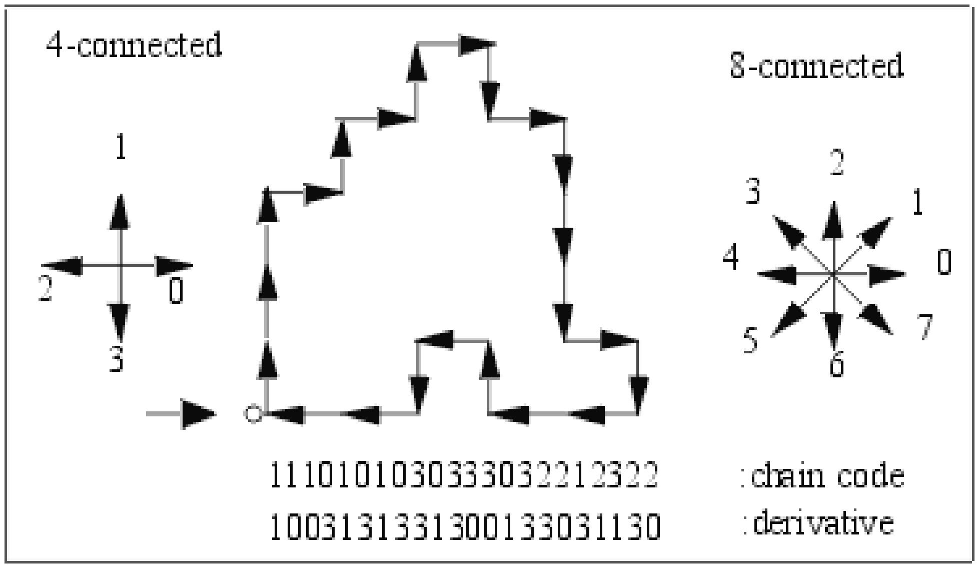
INF 4300 – Digital Image Analysis

OBJECT REPRESENTATION

FROM PIXELS TO REGIONS



Fritz Albregtsen 11.09.2018

Today

G & W Ch. 11.1 Representation

– Curriculum includes lecture notes.

We cover the following:

11.1.1 Boundary following (self study)

11.1.2 Chain Codes

11.1.3 Polygonal Approximations using MPP, Ch 11.1.3 (omitted)

Recursive boundary splitting

Sequential polygonization

11.1.5 Boundary Signatures

11.1.6 Boundary Segments (convex hull)

11.1.7 Skeletons

Fourier representation and Fourier descriptors

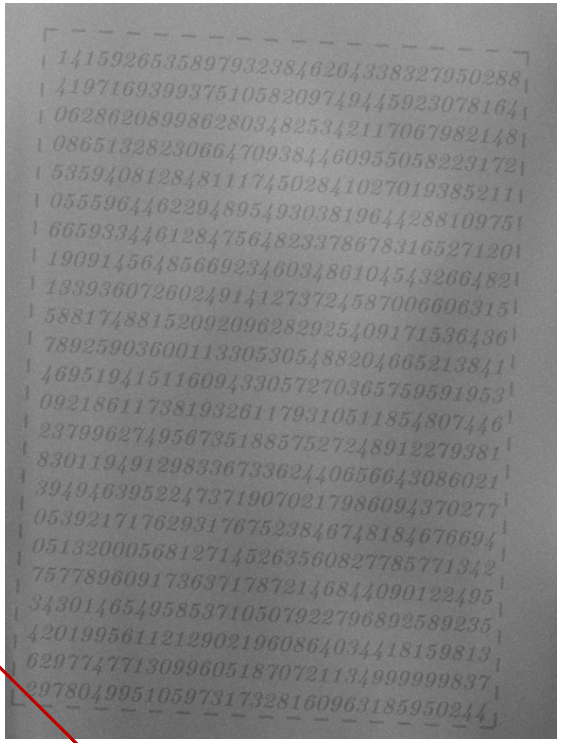
Region representation by Run Length Encoding (RLE)

An example – Creating a program that recognizes the numbers

- Goal: get the series of digits, e.g. 1415926535897……

Steps in the program:

1. Segment the image to find digit pixels.
2. Find angle of rotation and rotate back.
3. Create region objects – one object pr. digit or connected component.
4. Compute features describing the shape of the digits (next week).
5. Train a classifier on many objects of each digit.
6. Classify a new object as the one with the highest probability.



```
1415926535897932384626433832795028 8
4197169399375105820974944592307816 4
0628620899862803482534211706798214 8
0865132823066470938446095505822317 2
5359408128481117450284102701938521 1
0555964462294895493038196442881097 5
6659334461284756482337867831652712 0
1909145648566923460348610454326648 2
1339360726024914127372458700660631 5
5881748815209209628292540917153643 6
7892590360011330530548820466521384 1
4695194151160943305727036575959195 3
0921861173819326117931051185480744 6
2379962749567351885752724891227938 1
8301194912983367336244065664308602 1
3949463952247371907021798609437027 7
0539217176293176752384674818467669 4
0513200056812714526356082778577134 2
7577896091736371787214684409012249 5
3430146549585371050792279689258923 5
4201995611212902196086403441815981 3
6297747713099605187072113499999983 7
2978049951059731732816096318595024 4
```

Focus of this lecture

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:**
  methods to store and represent image objects
  - **Perimeter (and all the methods based on perimeter)**
  - **Interior (and all the methods ...)**

- Shape descriptor (next lecture):
  recipe to extract feature characterizing object shape.
  - The resulting feature value should be useful for discrimination between different object types.

Assumptions for shape representation

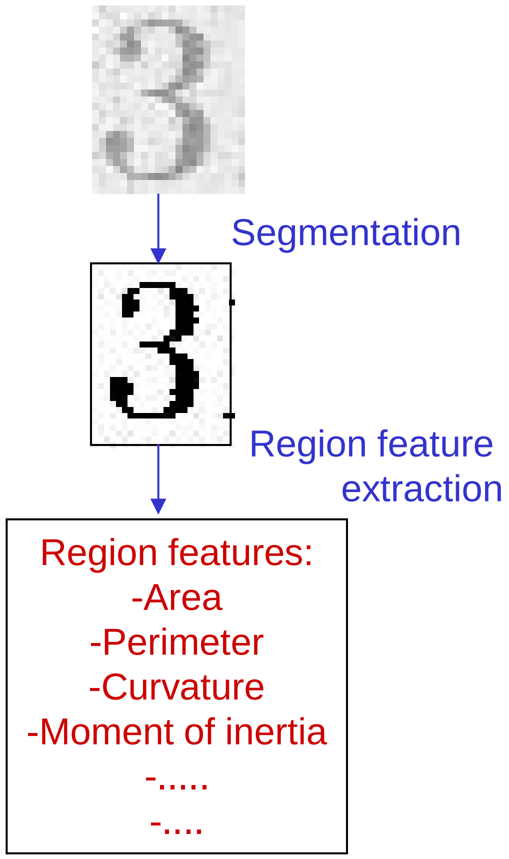
- The image objects can be represented as:
- Whole regions
  - grey level or color image
  - compressed image
  - binary image
- Contours (region boundaries)
  - in cartesian coordinates
  - in polar coordinates
  - in some other coordinates
  - through a chain code / transform
  - as coefficients of some transform (e.g. Fourier)
  - through a run length code / transform

Considerations

- Input representation form, boundaries or whole regions?
- Object reconstruction ability?
- Incomplete shape recognition ability?
- Local/global description?
- Mathematical or heuristic techniques?
- Statistical or syntactic object description?
- Robustness of description to
  translation, rotation, and scale transformations?
- Shape description properties in different resolutions?
  - description changes discontinuously.
- Robustness against
  - image noise
  - geometric sampling (discretization)
  - intensity quantization

From pixels to features – text recognition

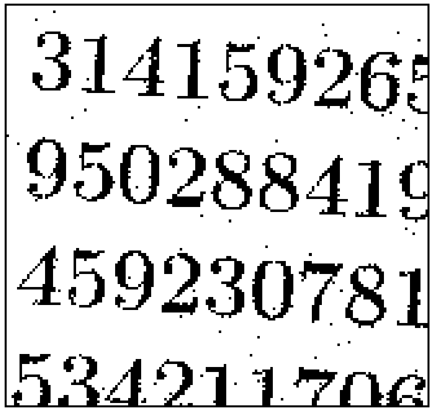
- Input to the classifier is normally a set of features derived from the image data, not the image data itself.
- Why can't we just use all the grey level pixels as they are for text recognition?
  - Objects corresponds to regions. We need the spatial relationship between the pixels.
  - For text recogntion: information is in shape, not in the grey levels.



