Digital Image Processing

Image description

Erchan Aptoula

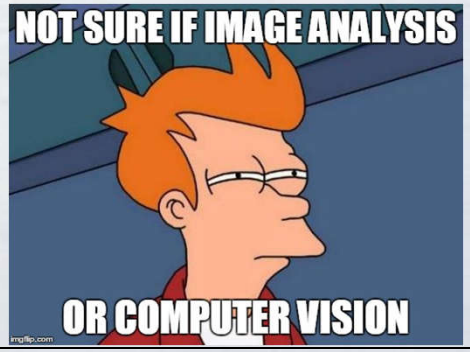
Spring 2016-2017

Image description

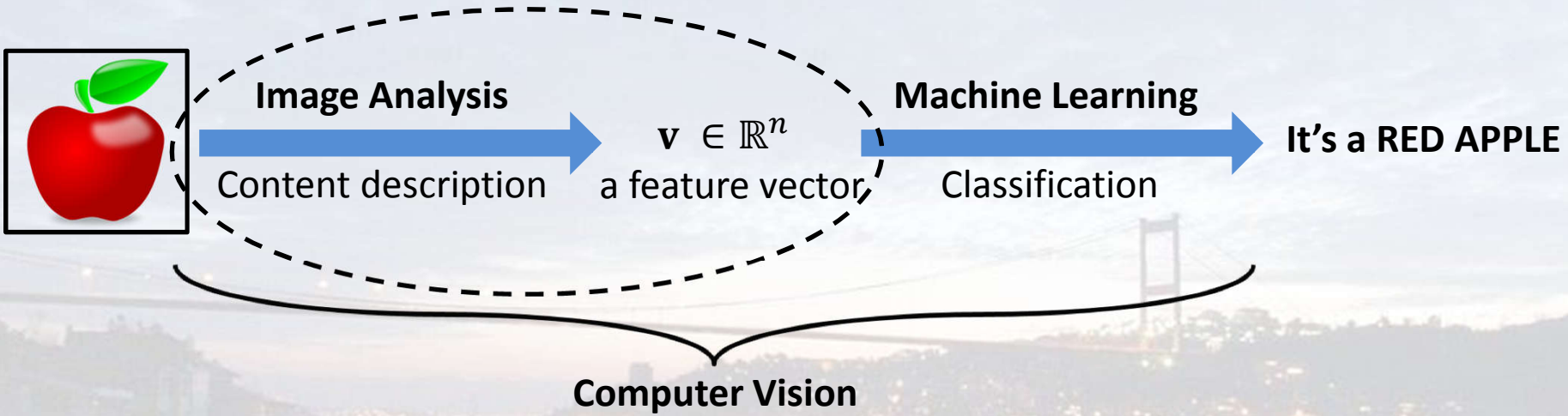
Contents

- Principles of image description
- Color
- Shape
- Texture
- Content-based image retrieval

Our topic is **description**: a key stage in the process of visual **recognition**.

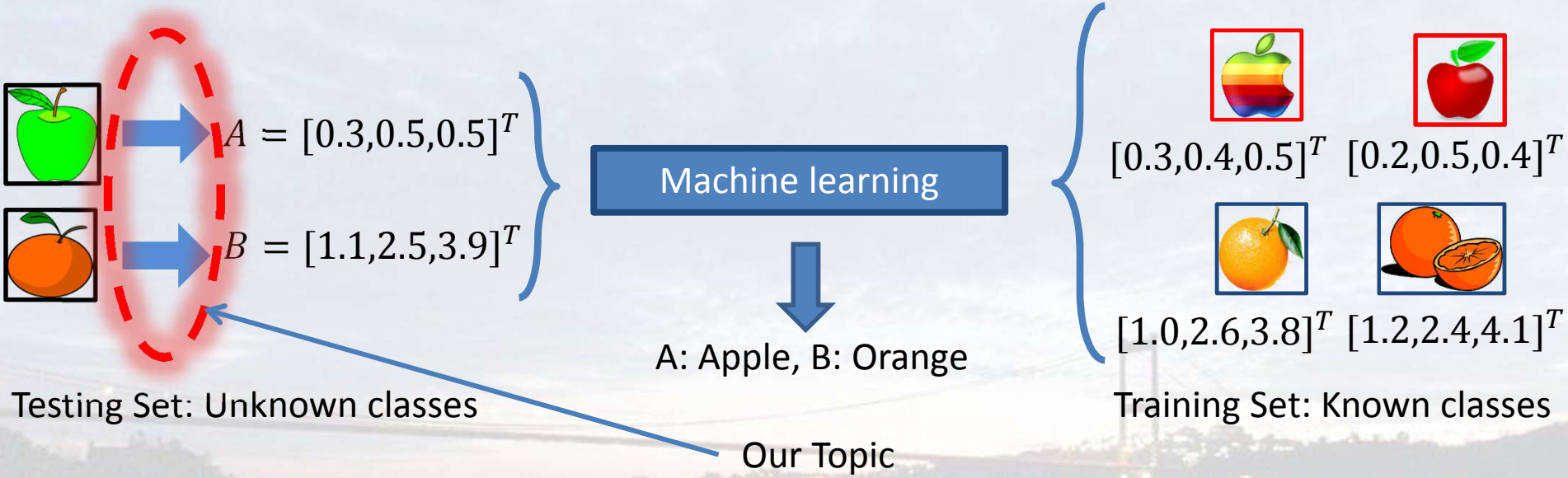


The goal of vision is to understand visual data; in other words to associate it with symbols.



Content description is about "abstractifying" a concept; in other words, given an image f of X, we reduce f to the essence that makes it an image of X.

This way we can compare apples and oranges, and tell them apart.



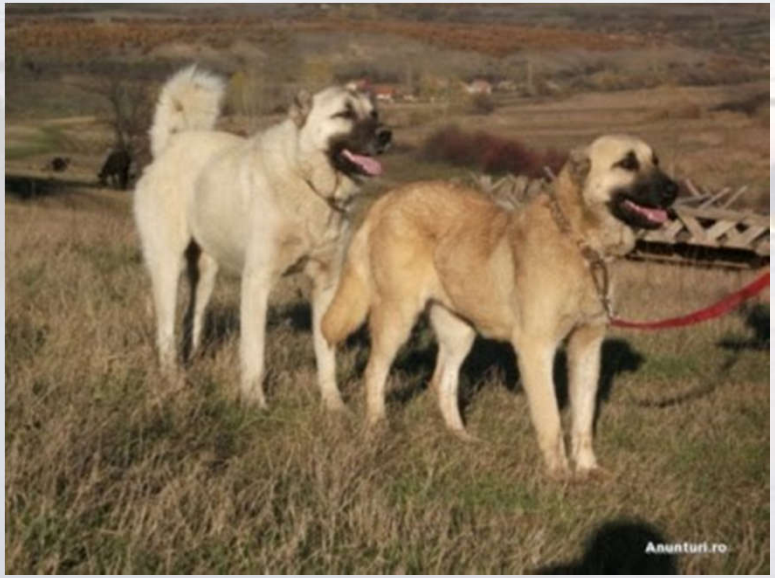
The tool that performs content description is called a **descriptor**.

The description that is produced by a descriptor is casually called a **feature vector**.

Nobody knows yet how exactly humans describe/recognize things; and descriptors try to simulate/approximate this mysterious ability.

What's a good descriptor ?

- A good descriptor has **high discriminatory** power
  - It produces similar feature vectors for inputs representing similar notions.
  - And dissimilar feature vectors for inputs representing distinct notions.



Dogs

Wolves

Image description

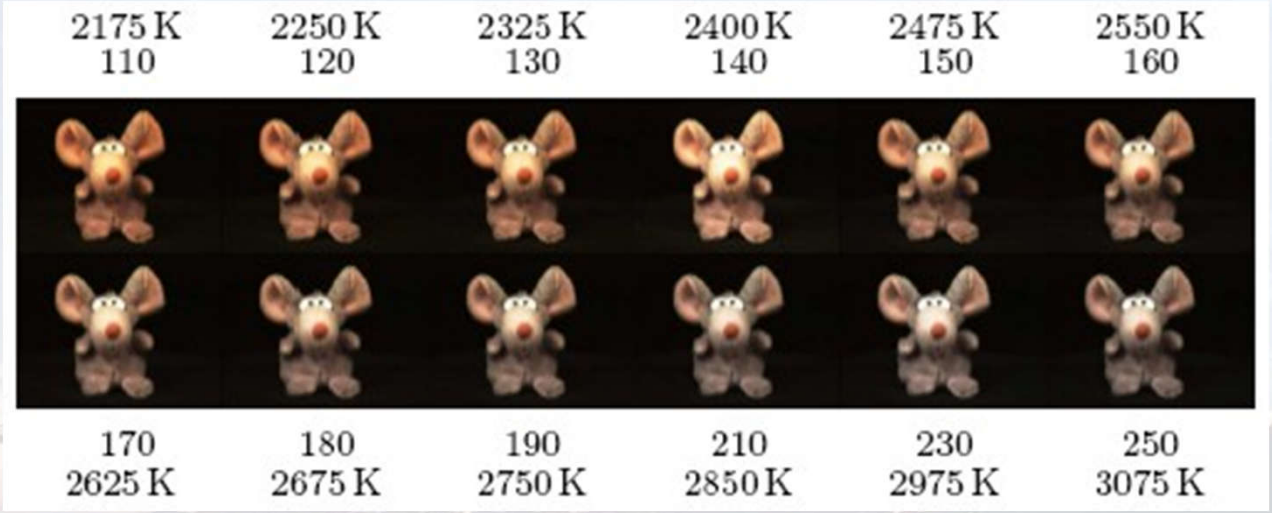
Not easy!

