WILEY
A JOHN WILEY & SONS, INC., PUBLICATION

Practical Image and Video Processing Using MATLAB®

Chapter 18 – Feature extraction and representation

By Oge Marques

What will we learn?

- What is feature extraction and why is it a critical step in most computer vision and image processing solutions?

- Which types of features can be extracted from an image and how is this usually done?

- How are the extracted features usually represented for further processing?

Introduction

- Feature extraction is the process by which certain features of interest within an image are detected and represented for further processing.
- It is a critical step in most computer vision and image processing solutions.
  - It marks the transition from pictorial to non-pictorial (alphanumerical, usually quantitative) data representation.
  - The resulting representation can be subsequently used as the input to a number of pattern recognition and classification techniques which will then label, classify, or recognize the semantic contents of the image or its objects.

Introduction

- Most techniques presented in this chapter assume that an image has undergone segmentation.
- Goal of feature extraction and representation techniques: to convert the segmented objects into representations that **better describe** their main features and attributes.
- There are **many ways** an image (and its objects) can be represented for image analysis purposes.
- In this topic we present several representative techniques for feature extraction using a broad range of image properties.

Feature vectors and vector spaces

- A *feature vector* is a $n \times 1$ array that encodes the n features (or measurements) of an image or object.
- The array contents may be:
  - Symbolic
  - **Numerical**
  - Both

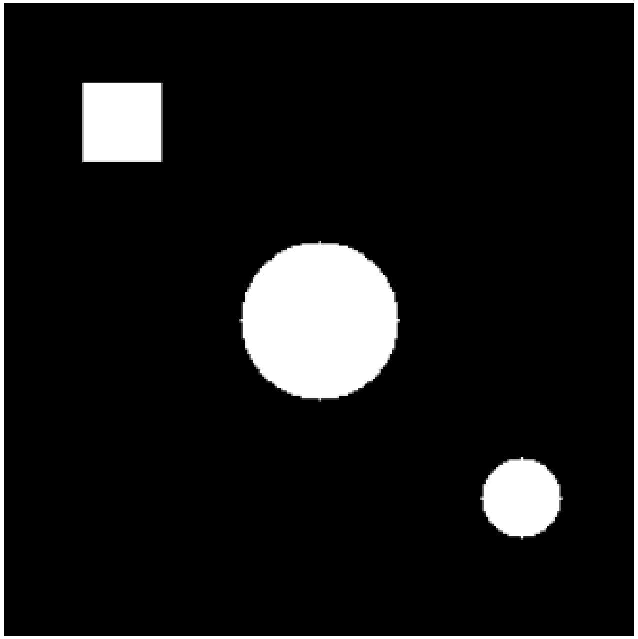
$$\mathbf{x} = (x_1, x_2, \cdots, x_n)^T$$

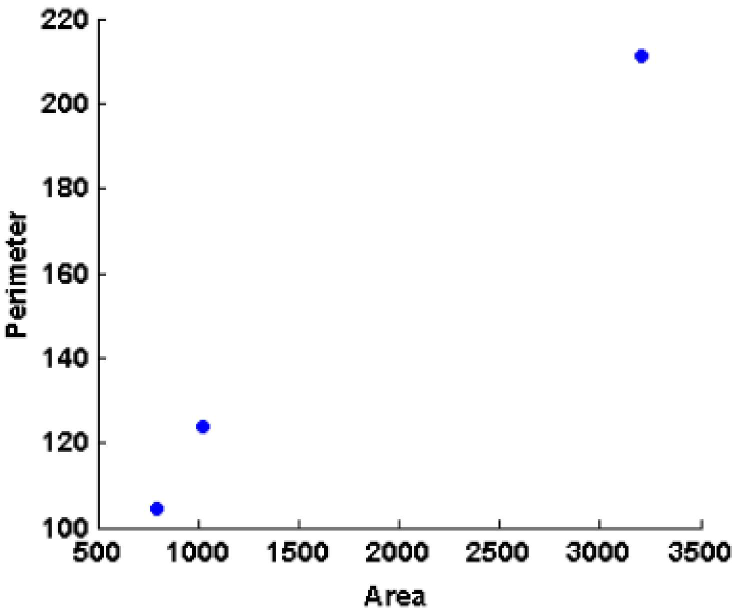
- The FV is a compact representation of an image (or object), which can be associated with the notion of a feature space, an n-dimensional *hyperspace* that allows the visualization (for $n < 4$) and interpretation of the feature vectors' contents, their relative distances, etc.

Feature vectors and vector spaces

- Example 18.1

Object	Area	Perimeter
Square (Sq)	1024	124
Large circle (LC)	3209	211
Small circle (SC)	797	105





By Oge Marques

Invariance and robustness

- It is often required that the features used to represent an image be invariant to rotation, scaling, and translation (RST).
  - RST invariance ensures that a machine vision system will still be able to recognize objects even when they appear at different size, position within the image, and angle (relative to a horizontal reference).
  - Clearly this requirement is application-dependent.