Characteristics of a good segmentation

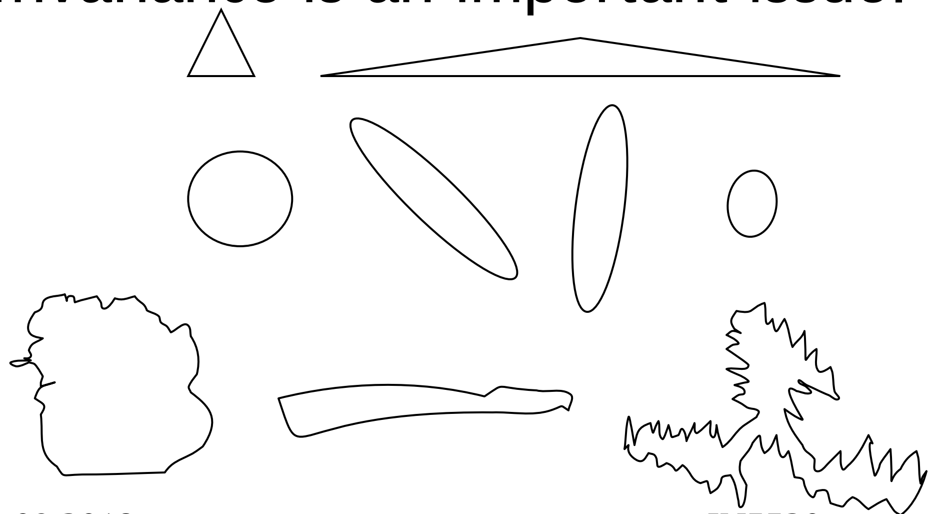
- Each object that is to be described has been identified during segmentation.
  - Ideally, one region in the segmented image should correspond to one object.
  - An object should not be fragmented into several non-connected regions.
  - Some small noise objects will often occurr, these can often be removed later.

  - This lecture and the next will describe how to find connected region objects and how to find scalars or vectors representing their shapes.



What is ”shape”?

- A numerical description of the spatial configuration.
- No generally accepted methodology of shape description.
- High curvature points give essential information.
- Many useful, application-dependent heuristics.
- Shape is often defined in a 2D image, but its usefulness in a 3D world depends on 3D -> 2D mapping.
- Invariance is an important issue.

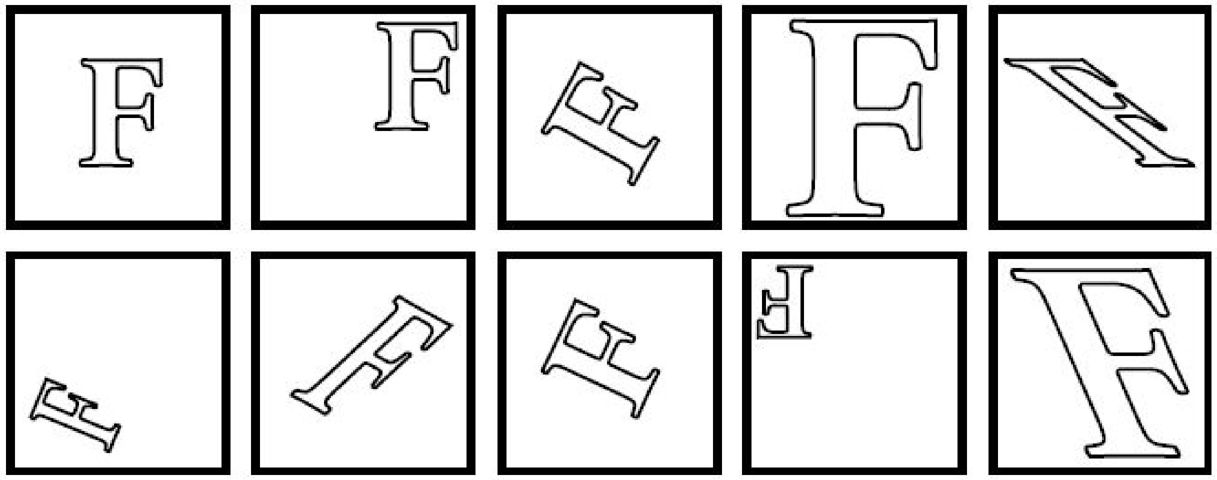


Three curvature points

Ellipses with different size and orientation

Objects with more irregular shape

Is invariance needed?



- Translation invariance
- Scale invariance
- Rotation invariance, but what about e.g., 6 and 9?
- Warp invariance
- You should check INF 2310 on geometrical operations

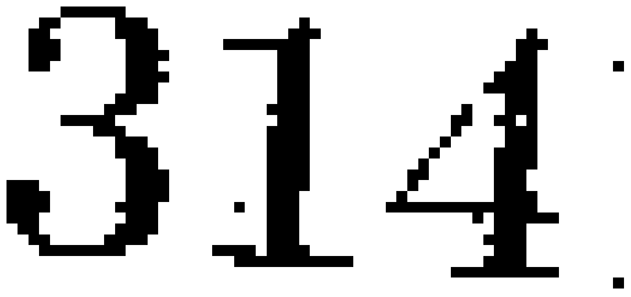
- Gray-level images: invariance to contrast and mean gray level

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.

Creating region objects from segmented image



- Start in upper left corner.
- Find first object pixel.

- Traverse the border of this object
  - (recursive programming efficient)

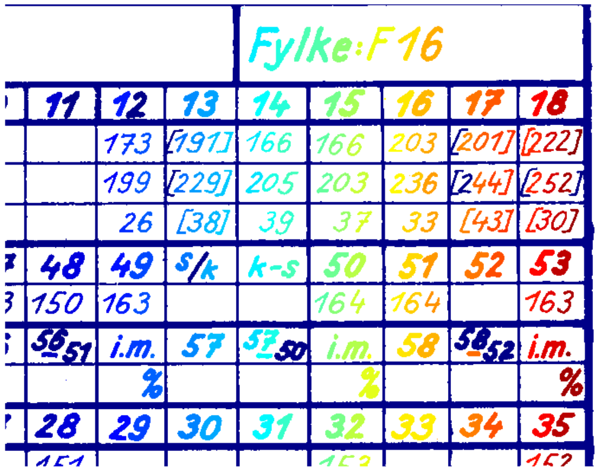
- Continue with other pixels not yet visited

- Implementing this is time consuming
  - (but may be interesting)

Region identification

- After a complete segmentation, the regions must be labeled.
- Search image pixel by pixel, sequentially number each foreground pixel you find according to the labeling of its neighbors.
- The algorithm is basically the same in 4-connectivity and 8-connectivity.
- Result of region identification is a matrix same size as image, with integers representing region label.
- This description of regions will be the input to our shape descriptors

- Matlab function implementing this is *bwlabel*



All pixels in a foreground object (connected Component) get the same label value.

Pixels in the next object get a different label etc.

Generating regions from segmented image

- Matlab: Functions bwlabel and regionprops are the starting points:

B1=bwlabel(B); B is a binary image, obtained by segmentation;

Useful for visualizing regions:

```
CC = bwconncomp(B1);
L = labelmatrix(CC);
RGBim = label2rgb(L); imshow(RGBim)
```

D=regionprops(B1,'area','boundingbox'); Compute area and boundingbox for all regions in the image.

aa=[D.area];

See help regionprops for a list of all descriptors.

Typical use: get a list of pixels for each region

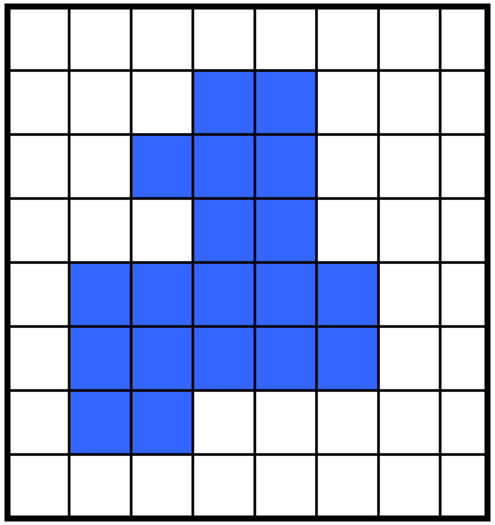
g=bwperim(B,8); Find perimeter pixels of all region in binary image B

Contour representation

- Goal of contour methods is to describe objects in images
- Hopefully, our contour detection method delivers
  a complete 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 (next lecture).