Problem: high intra-class variance

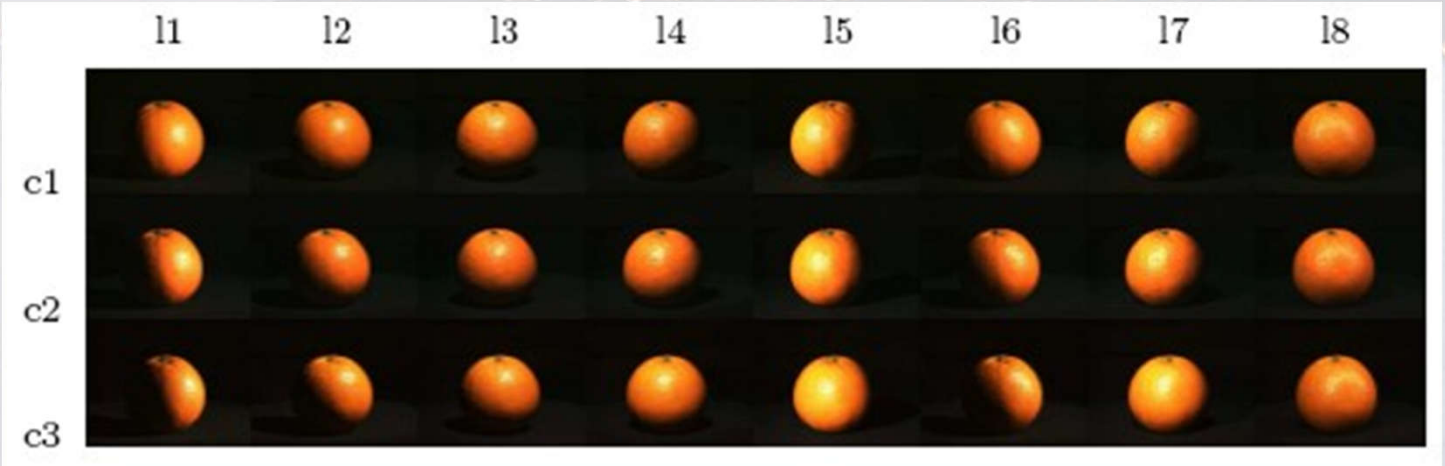
Problem: low inter-class variance.

- A good descriptor is **invariant** against disruptive factors

Such as illumination color (or temperature)

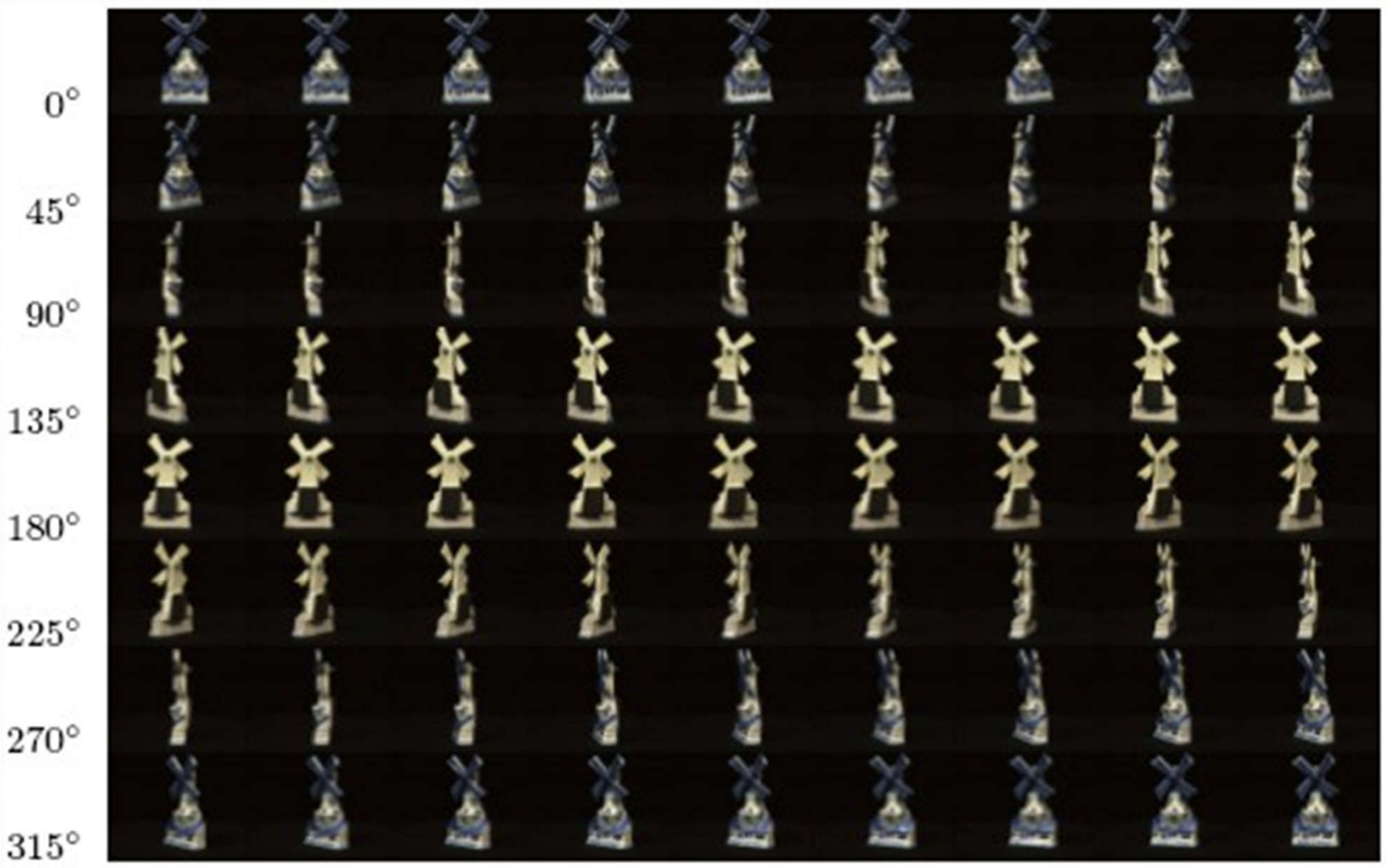


Illumination direction

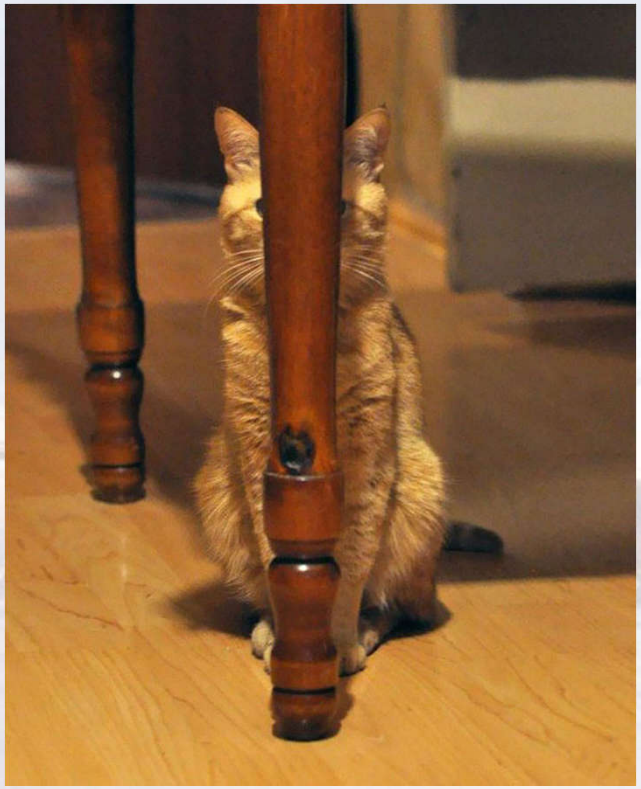


Images from ALOI

Viewpoint/Rotation



Occlusion



Deformations

Scale (distance) variations

©Warren Photographic

Plus, even if your descriptor is robust/invariant against all these, it also has to be **easy/fast to compute** and **memory efficient**.