Binary object features

- Notation:
  - A binary object, in this case, is a connected region within a binary image *f(x, y)*, which will be denoted as O_i, $i > 0$.
  - In MATLAB: **`bwlabel`**
  - Mathematically:

$$O_i(x, y) = \begin{cases} 1 & \text{if } f(x, y) \in O_i \\ 0 & \text{otherwise} \end{cases}$$

Binary object features

- Area:

$$A_i = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} O_i(x, y)$$

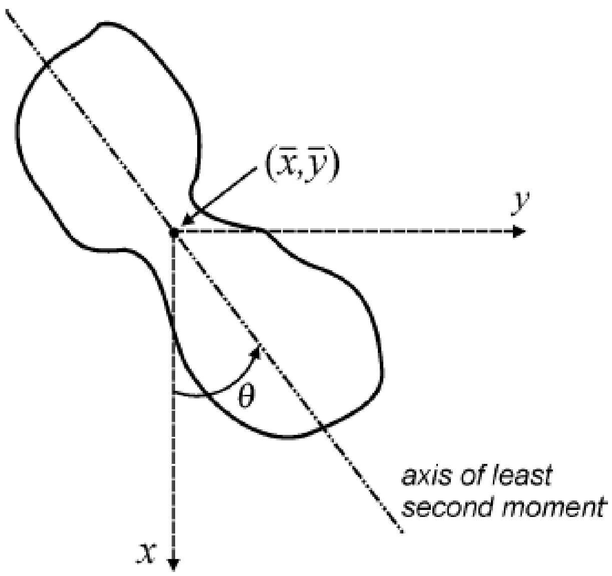
- Centroid:

$$\bar{x}_i = \frac{1}{A_i} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} x O_i(x, y)$$

$$\bar{y}_i = \frac{1}{A_i} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} y O_i(x, y)$$

Binary object features

- Axis of least second moment:



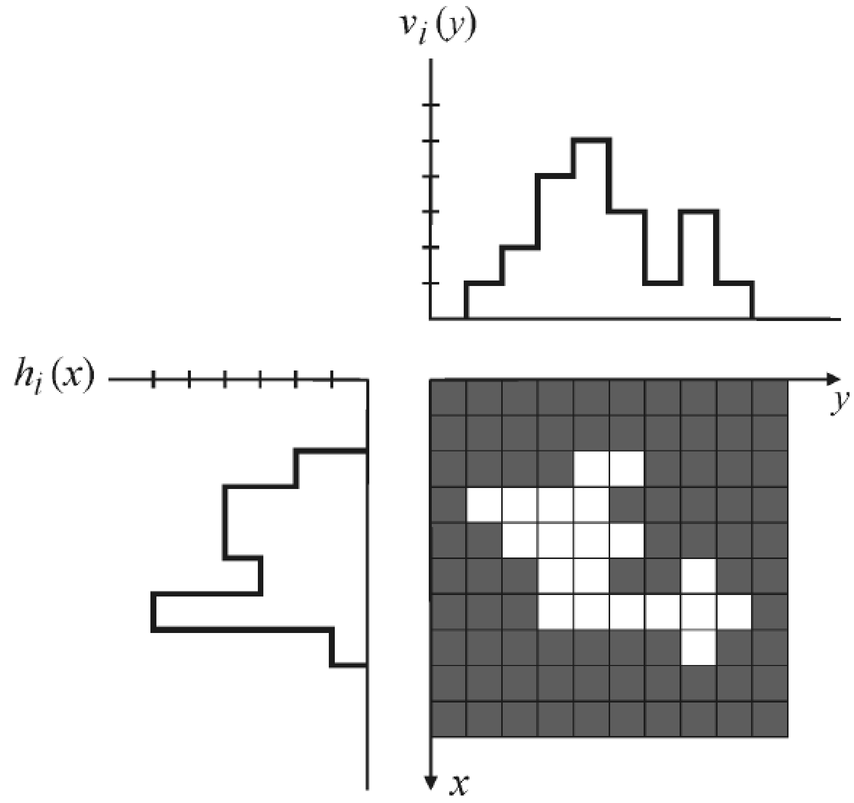
$$\tan(2\theta_i) = 2 \times \frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} x O_i(x,y)}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} x^2 O_i(x,y) - \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} y^2 O_i(x,y)}$$

Binary object features

- Projections:

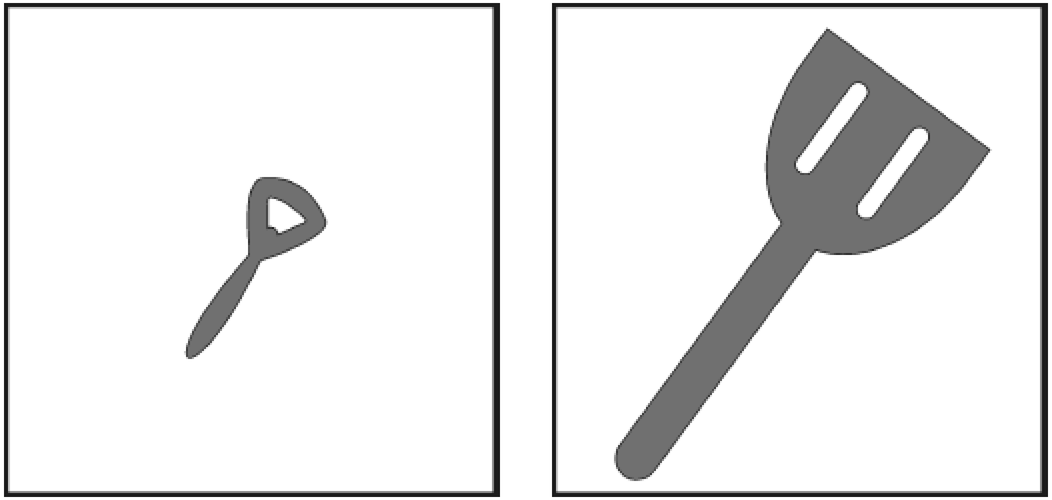
$$h_i(x) = \sum_{x=0}^{M-1} O_i(x,y)$$

$$v_i(y) = \sum_{y=0}^{N-1} O_i(x,y)$$



Binary object features

- Euler number:
  - E = C - H



Binary object features

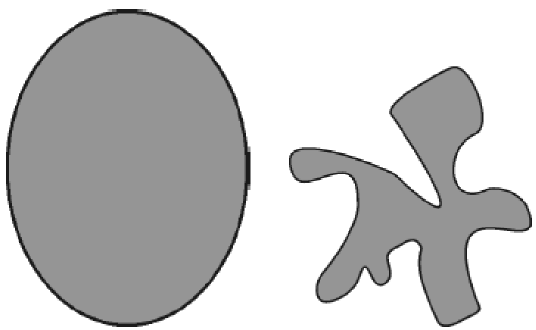
- Perimeter:
  - Can be calculated by counting the number of object pixels (whose value is 1) that have one or more background pixels (whose value is 0) as their neighbors.
  - Alternative method: extract the edge (contour) of the object and then count the number of pixels in the resulting border.
  - In MATLAB: **`bwperim`**

Binary object features

- Thinness ratio:

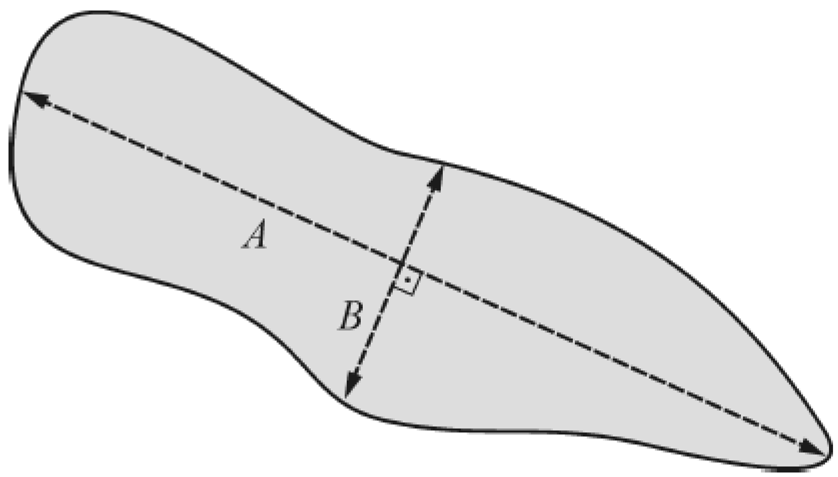
$$T_i = \frac{4\pi A_i}{{P_i}^2}$$

- Often used as a measure of roundness.
  - The higher the thinness ratio (as it approaches 1), the more round the object.
  - Its inverse is sometimes called *irregularity* or *compactness* ratio.



Binary object features

- Eccentricity:
  - The ratio of the major and minor axes of the object (A/B) .

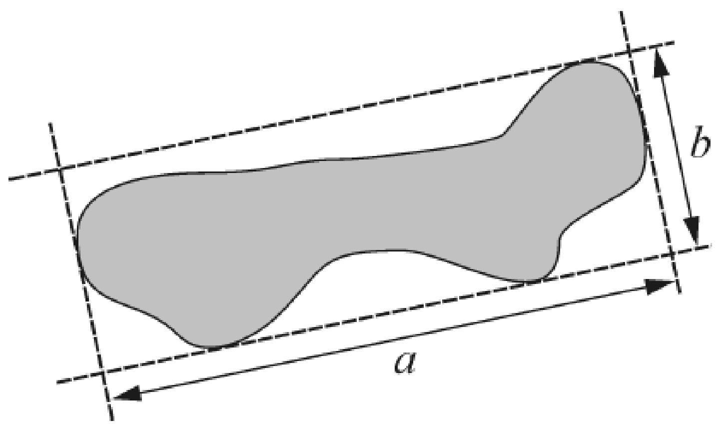


Binary object features

- Aspect ratio:
  - The ratio of the major and minor axes of the object .

$$AR = \frac{x_{max} - x_{min} + 1}{y_{max} - y_{min} + 1}$$

- Elongatedness (a/b):



Binary object features

- Moments:
  - The 2D *moment of order (p + q)* of a digital image *f(x, y)* is defined as:

$$m_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} x^p y^q f(x, y)$$

Binary object features

- Central moments:
  - Translation-invariant equivalent of moments.

$$\mu_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (x - \bar{x})^p (y - \bar{y})^q f(x, y)$$

$$\bar{x} = \frac{m_{10}}{m_{00}} \quad \text{and} \quad \bar{y} = \frac{m_{01}}{m_{00}}$$

Binary object features

- Normalized central moments:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}}$$

$$\text{for } (p+q) > 1$$

$$\gamma = \frac{p+q}{2} + 1$$

Binary object features

- RST-invariant moments [Hu, 1962]:

$$
\begin{aligned}
\phi_1 = &\quad \eta_{20} + \eta_{02} \\
\phi_2 = &\quad (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
\phi_3 = &\quad (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
\phi_4 = &\quad (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
\phi_5 = &\quad (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})\left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] \\
&\quad + (3\,\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] \\
\phi_6 = &\quad (\eta_{30} - \eta_{02})\left[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
\phi_7 = &\quad (3\,\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})\left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] - \\
&\quad (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right]
\end{aligned}
$$

Binary object features

- MATLAB **`regionprops`** function
  - Measures a set of properties for each labeled region *L*.
  - One or more properties from the list in Table 18.2 (pp. 457-458) can be specified as parameters.

