of the convex deficiency.
  - points of local maximal distance to CH.

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), k∈[0,M-1]
- 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.



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

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

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

- 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.

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.

What is feature extraction?

- Devijver and Kittler (1982):

  ”Extracting from the raw data the information which is most relevant for classification purposes, in the sense of minimizing the within-class pattern variability while enhancing the between-class variability”.

  - Within-class pattern variability: variance between objects belonging to the same class.

  - Between-class pattern variability: variance between objects from different classes.

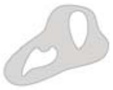
A B C D E F G H I J

Typical image analysis tasks

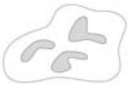
- Preprocessing/noise filtering
- Segmentation
- Feature extraction
  - Are the original image pixel values sufficient for classification, or do we need additional features?
  - What kind of features do we use in order to discriminate between the object classes involved?
- Exploratory feature analysis and selection
  - Which features separate the object classes best?
  - How many features are needed?
- Classification (next)
  - From a set of object examples with known class, decide on a method that separates objects of different types.
  - For new objects: assign each object/pixel to the class with the highest probability
- Validation of classifier accuracy

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”.

Geometric features from contours

- Boundary length/perimeter
- Area
- Curvature
- Diameter/major/minor axis
- Eccentricity
- Bending energy
- Basis expansion (Fourier – last week)

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.

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?

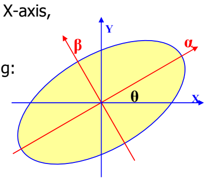
Object orientation - I

- 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

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



- 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 =>

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

$$\Downarrow$$

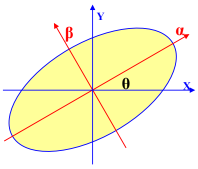
$$\sum_x \sum_y 2f(x,y)\left[xy\left(\cos^2\theta - \sin^2\theta\right)\right] = \sum_x \sum_y 2f(x,y)\left[x^2 - y^2\right]\sin\theta\cos\theta$$

$$\Downarrow$$

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

$$\Downarrow$$

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

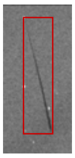


- So the object orientation is given by:

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

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

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

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

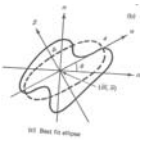
The best fitting ellipse

- Object ellipse is defined as the ellipse whose least and greatest moments of inertia equal those of the object.
- Semi-major and semi-minor axes are given by

$$\left(\hat{a},\hat{b}\right)=\sqrt{\frac{2\left[\mu_{20}+\mu_{02}\pm\sqrt{(\mu_{20}+\mu_{02})^2+4\mu_{11}^2}\right]}{\mu_{00}}}$$

- Numerical eccentricity is given by

$$\hat{\varepsilon}=\sqrt{\frac{\hat{a}^2-\hat{b}^2}{\hat{a}^2}}$$



- Orientation invariant object features.
- Gray scale or binary object.

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:

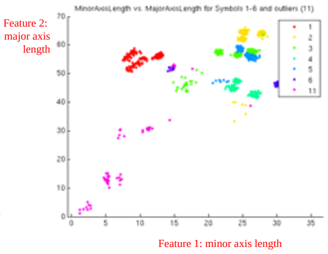
$$\eta_{pq}=\frac{\mu_{pq}}{(\mu_{00})^{\gamma}},\quad \gamma=\frac{p+q}{2}+1,\quad p+q\geq 2$$

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).

Scatter plots

- A 2D scatter plot is a plot of feature values for two different features. Each object's feature values are plotted in the position given by the features values, and with a class label telling its object class.
- Matlab: gscatter(feature1, feature2, labelvector)
- Classification is done based on more than two features, but this is difficult to visualize.
- Features with good class separation show clusters for each class, but different clusters should ideally be separated.



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.

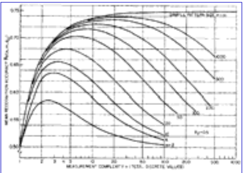


Fig. 3. Finite data set accuracy ($p_c = \frac{1}{2}$).

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).