Image description: short history

1990s: can we describe objects under controlled conditions?

First CBIR systems, global and regional descriptors, etc.

"Looks like a bird; but the angle isn't right, might also be a truck"

2000s: can we describe objects under *moderately* varied conditions?

Local descriptors, scale invariance, SIFT, bag-of-words, visual vocabularies.

"It's a bird, no question about it; I'd recognize it from farther too"

2010s: can we describe objects under varied conditions, simple actions, image context?

"It's a dove flying in the air with an olive branch"

Now: can we describe abstract concepts/ compound actions/context in uncontrolled conditions?

Deep learning, CNN, RNN, LSTM, etc.

"It's an image of peace"

We'll focus on **low-level** descriptors; i.e. "**old**" and relatively **simple** tools that aim to describe basic visual properties of their input.

Most of these tools may receive as input either the entire image, in which case they are called **global descriptors** or just a region of an image, in which case they are named **regional descriptors** (local and deep description is out of our scope).

The basic visual properties used for distinguishing objects are (for now):

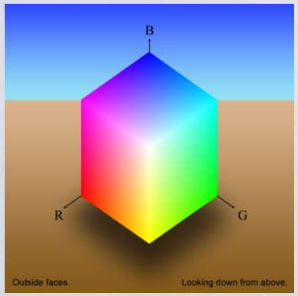
- color
- shape
- and texture

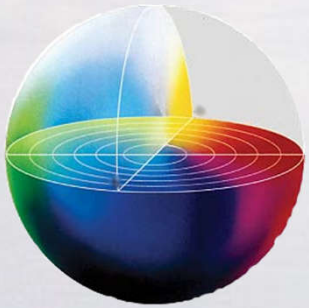
Color is an attractive attribute; it is robust against scale, distance, deformations and viewpoint; however it is also very sensitive to illumination conditions.

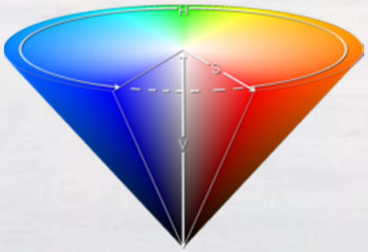
An important question: which color space should one use for description?

Popular choices:

- **Normalized RGB**: fast but not aligned with human color perception
- Phenomenal color spaces (**HSV, HLS**, etc): intuitive but hue is an angle…
- CIE variants (**CIELAB, CIELUV**, etc): perceptually uniform, suitable for color operations, device independent (if you know the white reference for each image), but expensive to convert.







16 million colors certainly look nice, but

a) they are not all necessary for recognition

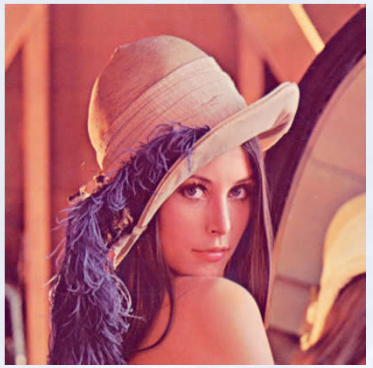
b) they lead to slow and memory intensive descriptors.

So, how many bits per channel should we use?

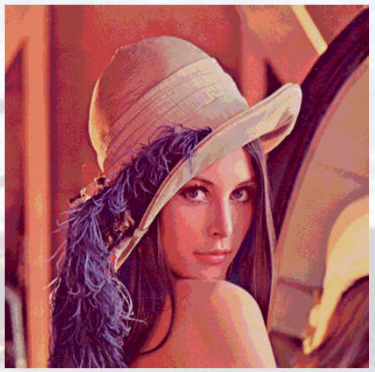
if they are too few, we lose too much information!

Red Green and Blue have “equivalent” significance, so they can be quantized to equal bits; e.g. 4 4 4 (64 colors) or even less.

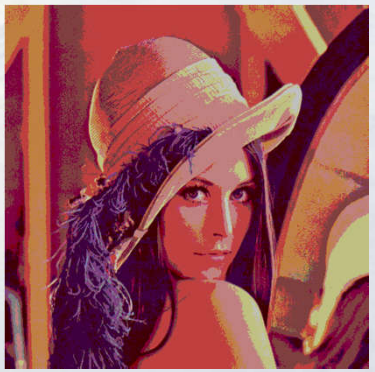
But HSV? Hue will need as many bits as there are dominant colors, S and V could work with less; i.e. unequal quantization.



16M



512



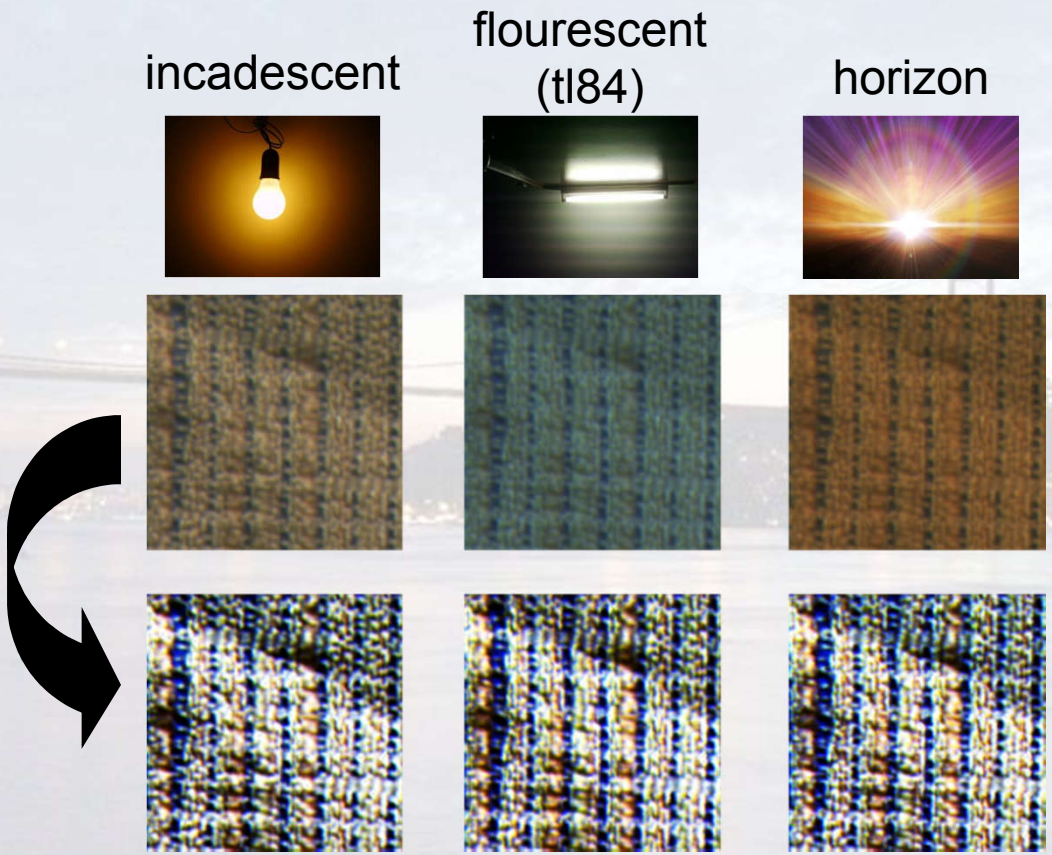
64

The type of quantization also matters. It could be uniform, or image specific.

16 colors

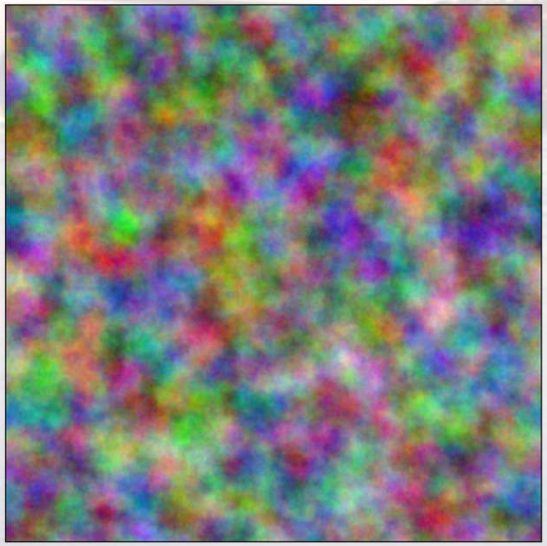
Easy to accomplish with a simple clustering of the 3D color space. Cluster centers will be your palette colors.