Boundary descriptors

- *Contour*-based (as opposed to *region*-based)
- Assumptions:
  - The contour (or boundary) of an object can be represented in a convenient coordinate system (Cartesian, polar, or tangential) and rely exclusively on boundary pixels to describe the region or object.
  - The pixels belonging to the boundary of the object (or region) can be traced, starting from any background pixel, using an algorithm known as *bug tracing*.

Boundary descriptors

- The *bug tracing* algorithm:
  - repeat
    - as soon as the conceptual bug crosses into a boundary pixel, it makes a left turn and moves to the next pixel;
      - if that pixel is a boundary pixel, the bug makes another left turn;
      - otherwise, it turns right;
  - until the bug is back to the starting point.
- As the conceptual bug follows the contour, it builds a list of the coordinates of the boundary pixels being visited.

Boundary descriptors

- Example 18.2

  The *bug tracing* algorithm in MATLAB (**`bwtraceboundary`**):

Boundary descriptors

- Example 18.3

  Boundary tracing in MATLAB (**`bwboundaries`**):

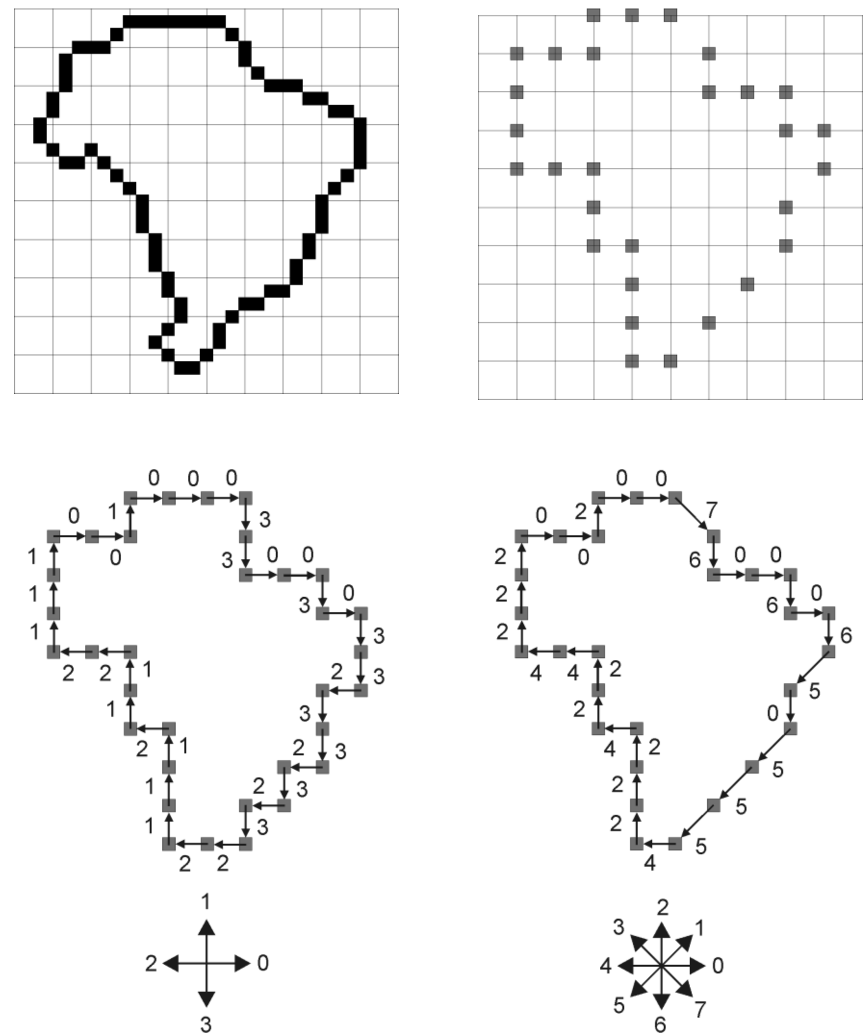
Chain code, Freeman code, and shape number

- A chain code is a boundary representation technique by which a contour is represented as a sequence of straight-line segments of specified length (usually 1) and direction.
- The simplest chain code mechanism (*crack code*) consists of assigning a number to the direction followed by a bug tracking algorithm as follows: right (0), down (1), left (2), and up (3).
  - Assuming that the total number of boundary points is p (the perimeter of the contour), the array C, where $C(p) = 0, 1, 2, 3$, contains the chain code of the boundary.

Chain code, Freeman code, and shape number

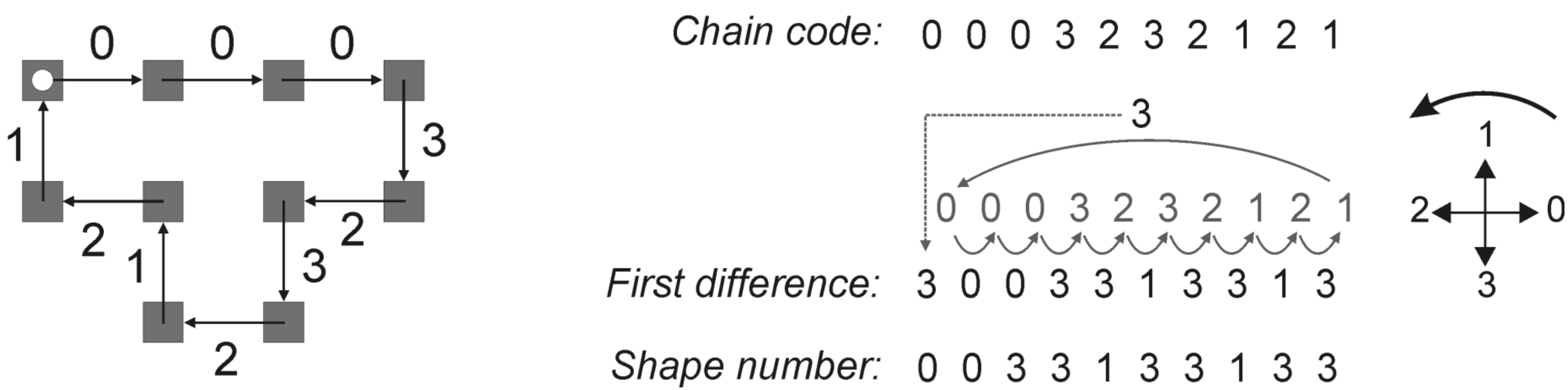
- Freeman code: a modified version of the basic chain code, using eight directions instead of four.