Chains

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

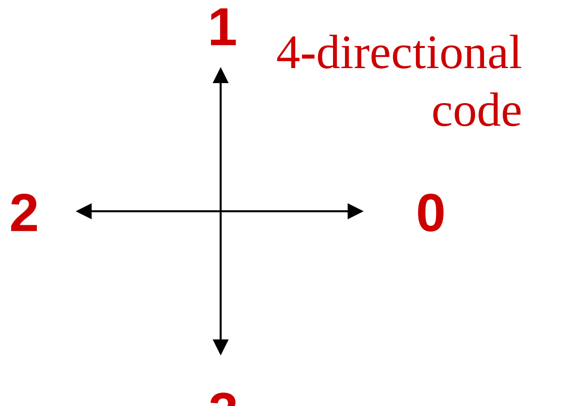


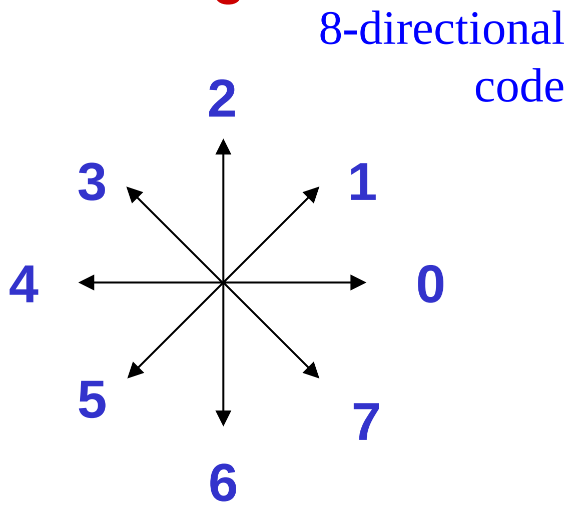
🟦 Object pixel

- 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.

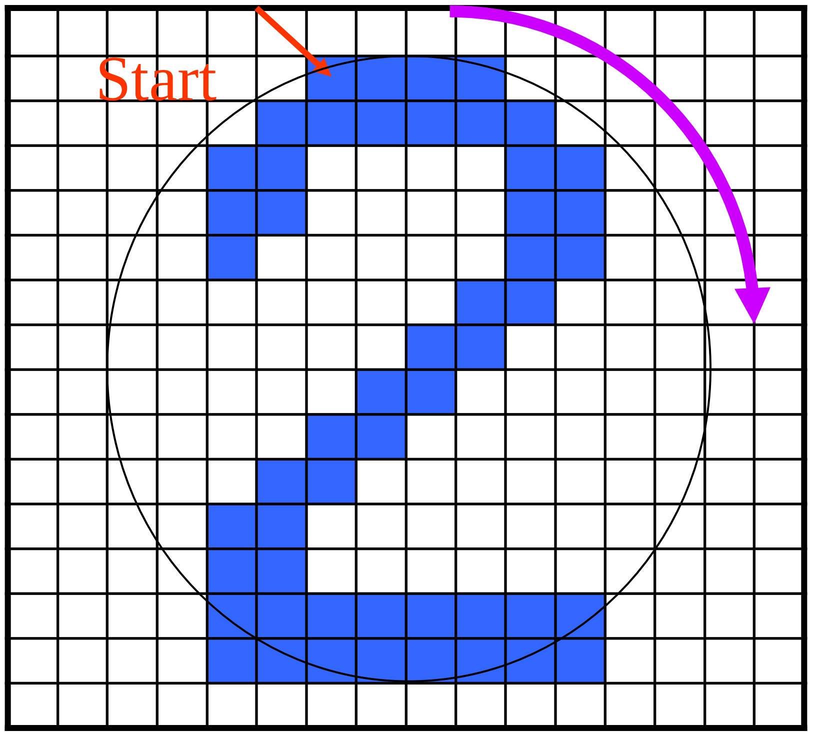
Chain codes

- 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.

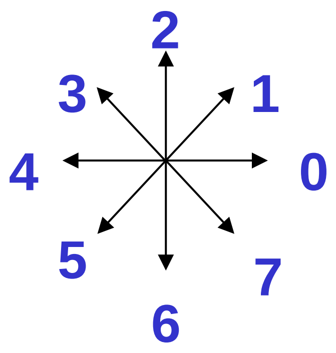




Absolute chain codes



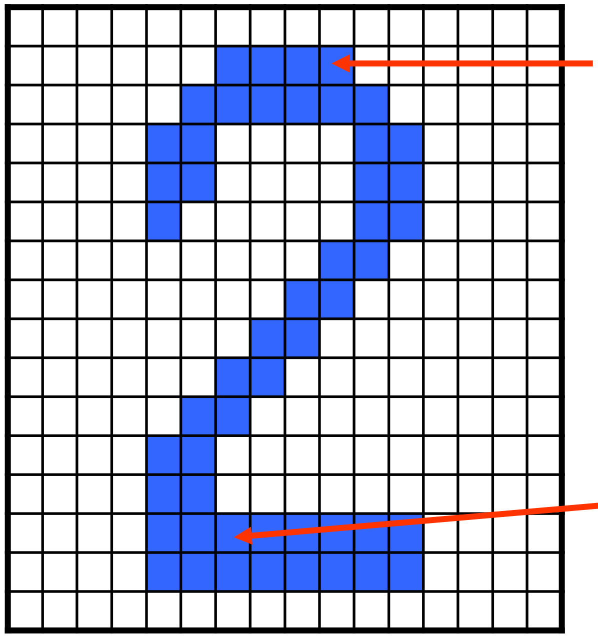
Search direction: look to the left first and check neighbors in clockwise direction



Chain code in clockwise direction:
0007766555556700000644444442221111112234445652211

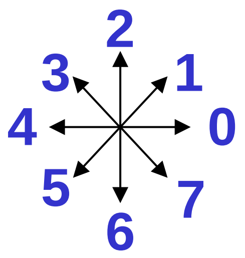
Start point invariance

- 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».



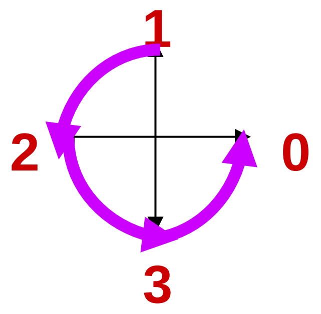
Example:
7766555555670000064444444
2221111112234445652211000

becomes
0000064444444222111111223
4445652211000776655555567
which is here



Rotation invariance

- We can also normalize for rotation by using the first difference of the chain code: (i.e., direction changes between code elements)
  - Example Code: 10103322 (in N_4)
  - 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**



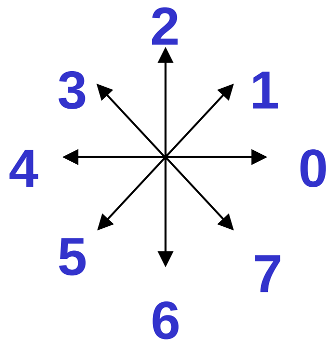
- 00323211 is the same object, only rotated 180 degrees. First difference is 30331330, min.circ.shift is 03033133.

- This invariance is only valid if the boundary itself is invariant to rotation and scale.

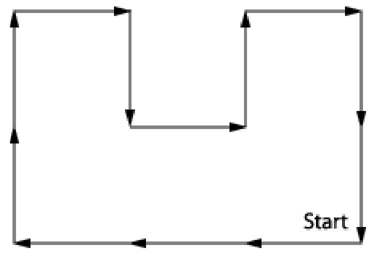
Relative chain code

- 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.*



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

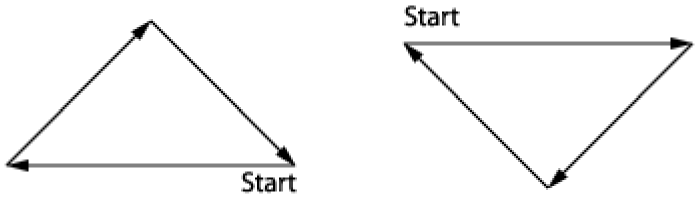


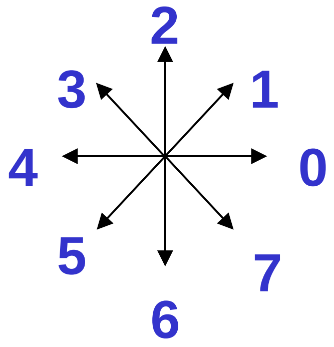
•Why is the relative code:
R,F,F,R,F,R,R,L,L,R,R,F ?