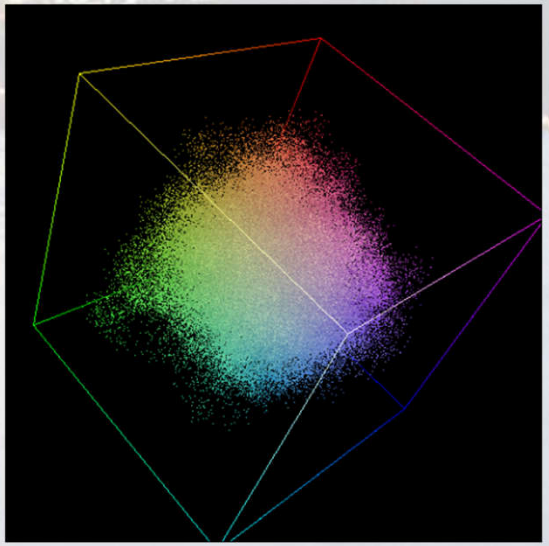
Note: one way of neutralizing the effect of illumination temperature is applying our old friend **histogram equalization**. But be careful with it, as it alters the appearance.



What should a color descriptor provide?

- The presence (or absence) of a particular color (e.g. red and blue)
- The relative amount in which a particular color is present (e.g. 80% red and 20% blue)
- The size distribution of each color (e.g. one large patch of red and multiple small patches of blue)
- The relative spatial position of each color with respect to others (e.g. red at left and blue at right)

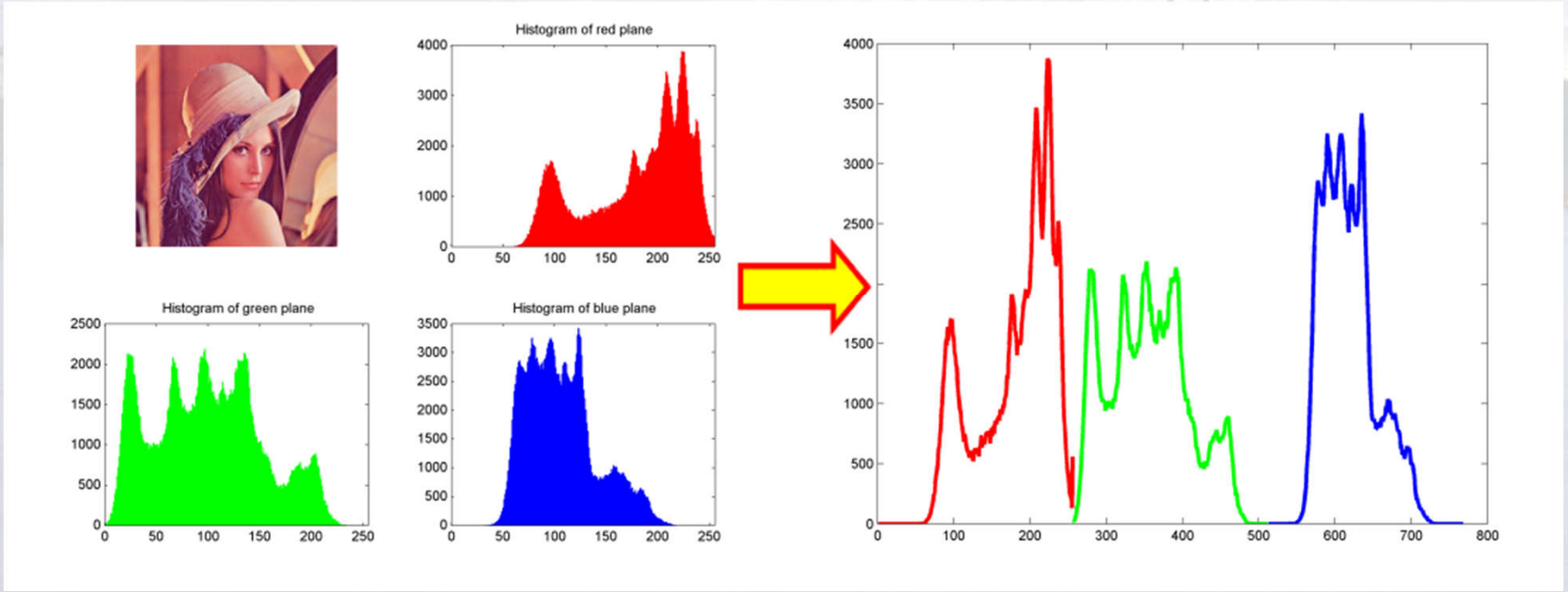




Color histogram: the most traditional (color) descriptor. It provides the occurrence frequency of colors in an image, usually after some form of quantization.

Robust against translation and rotation about the view axis, **fast** to calculate.

Main drawback: the complete **lack of spatial information,** has **low discriminatory power**; can be reinforced with various methods; e.g. image partitioning, etc.



Color auto-correlogram (1997)

A solid attempt to add spatial sensitivity to the color histogram.

Given a set of distances $\{k_i\}$; e.g. $\{1,3,5,7\}$ and an image f with n possible colors, it calculates for every distance k_i and every color c_j, the probability of encountering two pixels of color c_j separated by a distance k_i.

If f_c is the subset of pixels of f with color c, then:

$$\alpha_c{}^{k_i}(f) = P(p_2 \in f_c \mid |p_1 - p_2| = k_i, p_1 \in f_c)$$

Its length is n × $|\{k_i\}|$, so color quantization is mandatory.

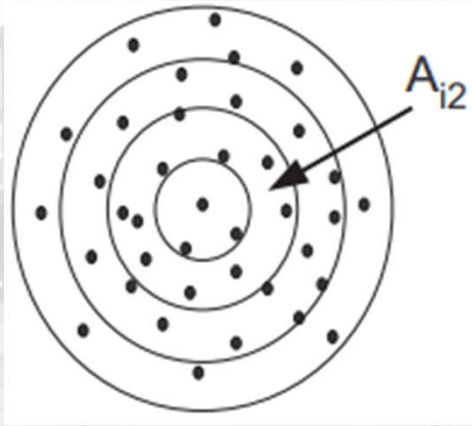
Color distribution entropy (CDE) (2006)

A more “recent” attempt at spatially sensitive color description.

Let A_i be the set of pixels of color $1 \leq i \leq n$ of a given image.

Let c_i and r_i be respectively their centroid and max distance to the centroid.

Draw N concentric circles with center c_i and radius $\frac{jr_i}{N}, 1 \leq j \leq N$; $|A_{ij}|$ now denotes the number of pixels in circle j of color i.



The **annular color histogram** is $P_i = (P_{i1}, \dots, P_{iN})$

where $P_{ij} = |A_{ij}|/|A_i|$

And the CDE becomes $(h_1, E(P_1), \dots, h_n, E(P_n))$

where h_i is the histogram of bin i and E is distribution entropy.

$$E(P_i) = -\sum_{j=1}^{N} P_{ij} \log(P_{ij})$$

There are many more color descriptors:

- Haar wavelet transform,
- Dominant color transform,
- Color layout transform,
- Color structure descriptor,
- Color moments,
- Color histogram sets,
- Border/Interior pixel classification, etc.

Remember: color descriptors are often robust to transformations, fast and not bad at providing an “overall” description of a scene.

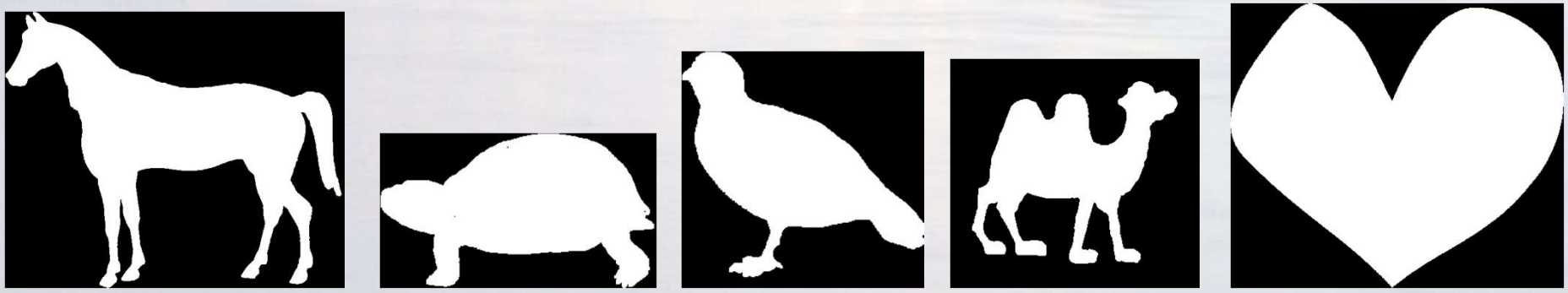
But they are sensitive to illumination conditions and scale, and mostly lack spatial information. All the same, they can help by providing valuable **complementary information** to more powerful descriptors, especially under controlled illumination and/or with real-time systems.