- Example:



Chain code, Freeman code, and shape number

- Shape number: normalized, rotation-invariant, equivalent of the chain code.

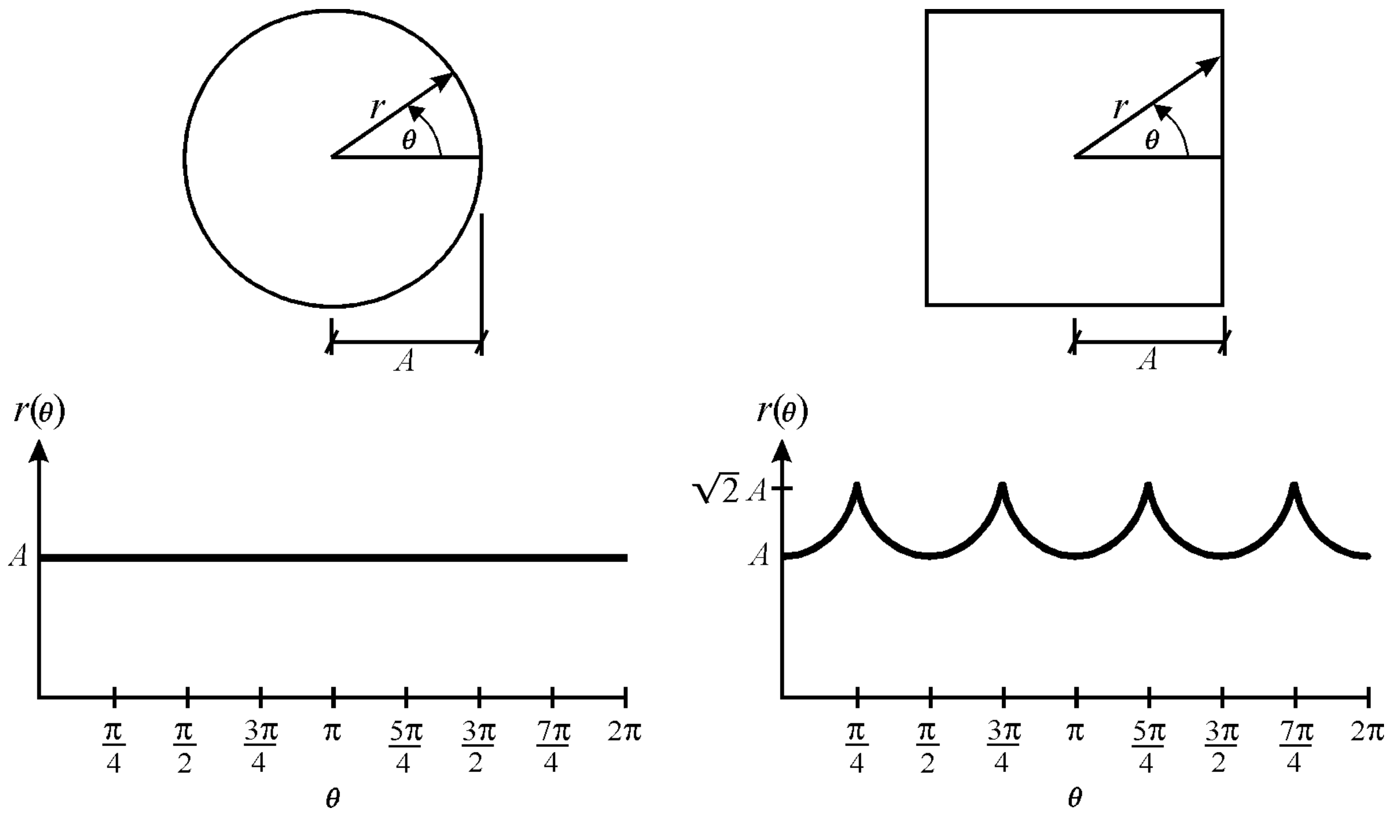


Signatures

- 1D representation of a boundary, usually obtained by representing the boundary in a polar coordinate system and computing the distance *r* between each pixel along the boundary and the centroid of the region, and the angle θ subtended between a straight line connecting the boundary pixel to the centroid and a horizontal reference.
- The resulting plot provides a concise representation of the boundary, that is RT-invariant, but ***not*** scaling-invariant.