Relative chain code

- The absolute chain code for the triangles are

4,1,7 and 0, 5, 3



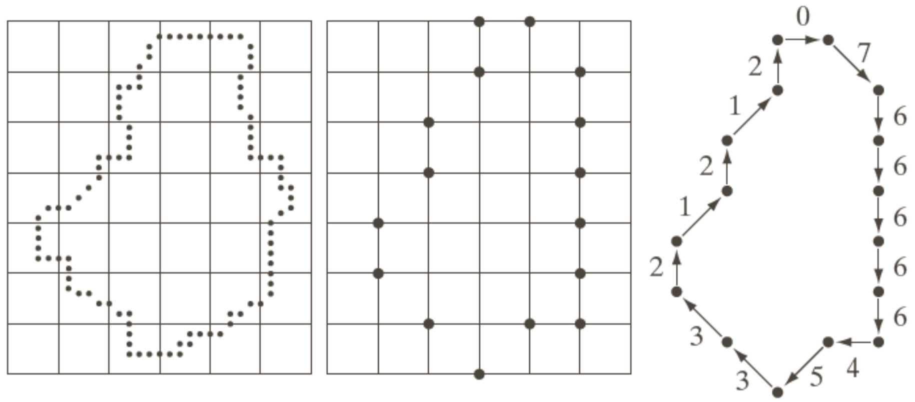


- The relative codes are 7, 7, 0. *(Remember "Forward" is 2)*

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

- Relative code is invariant to rotation, as long as starting point remains the same.
- Start-point invariance by "Minimum circular shift".
- Note: To find the first element, we look back to the end of the contour.

Using all border point vs. resampling the border



- If chain codes are generated from every point on the border, the chain will be long and can be sensitive to small disturbances along the border.
- The border is often resampled to a larger grid before chain coding is done. The accuracy of the chain code depends on the grid size.

Chain code example with smoothing

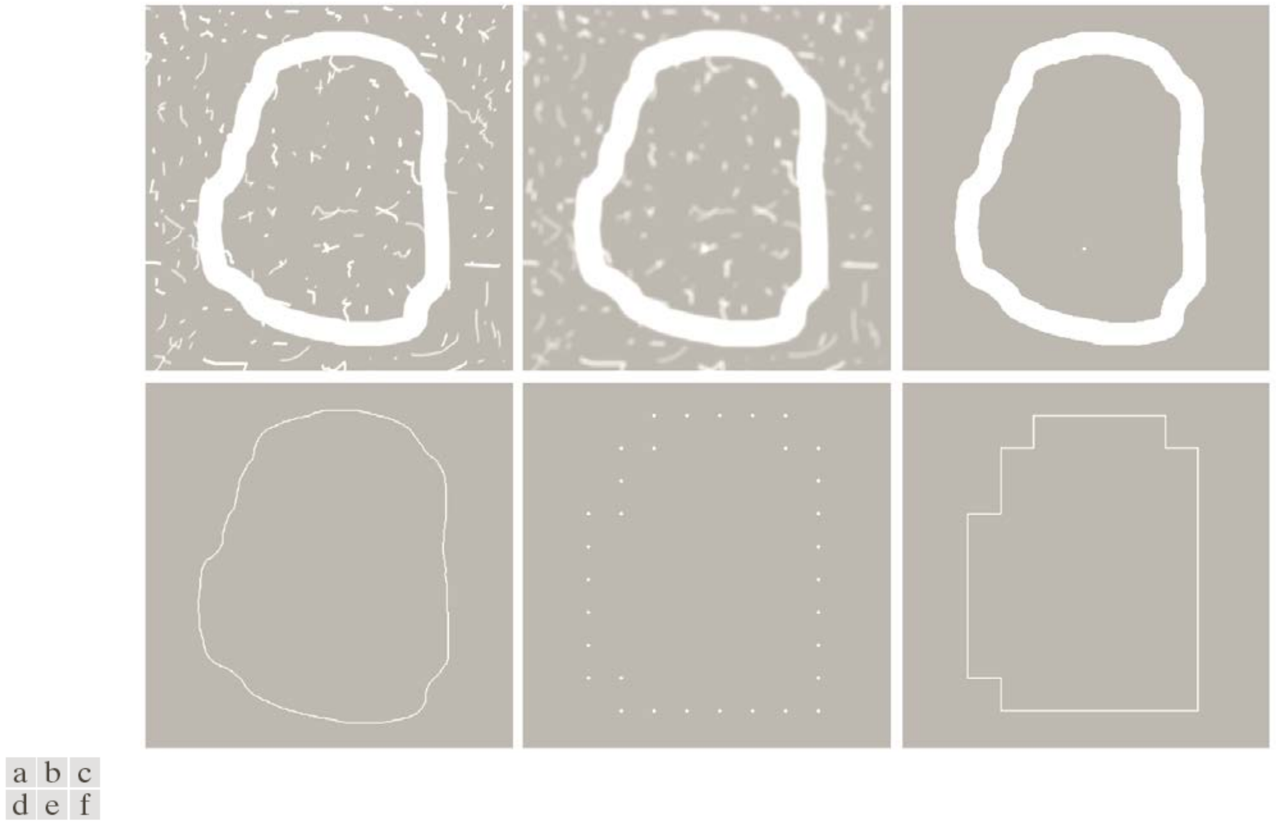


FIGURE 11.5 (a) Noisy image. (b) Image smoothed with a 9 × 9 averaging mask. (c) Smoothed image, thresholded using Otsu's method. (d) Longest outer boundary of (c). (e) Subsampled boundary (the points are shown enlarged for clarity). (f) Connected points from (e).

Polygonal boundary approximation: Recursive boundary splitting (11.1.4)

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

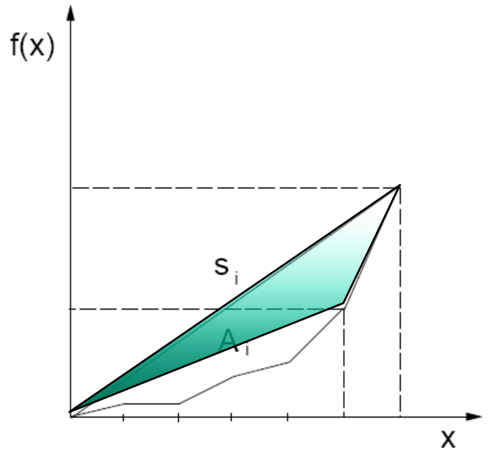
1. For each intermediate point:
2. Compute the point-to-line distance
3. Find the point with the greatest distance from the line.
4. If this distance is greater than given threshold, we have a new breakpoint.
5. The previous line segment is replaced by two, and 1-4 above is repeated for each of them.

- 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:

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

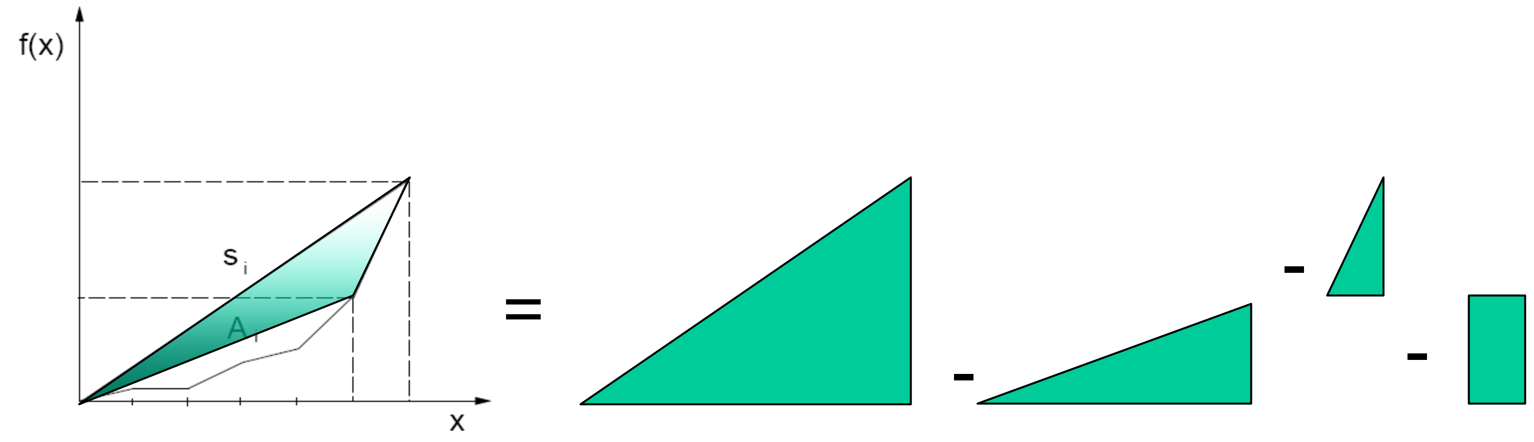


- Let previous point be new breakpoint if
  - area deviation ***A*** per unit length ***s*** of approximating line segment exceeds a specified tolerance, ***T***.