Our next stop is another significant object attribute: **shape**!

2D Shape descriptors can be separated into **2 categories**:

- **Boundary descriptors**: aiming to describe shape based only on their boundary pixels,
- and **regional descriptors**, that try to accomplish the same, but by using all the pixels of the shape under consideration.

Nice shapes are not easy to come by, as they require you to first **separate the object of interest from the background** (through segmentation, more on this later).



Simple descriptors

- Area (A)
- Perimeter (P) (i.e. number of border pixels)
- Compactness or circularity: how compact a shape is: P^2/A. A circle has a circularity of 4π all other shapes have a circularity greater than that.
- Rectangularity: how rectangular a shape is; i.e. how much it fills its minimal bounding box (area of object/area of bounding box). It'll be 1 for a perfect rectangle.

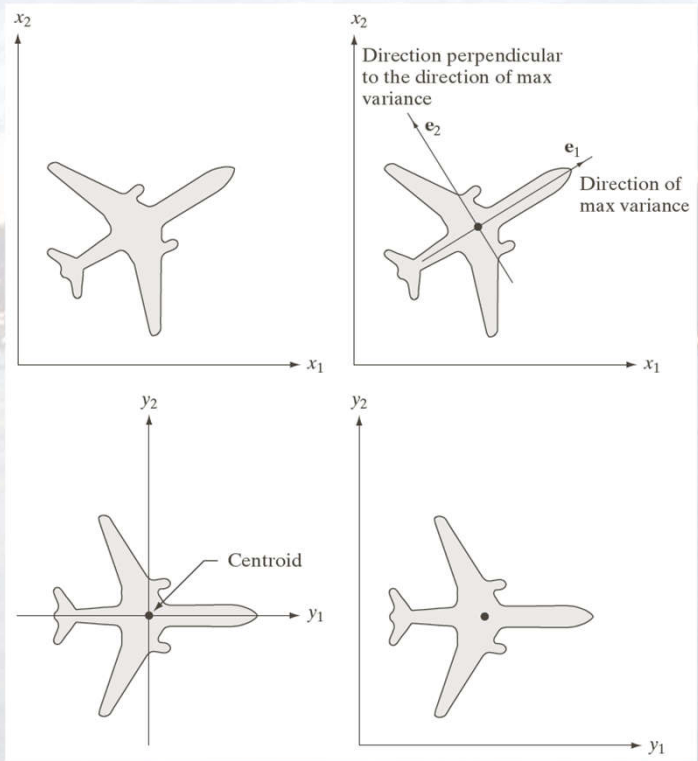
Simple descriptors

- Eccentricity: the ratio of principal axes of inertia. The length of the major axis is the larger eigenvalue of the covariance matrix of the boundary points, and the length of the minor axis is the smaller one.

$$det(C - \lambda_{1,2} I) = det \begin{pmatrix} c_{xx} - \lambda_{1,2} & c_{xy} \\ c_{yx} & c_{yy} - \lambda_{1,2} \end{pmatrix}$$

$$= (c_{xx} - \lambda_{1,2})(c_{yy} - \lambda_{1,2}) - c_{xy}^2 = 0$$

$$\begin{cases} \lambda_1 = \frac{1}{2}\left[c_{xx} + c_{yy} + \sqrt{(c_{xx} + c_{yy})^2 - 4\left(c_{xx}c_{yy} - c_{xy}^2\right)}\right] \\ \lambda_2 = \frac{1}{2}\left[c_{xx} + c_{yy} - \sqrt{(c_{xx} + c_{yy})^2 - 4\left(c_{xx}c_{yy} - c_{xy}^2\right)}\right] \end{cases}$$



a b
c d

FIGURE 11.43
(a) An object. (b) Object showing eigenvectors of its covariance matrix. (c) Transformed object, obtained using Eq. (11.4-6). (d) Object translated so that all its coordinate values are greater than 0.

$$\mathbf{C_x} = \begin{bmatrix} 3.333 & 2.00 \\ 2.00 & 3.333 \end{bmatrix} \quad \mathbf{e}_1 = \begin{bmatrix} 0.707 \\ 0.707 \end{bmatrix}$$

$$\mathbf{m_x} = \begin{bmatrix} 3 \\ 3 \end{bmatrix} \quad \mathbf{e}_2 = \begin{bmatrix} -0.707 \\ 0.707 \end{bmatrix}$$

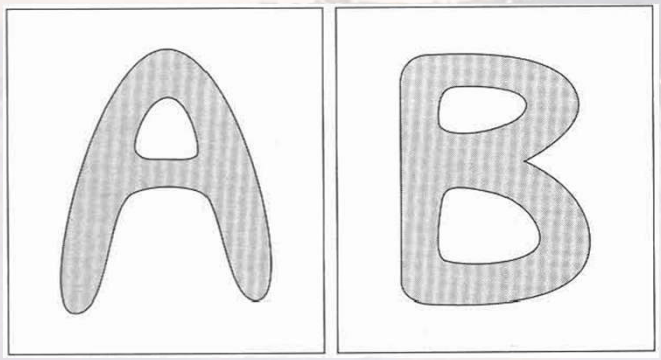
Euler's characteristic (topological descriptor)

$$E = C - H$$

C: the number of connected components

H: the number of holes

Very useful for separating simple shapes.

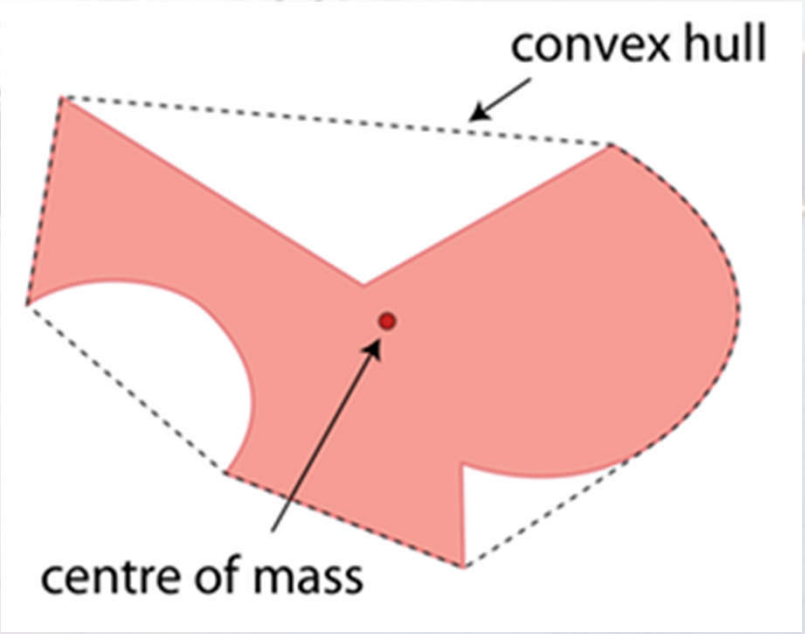


Convexity (Regional descriptor)

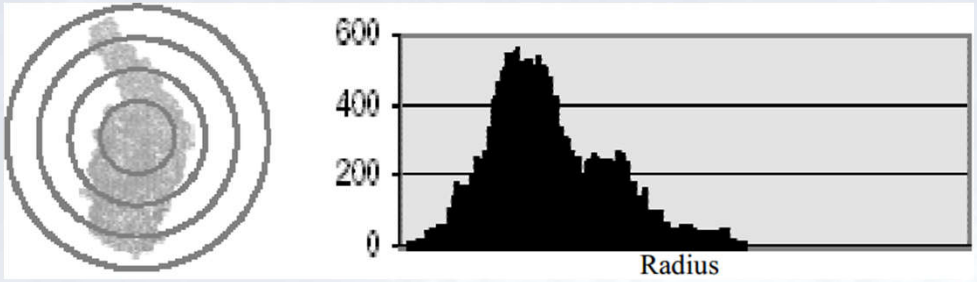
The convex hull of a shape S, is the smallest convex shape that contains it.