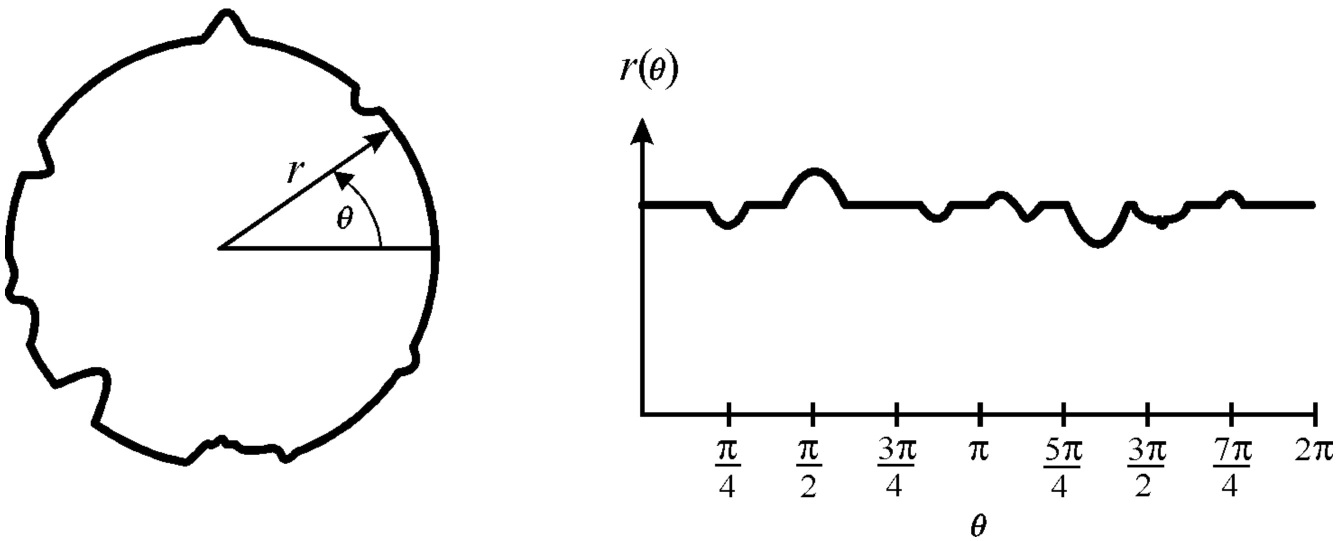
Signatures

- Examples:



Signatures

- Effects of noise:

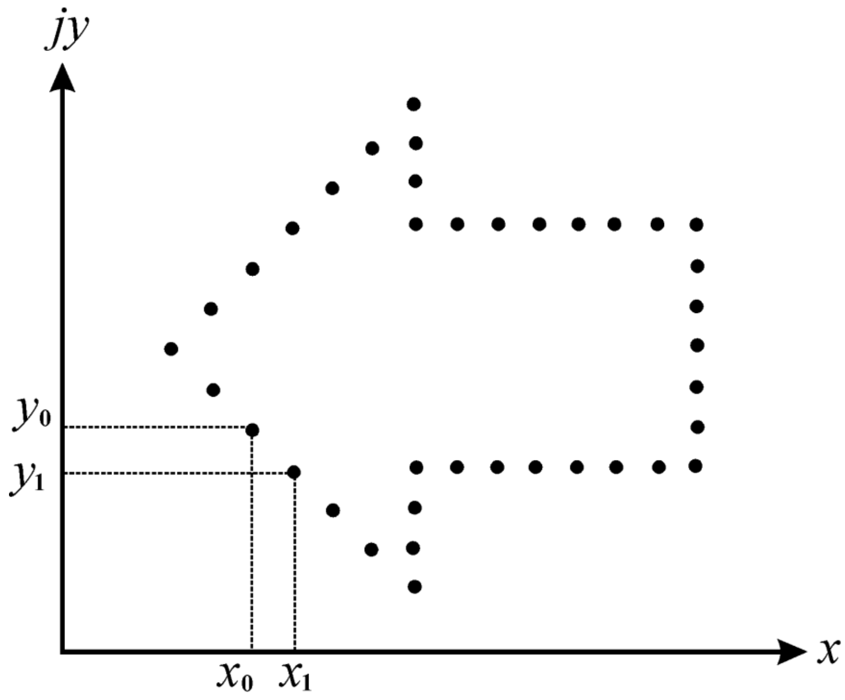


Fourier descriptors

- Basic idea: to traverse the pixels belonging to a boundary, starting from an arbitrary point, and record their coordinates.
- Each value in the resulting list of coordinate pairs (x_0,y_0), (x_1,y_1), …, (x_{K-1}, y_{K-1}) is then interpreted as a complex number $x_k + j\, y_k$, for $k = 0, 1, \ldots, K-1$.
- The Discrete Fourier Transform (DFT) of this list of complex numbers is the *Fourier descriptor* of the boundary.
- The inverse DFT restores the original boundary.