- 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



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

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

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

Signature representations

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

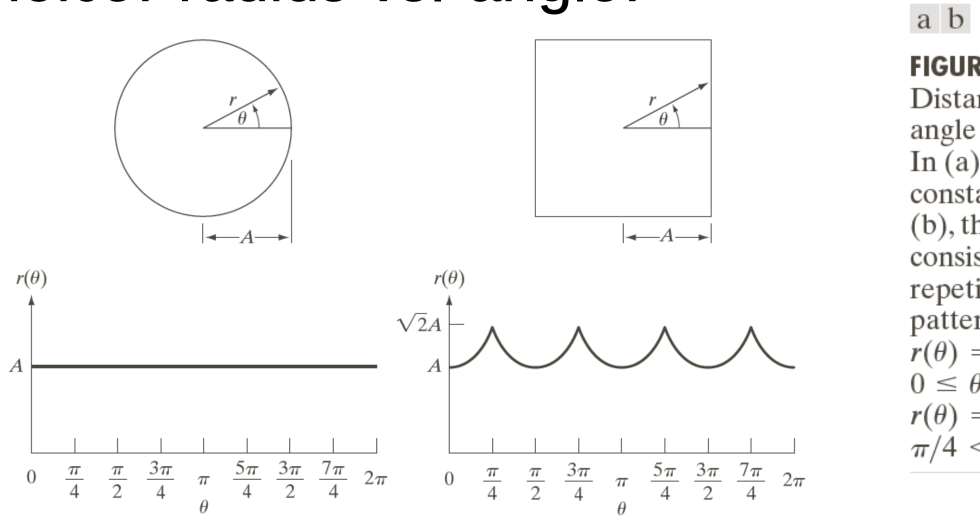


FIGURE 11.10 Distance-versus-angle signatures. In (a) $r(\theta)$ is constant. In (b), the signature consists of repetitions of the pattern $r(\theta) = A \sec \theta$ for $0 \leq \theta \leq \pi/4$ and $r(\theta) = A \csc \theta$ for $\pi/4 < \theta \leq \pi/2$.

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

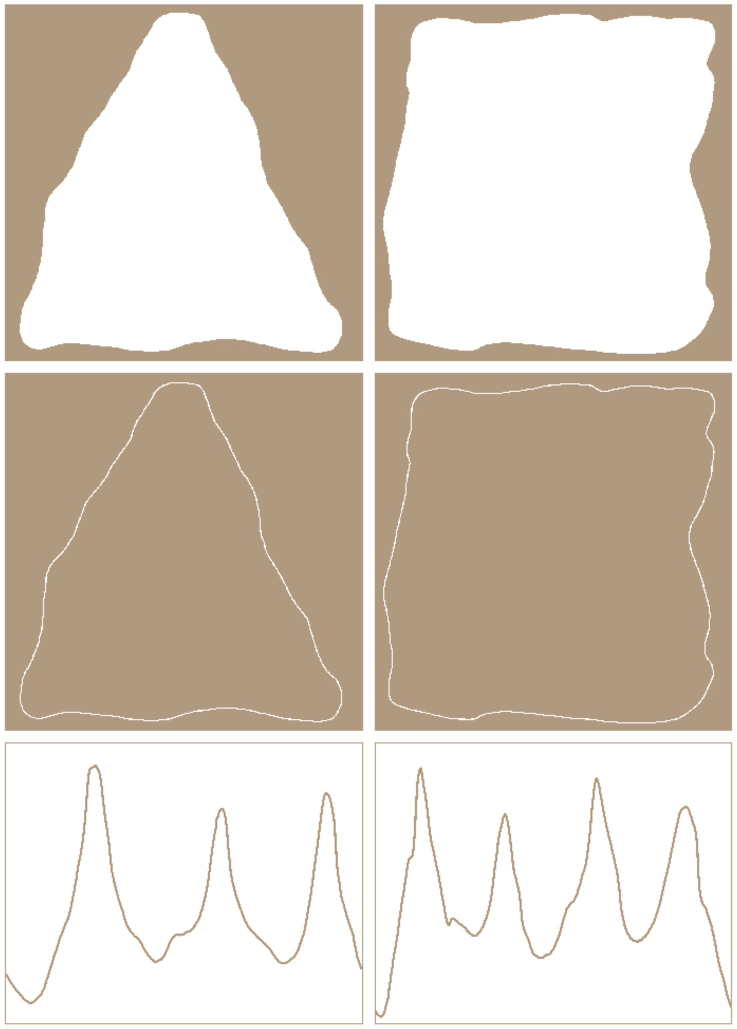
Normalization of (r,θ) signature

- Normalization for starting point and rotation:
  - Find unique point on the boundary according to some criteria:
    - Farthest away from the center (may not be unique)
    - Obtain the chain code and normalize as described for chain codes.

- Normalization for scale:
  - If the boundary is sampled at equal intervals in angle, changes in size only affect the amplitude (*r*).
    - The function can be scales so that it always spans the range [0,1].
      - This is not robust for extreme values of min/max.
    - Scaling by subtracting the mean and dividing by the standard deviation is more robust.

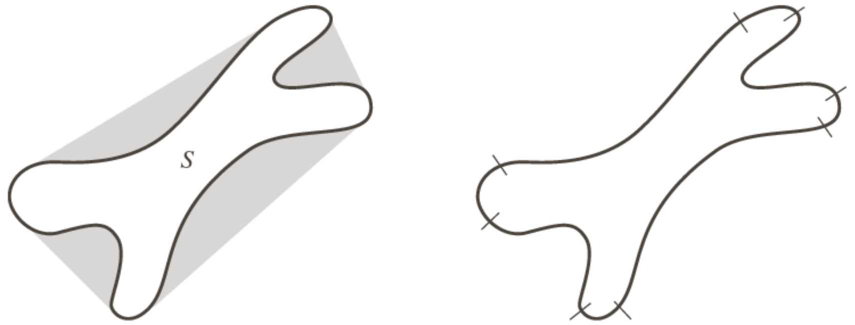
$$\frac{r-\mu}{\sigma}$$

Signature example



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 Convex Hull

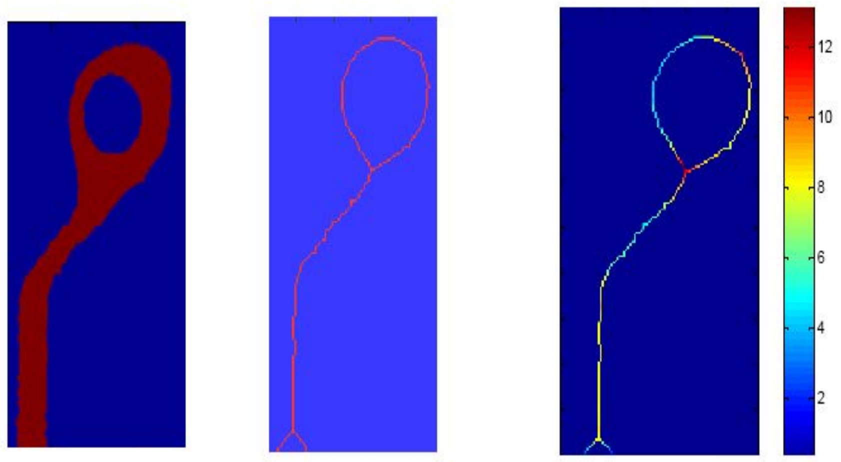
Useful features for shape characterization can be e.g.:

- 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.

Skeletons

- The skeleton of a region is defined by the medial axis transform: For a region R with border B, for every point p in R, find the closest neighbor in B.
- If p has more than one such neighbor, it belong to the medial axis.
- The skeleton S(A) of an object is the axis of the object.
- The medial axis transform gives the distance to the border.



Thinning algorithm to find the skeleton

- There are many algorithms in the literature, here is just one.
- Assume that region points have value 1 and background points 0.
- Define a contour point as a point that has value 1 and at least one 8-neighbor with value 0.
- Apply successive passes of two steps to contour points of region.
- Step 1: *Flag* contour point p_1 for deletion if:

  a) $2 \leq N(p_1) \leq 6$ (N=1 => end point)

  b) $T(p_1) = 1$ (T=1 => 1 pixel thick)

  c) $p_2 \cdot p_4 \cdot p_6 = 0$

  d) $p_4 \cdot p_6 \cdot p_8 = 0$

  Where $N(p_1)$ is the number of nonzero neighbors of p_1.