A shape is convex if the segment that connects any two points in it, lies completely inside it.

Convexity is defined as the area of the shape, divided by the area of its convex hull.



Shape histograms (1999) (Regional descriptor)

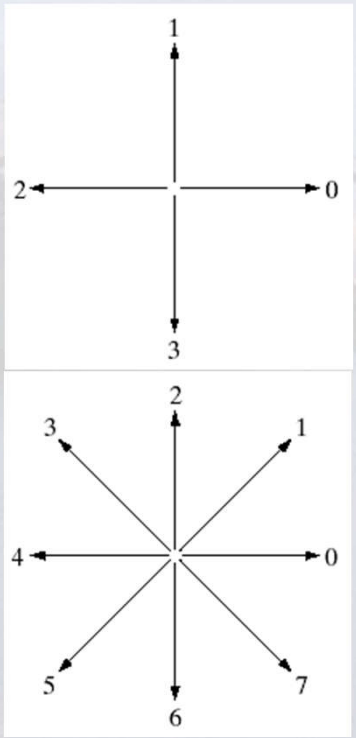


After calculating the center of mass of the image, you draw regularly spaced n concentric circles around it, up until the farthest point from the center. The descriptor is constructed by measuring the amount of surface in each circular band.

It is evidently **rotation invariant**, fast to calculate and if you additionally normalize every bin by the total surface, it becomes reasonably **scale invariant** as well.

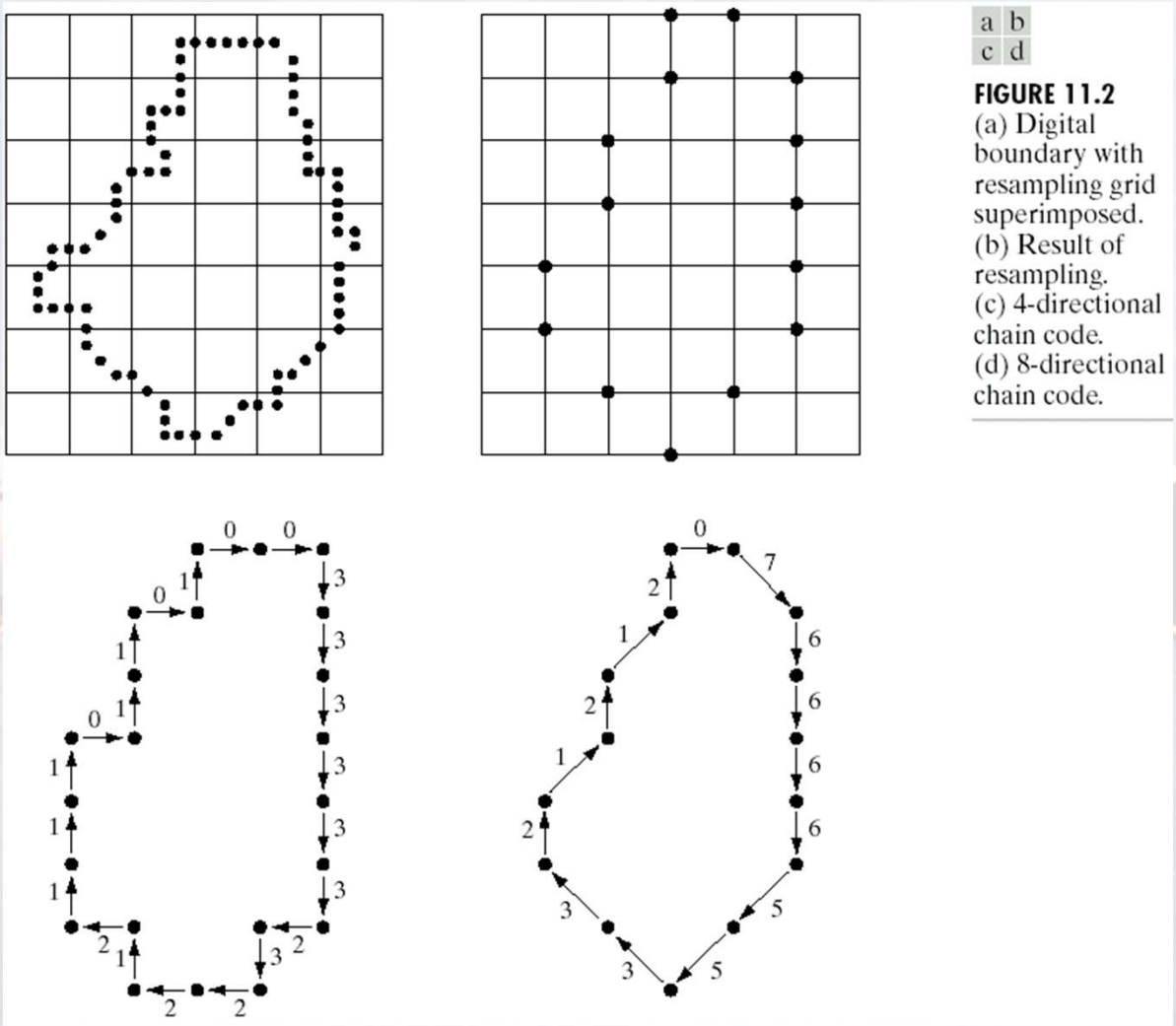
Chain code (Boundary descriptor)

It's possible to describe the boundary pixels of a shape by encoding the directions



a b

FIGURE 11.1 Direction numbers for (a) 4-directional chain code, and (b) 8-directional chain code.



a b
c d

FIGURE 11.2 (a) Digital boundary with resampling grid superimposed. (b) Result of resampling. (c) 4-directional chain code. (d) 8-directional chain code.

Chain code (Boundary descriptor)

You pick a starting point, and a direction (e.g. clockwise) and note down the directions that the boundary takes. In order to get fixed length results (and somewhat scale invariant), you can subsample the boundary points.

Rotation invariance can be achieved by taking the sequence's derivative.