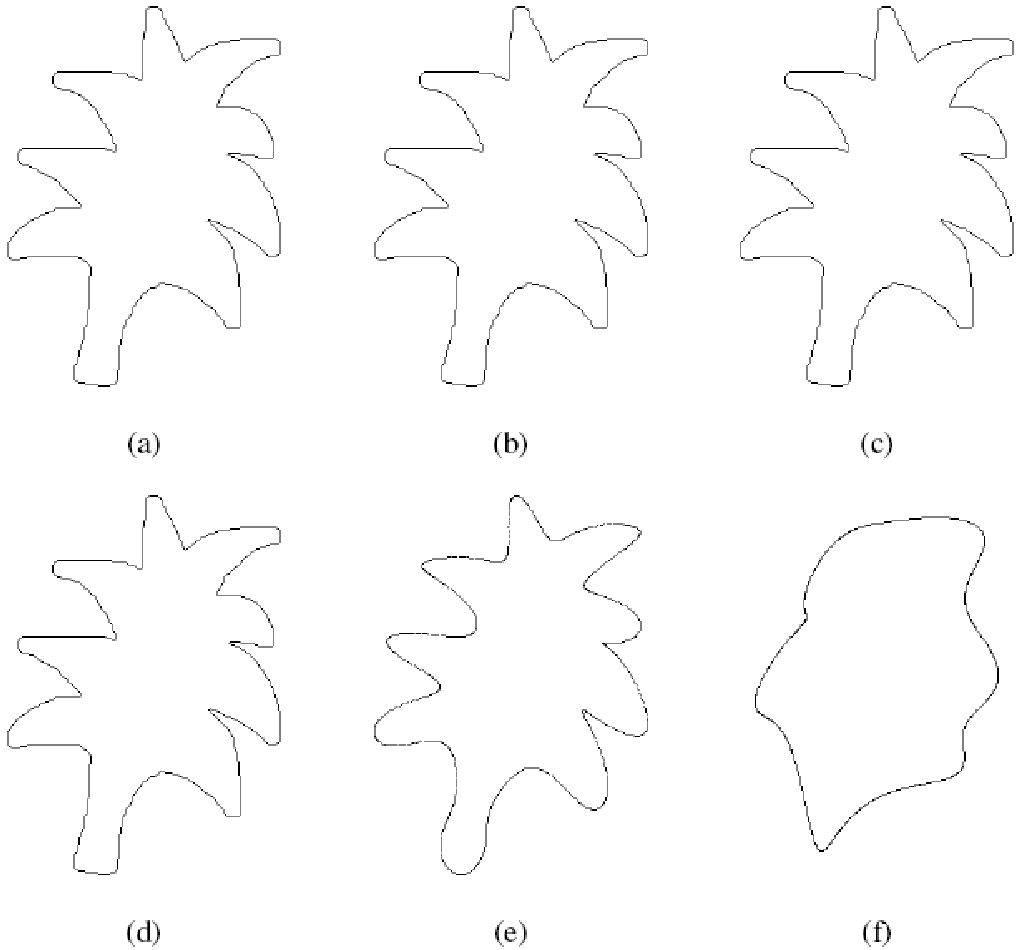
Fourier descriptors

- Basic idea



Fourier descriptors

- Interesting property: contour can be encoded using very few values (coefficients).
- Example 18.4:



Histogram-based features

- Average gray value:

$$m = \sum_{j=0}^{L-1} r_j p(r_j)$$

$$m = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y)$$

Histogram-based features

- Standard deviation:

$$\sigma = \sqrt{\sum_{j=0}^{L-1} (r_j - m)^2 p(r_j)}$$

- Skew:

$$\text{skew} = \frac{1}{\sigma^3} \sum_{j=0}^{L-1} (r_j - m)^3 p(r_j)$$

Histogram-based features

- Energy:

$$\text{energy} = \sum_{j=0}^{L-1} [p(r_j)]^2$$

- Entropy:

$$\text{entropy} = -\sum_{j=0}^{L-1} p(r_j) \log_2[p(r_j)]$$

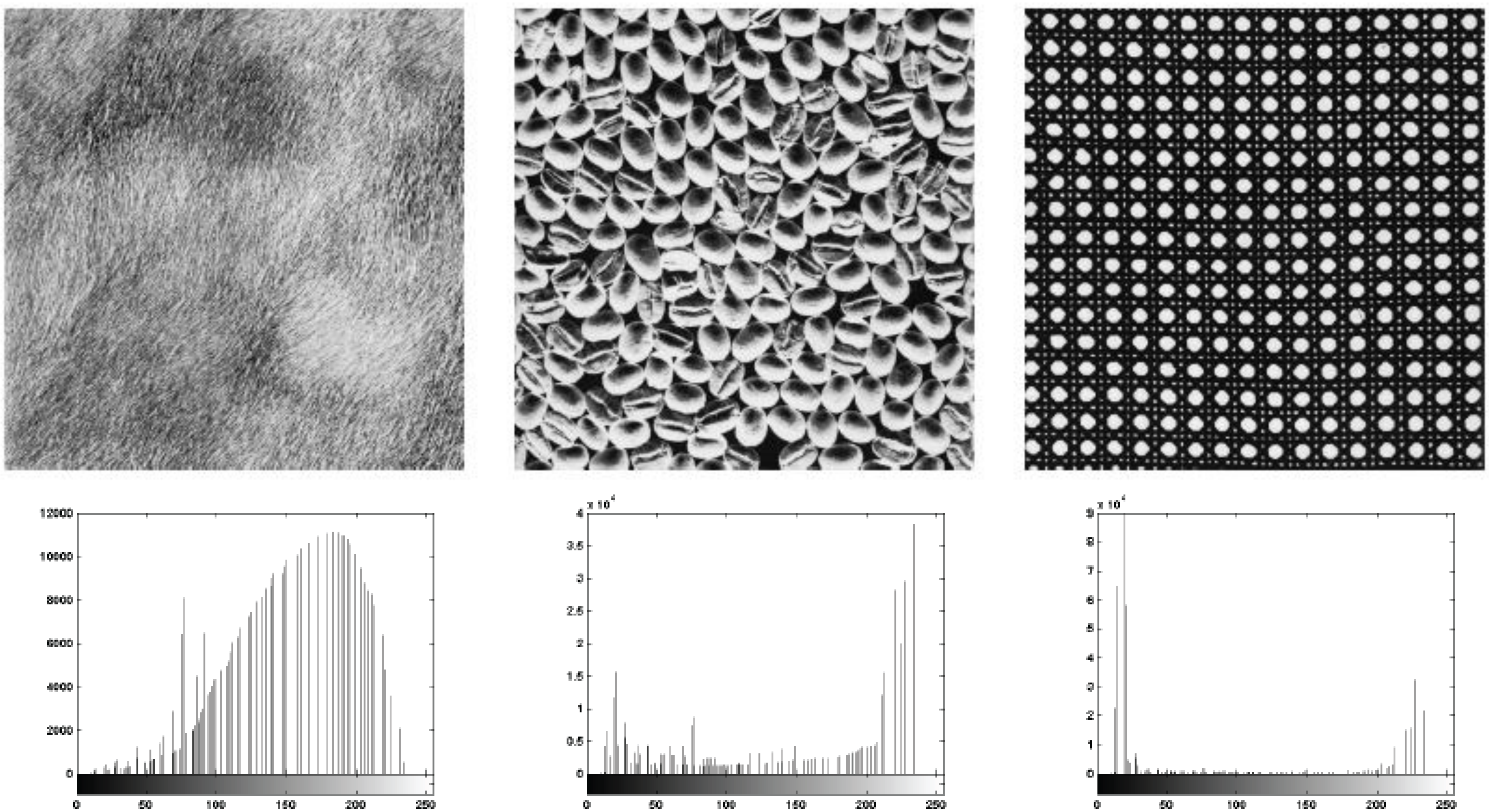
Texture features

- Texture can be a powerful descriptor of an image (or one of its regions).

- Although there is not a universally agreed upon definition of texture, image processing techniques usually associate the notion of texture with image (or region) properties such as *smoothness* (or its opposite, *roughness*), *coarseness*, and *regularity*.

Texture features

- Texture – examples:

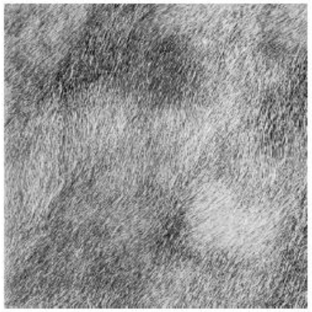
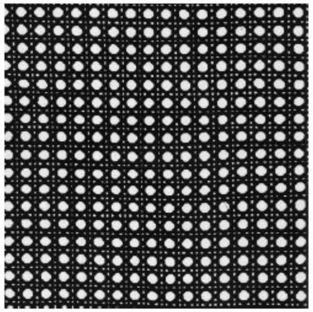


By Oge Marques

Texture features

- Texture – Example 18.5:

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

Texture	Mean	Standard deviation	Roughness R	Skew	Uniformity	Entropy
Smooth	147.1459	47.9172	0.0341	-0.4999	0.0190	5.9223
Coarse	138.8249	81.1479	0.0920	-1.9095	0.0306	5.8405
Regular	79.9275	89.7844	0.1103	10.0278	0.1100	4.1181

Texture features

- Gray-level co-occurrence matrix (GLCM)
  - Displacement vector: $\mathbf{d} = (d_x, d_y)$
  - Example [d=(0,1)]:

0	1	5	5	2	0
3	6	3	0	7	6
7	7	5	7	0	1
3	2	6	3	1	7
6	3	6	3	5	1
4	7	5	3	5	4

i \ j	0	1	2	3	4	5	6	7
0	0	2	0	0	0	0	0	1
1	0	0	0	0	0	1	0	1
2	1	0	0	0	0	0	1	0
3	1	1	1	0	0	2	2	0
4	0	0	0	0	0	0	0	1
5	0	1	1	1	1	2	0	0
6	0	0	0	4	0	0	0	0
7	1	0	0	0	0	2	1	1

Texture features

- Gray-level co-occurrence matrix (GLCM)
  - Another example [d=(1,0)]:

0	1	5	5	2	0
3	6	3	0	7	6
7	7	5	7	0	1
3	2	6	3	1	7
6	3	6	3	5	1
4	7	5	3	5	4

i \ j	0	1	2	3	4	5	6	7
0	0	0	0	0	0	1	0	1
1	1	0	0	0	0	0	1	1
2	0	0	0	0	0	0	0	1
3	1	0	1	2	0	1	0	2
4	0	1	0	0	0	0	1	0
5	0	1	0	1	0	1	1	0
6	1	1	0	1	0	1	1	0
7	1	1	1	2	0	0	1	0

Texture features

- Normalized GLCM and associated features:

$$N_g(i,j) = \frac{g(i,j)}{\sum_i \sum_j g(i,j)}$$

$$\text{Maximum probability} = \max_{i,j} N_g(i,j)$$

$$\text{Energy} = \sum_i \sum_j {N_g}^2(i,j)$$

$$\text{Entropy} = -\sum_i \sum_j N_g(i,j) \log_2 N_g(i,j)$$

$$\text{Contrast} = \sum_i \sum_j (i-j)^2 N_g(i,j)$$

$$\text{Homogeneity} = \sum_i \sum_j \frac{N_g(i,j)}{1+|i-j|}$$

$$\text{Correlation} = \frac{\sum_i \sum_j (i-\mu_i)(j-\mu_j) N_g(i,j)}{\sigma_i \sigma_j}$$

Texture features

- Texture descriptors based on GLCM
  Example 18.6:

Texture	Max Probability	Correlation	Contrast	Uniformity (Energy)	Homogeneity	Entropy
Smooth	0.0013	0.5859	33.4779	0.0005	0.0982	7.9731
Coarse	0.0645	0.9420	14.5181	0.0088	0.3279	6.8345
Regular	0.1136	0.9267	13.1013	0.0380	0.5226	4.7150

Hands-on

- Tutorial 18.1: Feature extraction and representation (page 470)