  $T(p_1)$ is the number of 0-1 transitions in the ordered sequence $p_2, p_3, \ldots, p_8, p_9, p_2$.

p_9	p_2	p_3
p_8	p_1	p_4
p_7	p_6	p_5

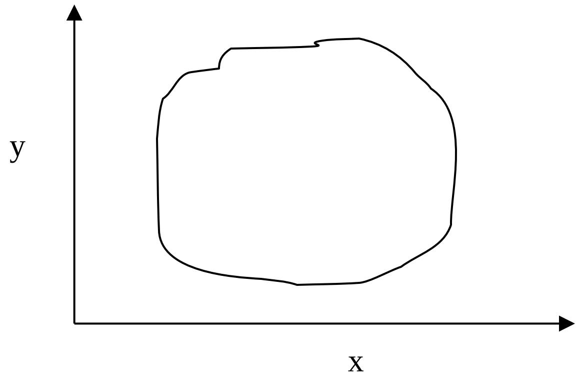
Thinning algorithm – cont.

- Step 2: change condition c) and d) to

  c') $p_2 \cdot p_4 \cdot p_8 = 0$

  d') $p_2 \cdot p_6 \cdot p_8 = 0$
- Step 1 is applied to every border pixel in the region under consideration. If any of the conditions a)-d) are violated, the value of the point is unchanged.
  - If all conditions are satisfied, the point is flagged.
  - The point is not deleted until all points have been satisfied.
  - After completion of step 1, the flagged points are deleted.
- Step 2 is then applied in the same manner as step 1.
- Step1+Step2 defines one iteration of the algorithm.
- More iterations are done
  until no further points are deleted.

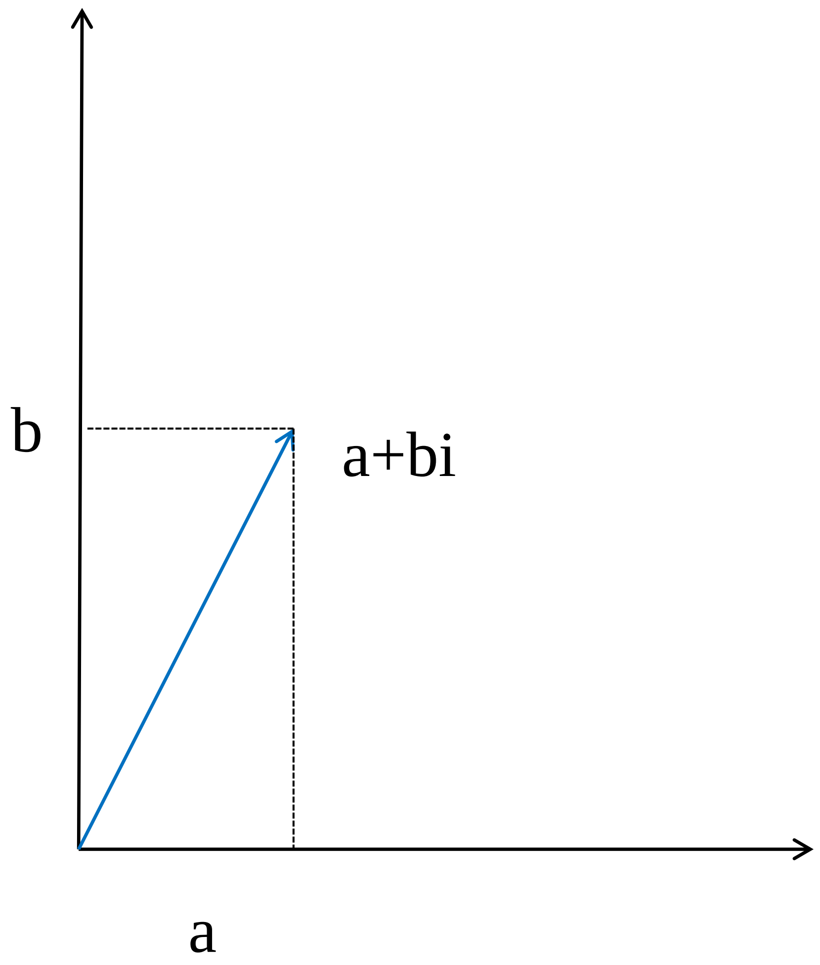
Introduction to Fourier descriptors



- Suppose that we have an object *S* and that we are able to find the length of its contour. The contour should be a closed curve.
- We partition the contour into *M* segments of equal length, and thereby find *M* equidistant points along the contour of *S*.
- Traveling anti-clockwise along contour at constant speed, we can collect pairs of coordinates, x(k) and y(k). Any 1D signal representation can be used for these.
- If the speed is such that one circumnavigation of the object takes 2π, x(k) and y(x) will we periodic with period 2π.

Reminder: complex numbers

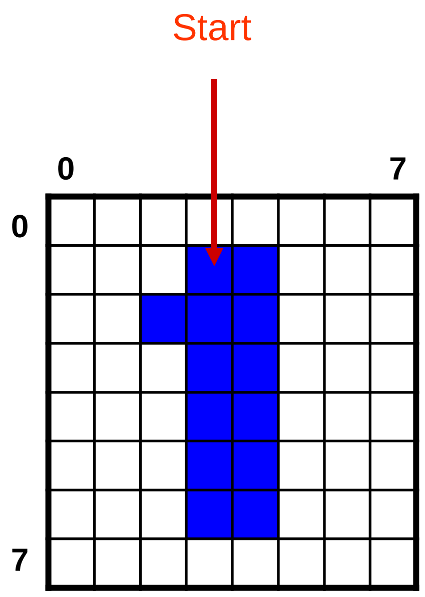
- a+bi



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.



$s(1)=3+1i$

$s(2)=2+2i$

$s(3)=3+3i$

$s(4)=3+4i$

M

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

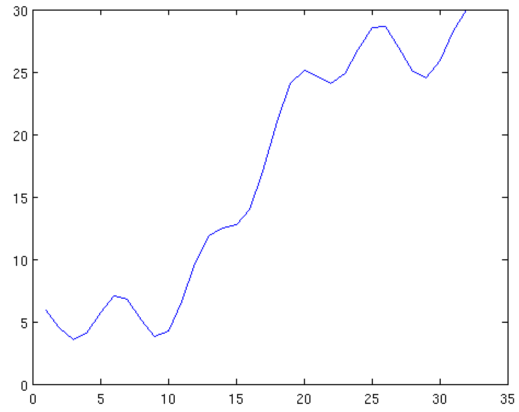
$$a(u) = \frac{1}{M}\sum_{k=0}^{M-1} s(k)\exp\left(\frac{-2\pi iuk}{M}\right) = \frac{1}{M}\sum_{k=0}^{M-1} s(k)\left(\cos\left(\frac{2\pi uk}{M}\right) - i\sin\left(\frac{2\pi uk}{M}\right)\right), \qquad u \in [0, M-1]$$

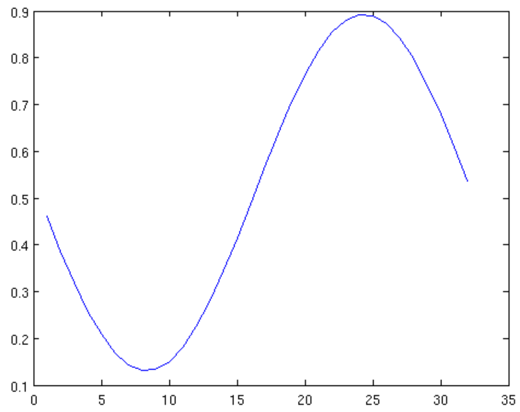
- Complex coefficients *a(u)* are the Fourier representation of boundary.
- *a(0)* contains the center of mass of the object.
- Exclude *a(0)* as a feature for object recognition.
- a*(1),* a*(2),….,a(M-1)* will describe the object in increasing detail.
- These depend on rotation, scaling and starting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

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

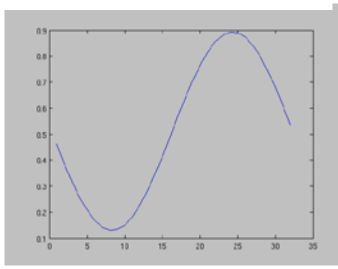
Approximating a curve with Fourier coefficients in 1D

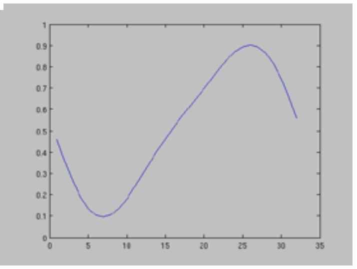
- Do the Fourier transform of the signal of length M
- Keep only N (<M/2) first coefficients (set the others to 0 in amplitude).
- Compute the inverse 1D Fourier transform of the modified signal.
- Display the reconstructed M-point signal based on only N coefficients.