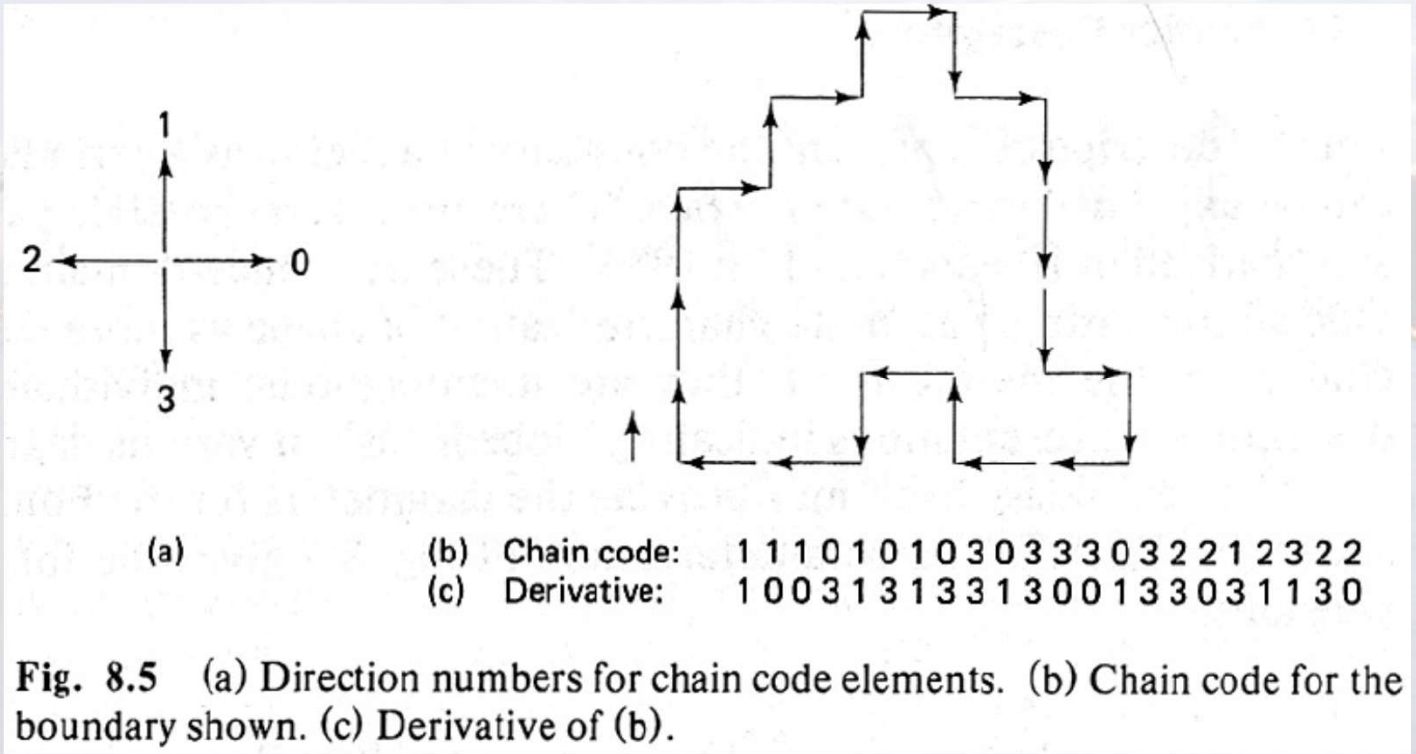
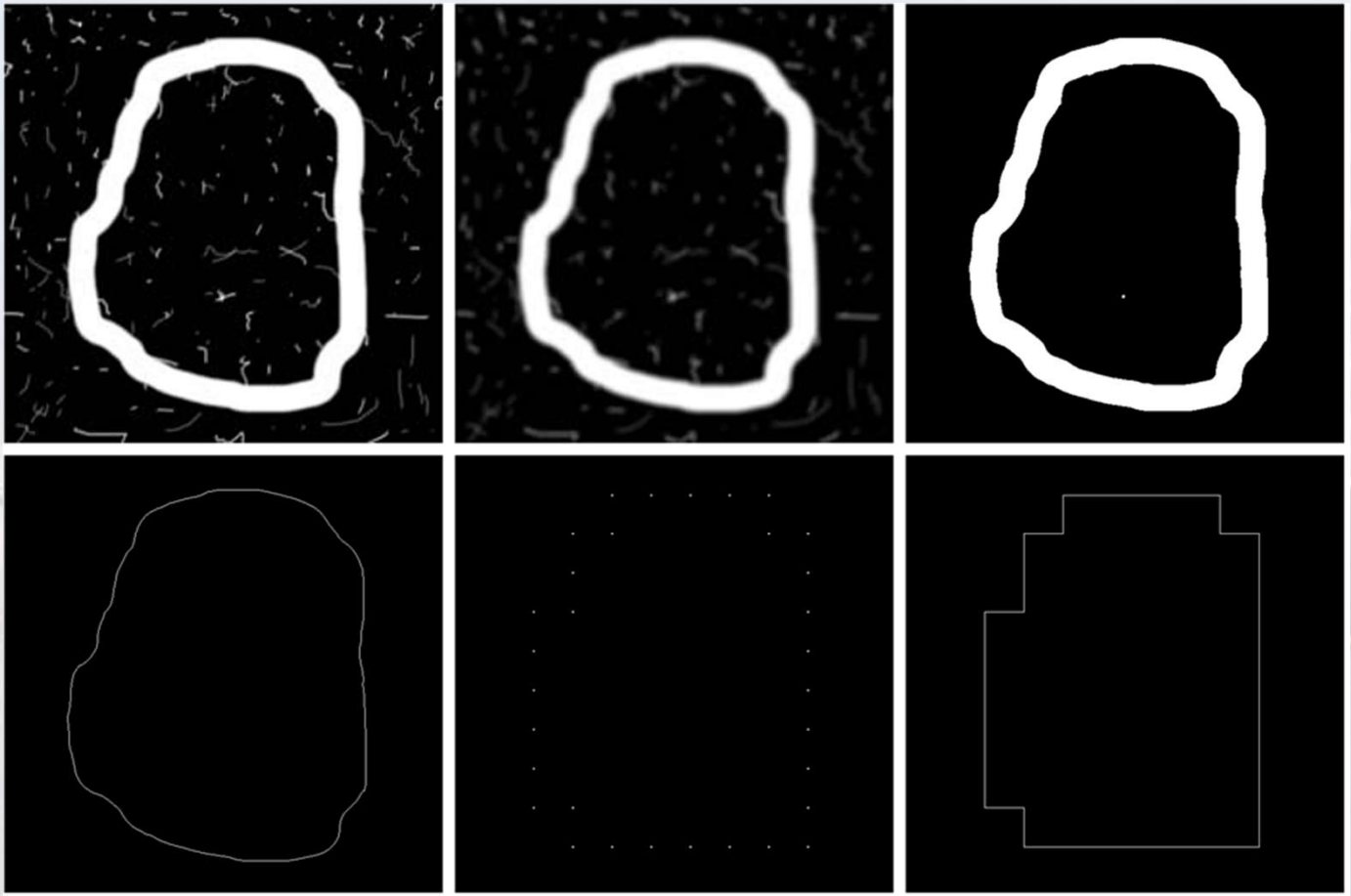


Fig. 8.5 (a) Direction numbers for chain code elements. (b) Chain code for the boundary shown. (c) Derivative of (b).

The effect of the starting point can be neutralized by treating it as a circular sequence.

Or you can take a histogram of the encoded directions.



a b c
d e f

FIGURE 11.2 (a) Noisy image. (b) Image smoothed with a 9 × 9 averaging mask. (c) Thresholded image. (d) Boundary of binary image. (e) Subsampled boundary. (f) Connected points from (e).

Fourier descriptor (boundary descriptor)

Let x_m and y_m be the coordinates of the m-th pixel on the boundary of a given 2D shape, containing N pixels. A complex number can be formed as $z_m = x_m + iy_m$, and the *Fourier Descriptor (FD)* of this shape is defined as the DFT of $z = (z_1, z_2, \dots, z_N)$:

$$Z(k) = DFT(z) = \frac{1}{N}\sum_{m=0}^{N-1} z_m e^{-i2\pi mk/N}, k = 0,1,2,..,N-1$$

FD is independent of translation, scaling, rotation and starting point.

If you use only the first M < N Fourier coefficients, that would mean omitting the higher frequencies (i.e. details) of the boundary, hence “capturing” only its overall shape properties.