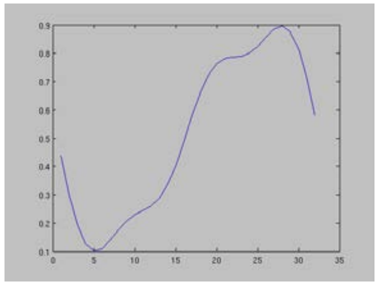

Original signal of length N=36

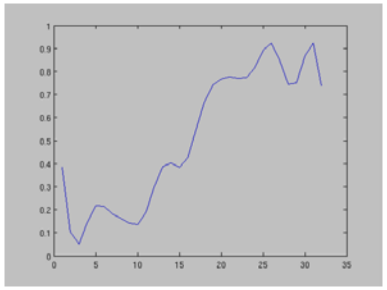

Reconstructed signal using only 2 Fourier coefficients

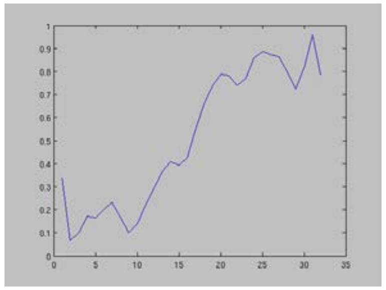
Approximating in increasing detail

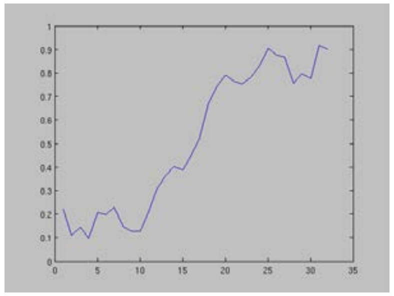

m=2

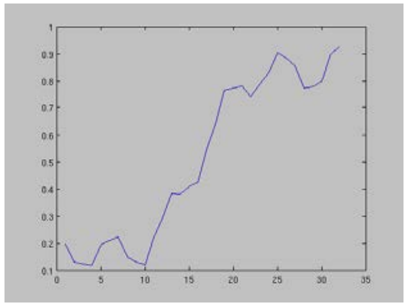

m=3

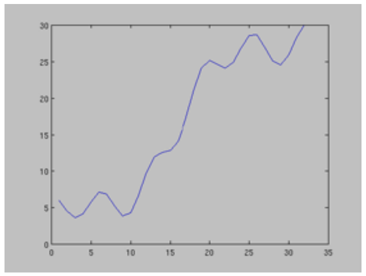

m=4


m=8

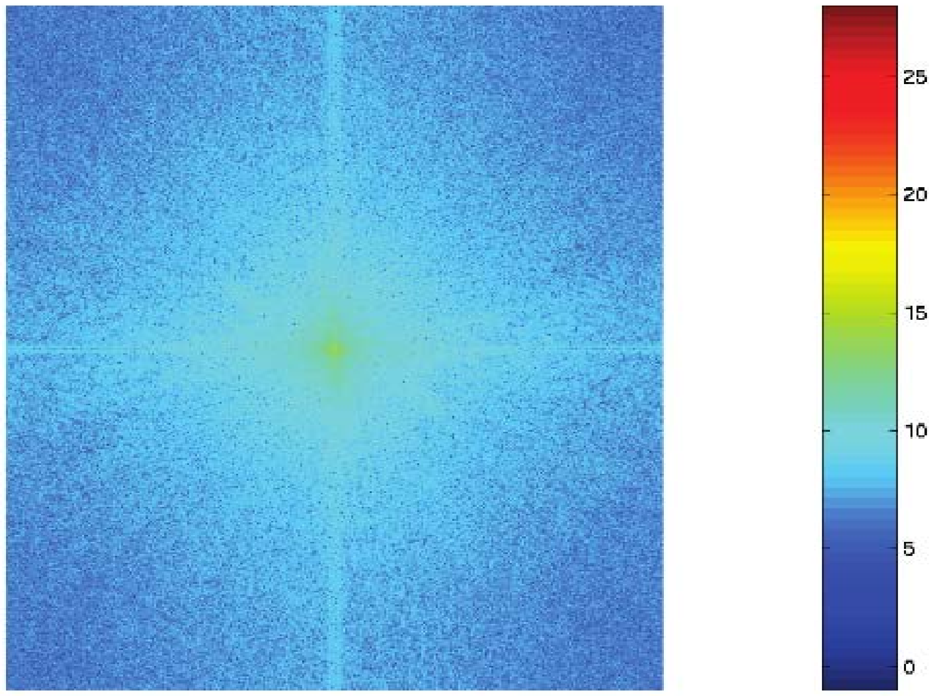

m=10


m=13


m=15


Original

Look back to 2D Fourier spectra (2310)



Most of the energy is concentrated along the lowest frequencies. We can reconstruct the image with an increasing accuracy by starting with the lowest frequencies and adding higher frequencies.

Fourier Symbol reconstruction

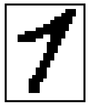
- Inverse Fourier transform gives an approximation to the original contour

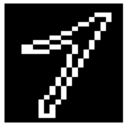
$$\hat{s}(k) = \sum_{k=0}^{N-1} a(u) \exp\left(\frac{2\pi iuk}{M}\right), \qquad k \in [0, M-1]$$

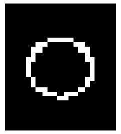
- We have only used N features to reconstruct each component of $\hat{s}(k)$.
- The number of points in the approximation is the same (M),
  but the number of coefficients (features) used to reconstruct each point is smaller ($N<M$).

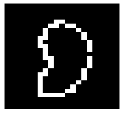
- Use an even number of descriptors.
- The first 10-16 descriptors are found to be sufficient for character description. They can be used as features for classification.
- The Fourier descriptors can be invariant to translation and rotation if the coordinate system is appropriately chosen.
- All properties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

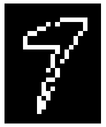
Fourer descriptor example

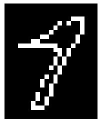

Image, 26x21 pixels

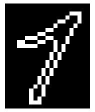

Boundary


2 coefficients


4 coefficients


6 coefficients


8 coefficients


20 coefficients

Matlab DIPUM Toolbox:

```
b=boundaries(f);
b=b{1};
%size(b) tells that the contour is 65
    pixels long
bim=bound2im(b,26,21);
%must tell image dimension
z=frdescp(b);
zinv2=ifrdesc(z,2);
z2im=bound2im(zinv2,26,21);
imshow(z2im);
```

Fourier coefficients and invariance

- Translation affects only the center of mass (a(0)).
- Rotation only affects the phase of the coefficients.
- Scaling affects all coefficients in the same way,
  so ratios $a(u_1)/a(u_2)$ are not affected.
- The start point affects the phase of the coefficients.
- Normalized coefficients can be obtained,
  but is beyond the scope of this course.
  - See e.g., Ø. D. Trier, A. Jain and T. Taxt,
    Feature extraction methods for character recognition – a survey.
    Pattern Recognition, vol. 29, no. 4, pp. 641-662, 1996.

Examples from Trier et al. 1996

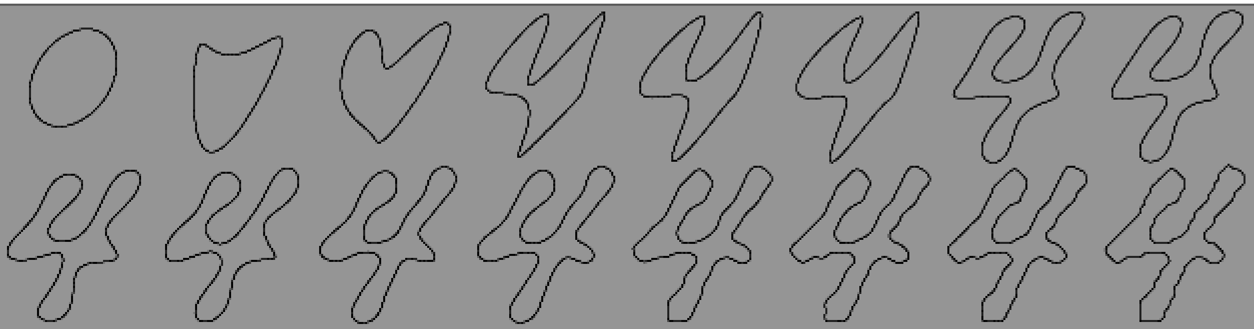


Figure 18: Character '4' reconstructed by elliptic Fourier descriptors of orders up to 1, 2, . . . 10; 15, 20, 30, 40, 50, and 100, respectively.

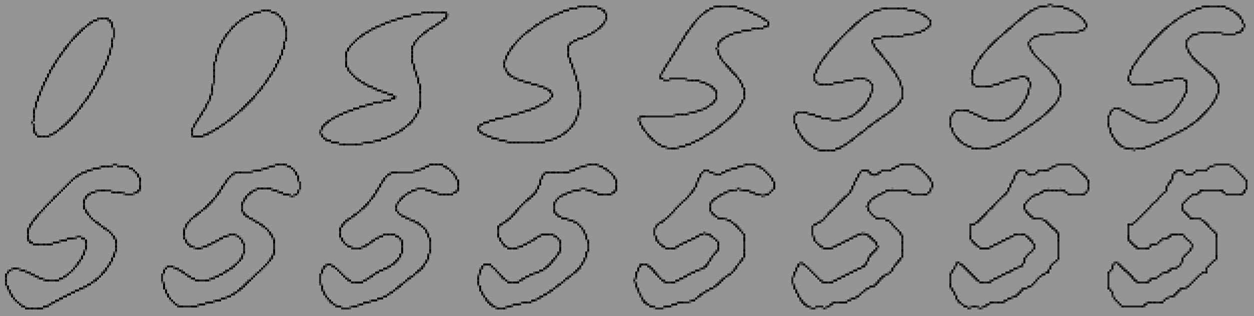
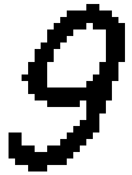


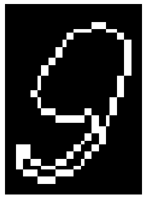
Figure 19: Character '5' reconstructed by elliptic Fourier descriptors of orders up to 1, 2, . . . 10; 15, 20, 30, 40, 50, and 100, respectively.

Remark about Fourier descriptors

Note that the contour of



is



Objects with holes, like 8, 9, 6 will only by described by the outer contour.

The length of a(u) will depend on the number of boundary points (and the size of the object).

We should resample the contour by selecting M evenly spaced points along the contour.

Run Length Encoding of Objects

- See also GW 8.2.5
- 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.

“Gray code”

Is the conventional binary representation of graylevels optimal?

- Consider a single band graylevel image having b bit planes.
- We desire a minimum complexity in each bit plane
  - Because the run-length transform will be most efficient.
- Conventional binary representation gives high complexity.
  - If the graylevel value fluctuates between 2^k-1 and 2^k, k+1 bits will change value: example: 127 = 01111111 while 128 = 10000000
- In ”Gray Code” only one bit changes if graylevel is changed by 1.
- The transition from binary code to ”gray code” is a reversible transform, while both binary code and ”gray code” are codes.

Gray Code transforms

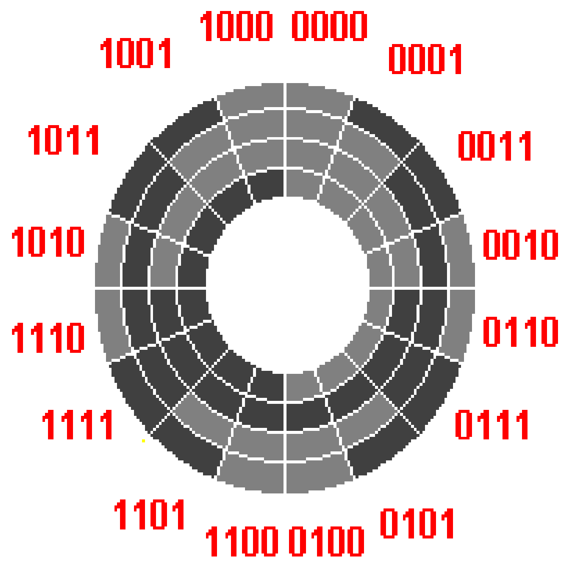
- ”Binary Code” to ”Gray Code”:
  1. Start by MSB in BC and keep all 0 until you hit 1
  2. 1 is kept, but all following bits are complemented until you hit 0
  3. 0 is complemented, but all following bits are kept until you hit 1
  4. Go to 2.

- ”Gray Code” to ”Binary Code”:
  1. Start by MSB in GC and keep all 0 until you hit 1
  2. 1 is kept, but all following bits are complemented until you hit 1.
  3. 1 is complemented, but all following bits are kept until you hit 1.
  4. Go to 2.

”Binary reflected gray code”

4 bits Gray code and binary

Gray	Binary	Decimal
0000_g	0000_b	0_d
0001	0001	1
0011	0010	2
0010	0011	3
0110	0100	4
0111	0101	5
0101	0110	6
0100	0111	7
1100	1000	8
1101	1001	9
1111	1010	10
1110	1011	11
1010	1100	12
1011	1101	13
1001	1110	14
1000	1111	15



”Gray code shaft encoder”

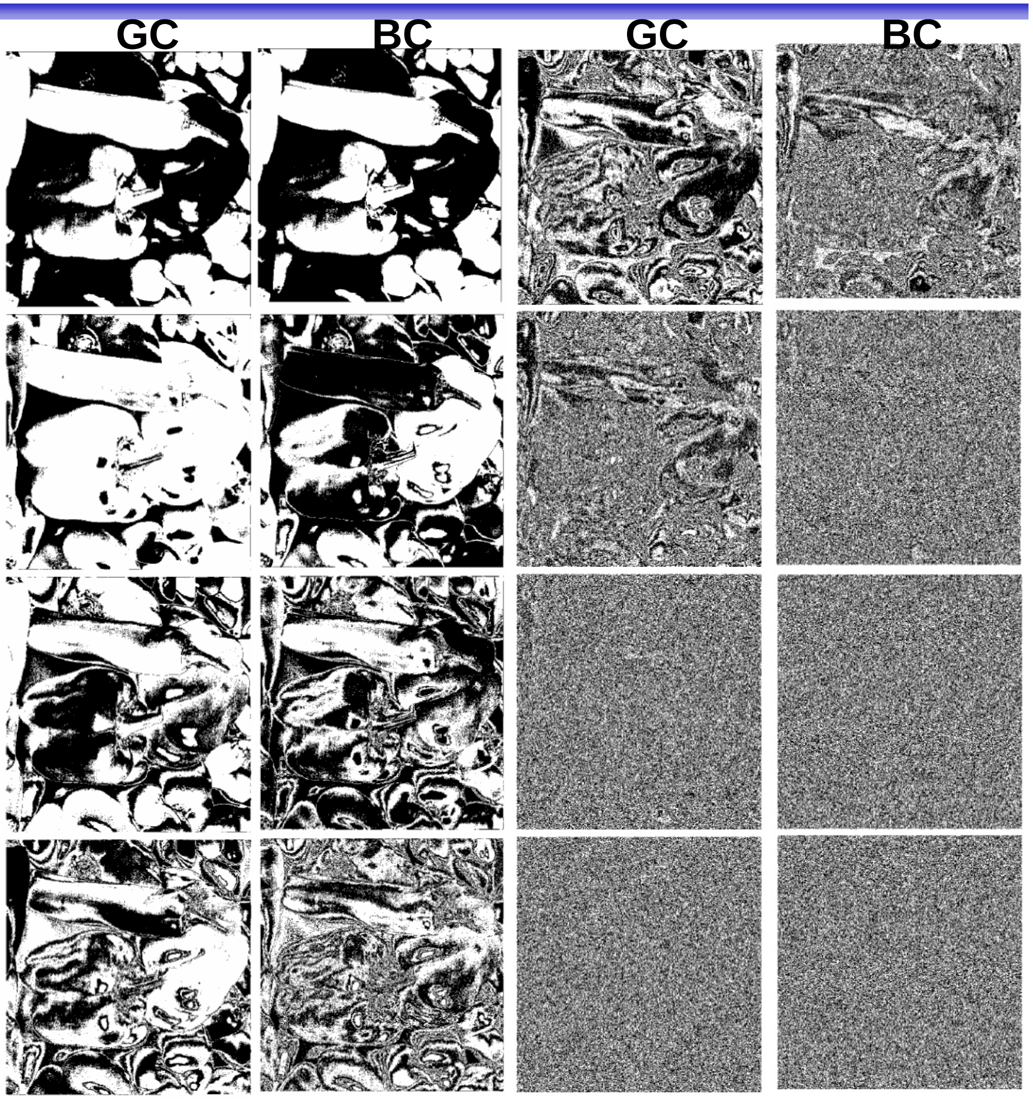
Gives secure reading of angle.

Émile Baudot’s telegraph 1878.

Code patented by Gray in 1953.

Gray code in graylevel images

- MSB is the same in the two representations.
- Larger homogeneous areas in each bitplane in Gray Code than in natural binary code.
- Less noisy bitplanes in Gray Code.
- More compact run-length coding of bitplanes in Gray Code.



Learning goals - representation

- Understand difference between representation and description.
- Chain codes
  - Absolute
    - First difference
  - Relative
    - Minimum circular shift
- Polygonization
  - Recursive and sequential
- Signatures
- Convex hull
- Skeletons
  - Thinning
- Fourier descriptors
- Run Length Encoding