Fourier descriptor

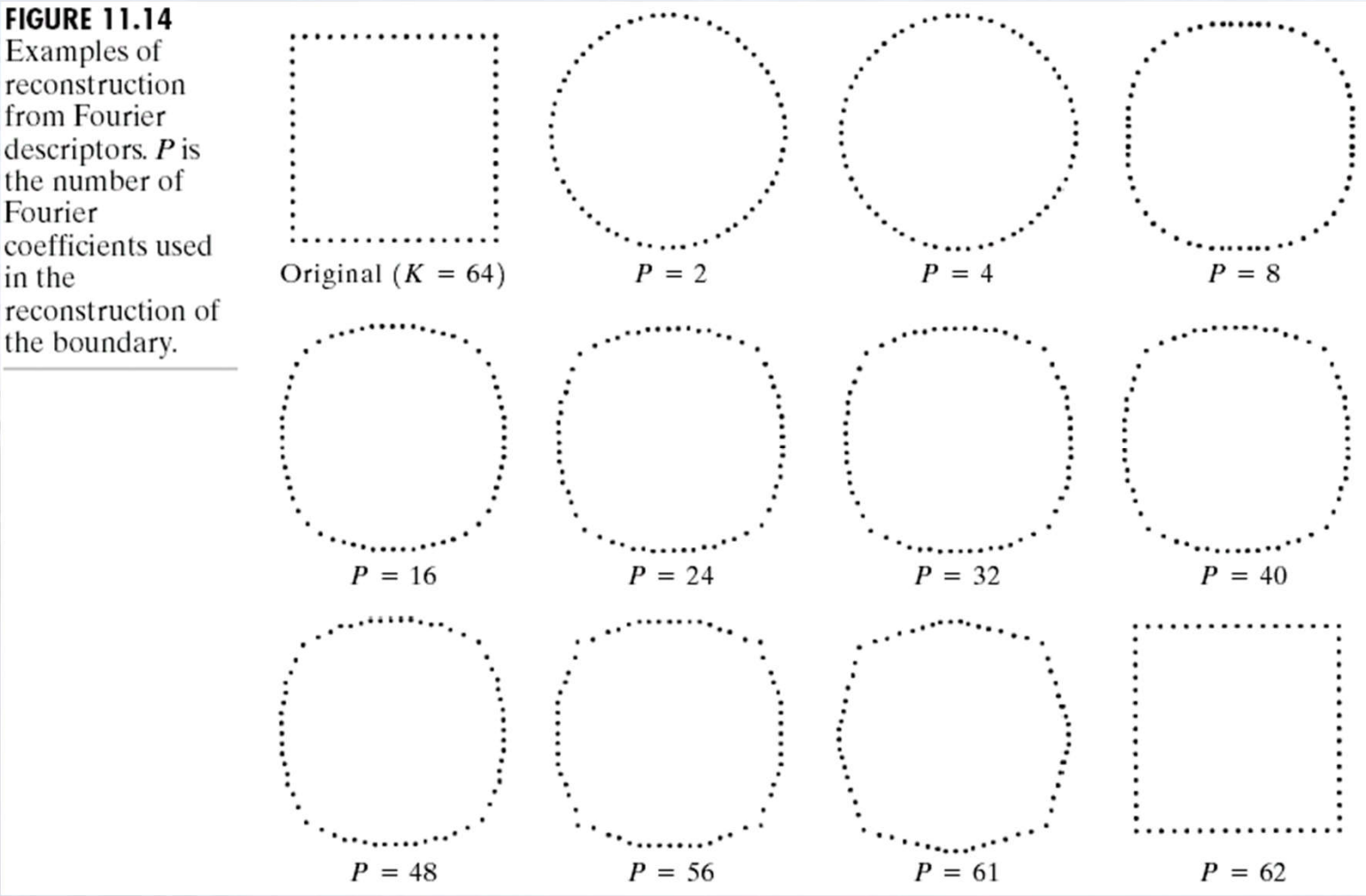
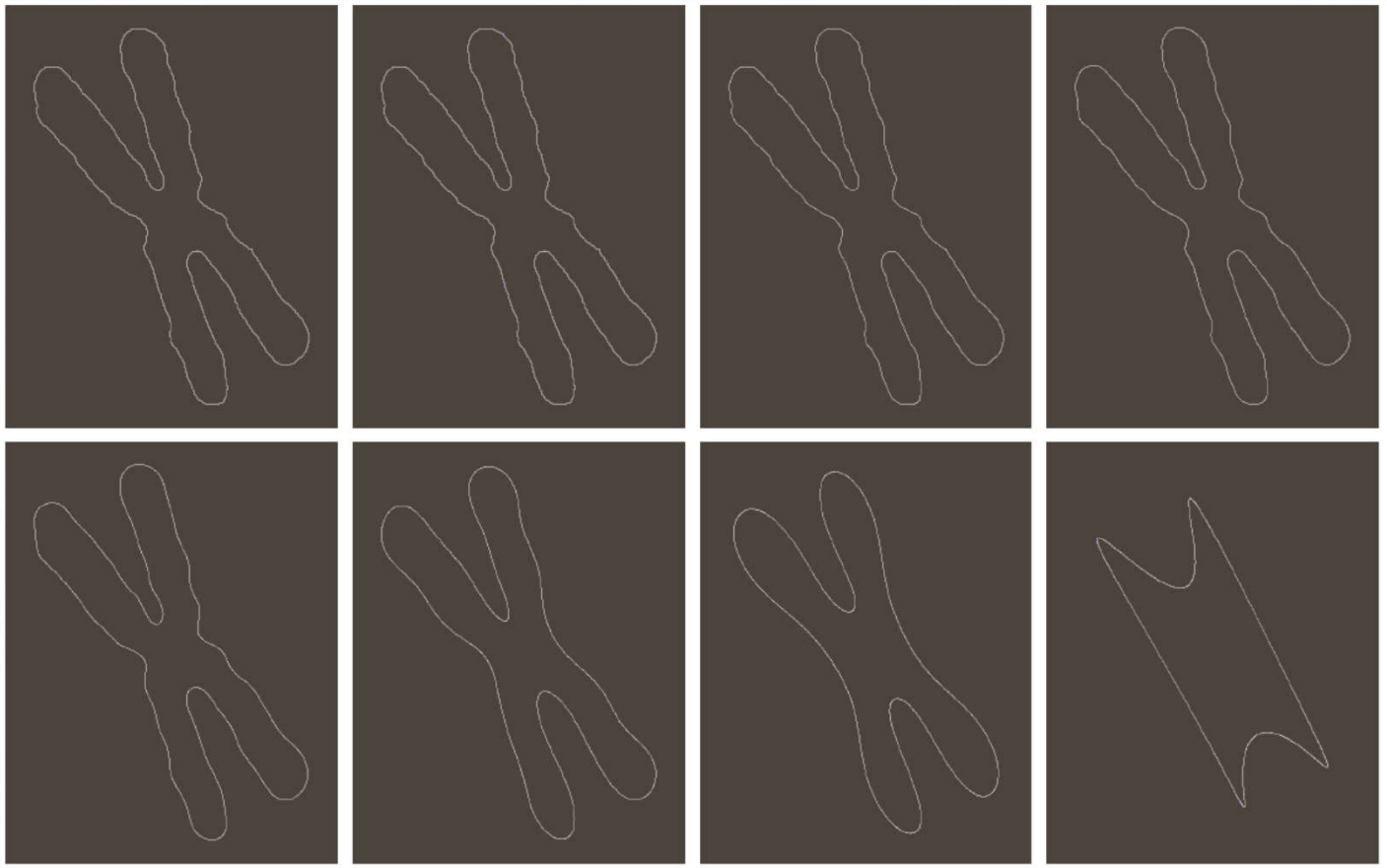


FIGURE 11.14 Examples of reconstruction from Fourier descriptors. P is the number of Fourier coefficients used in the reconstruction of the boundary.

Fourier descriptor



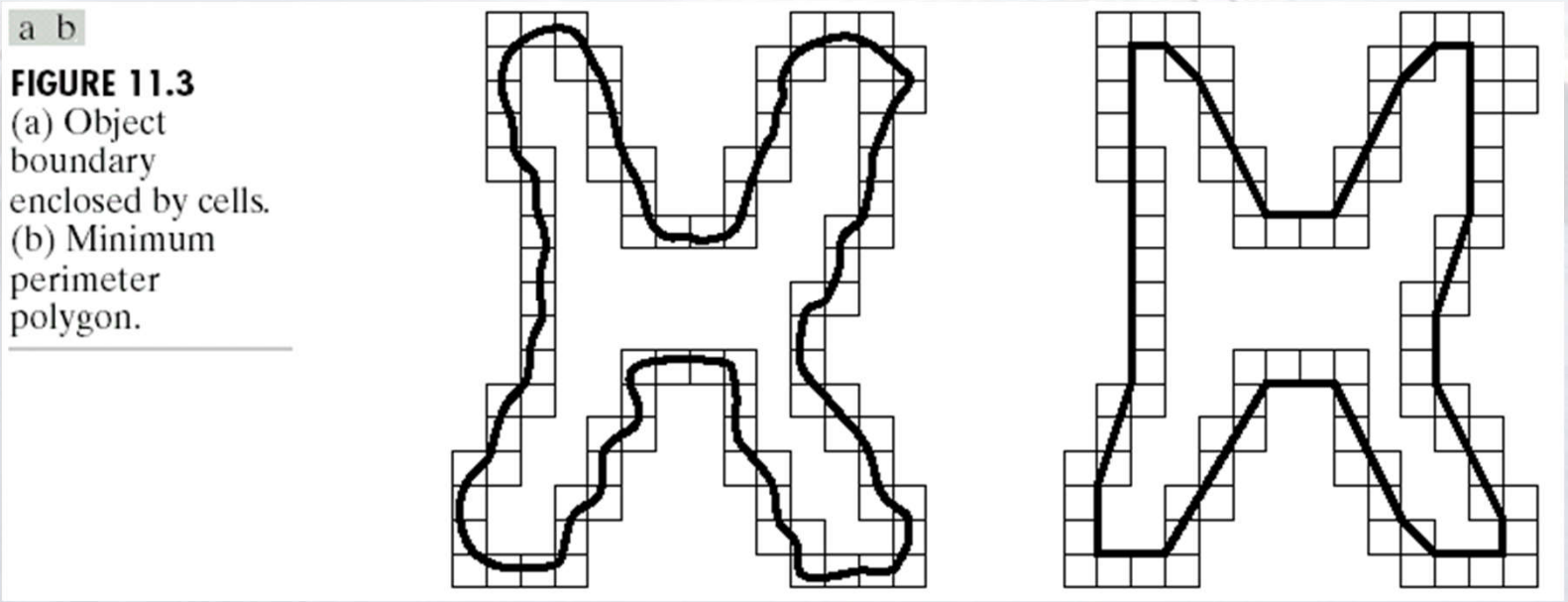
a b c d
e f g h

FIGURE 11.20 (a) Boundary of human chromosome (2868 points). (b)–(h) Boundaries reconstructed using 1434, 286, 144, 72, 36, 18, and 8 Fourier descriptors, respectively. These numbers are approximately 50%, 10%, 5%, 2.5%, 1.25%, 0.63%, and 0.28% of 2868, respectively.

Polygonal approximations

This is not a descriptor but an effective boundary simplification method, that can help you to both reduce its length and strip it off of unwanted details, while preserving the “main shape”. There are multiple ways for calculating it:

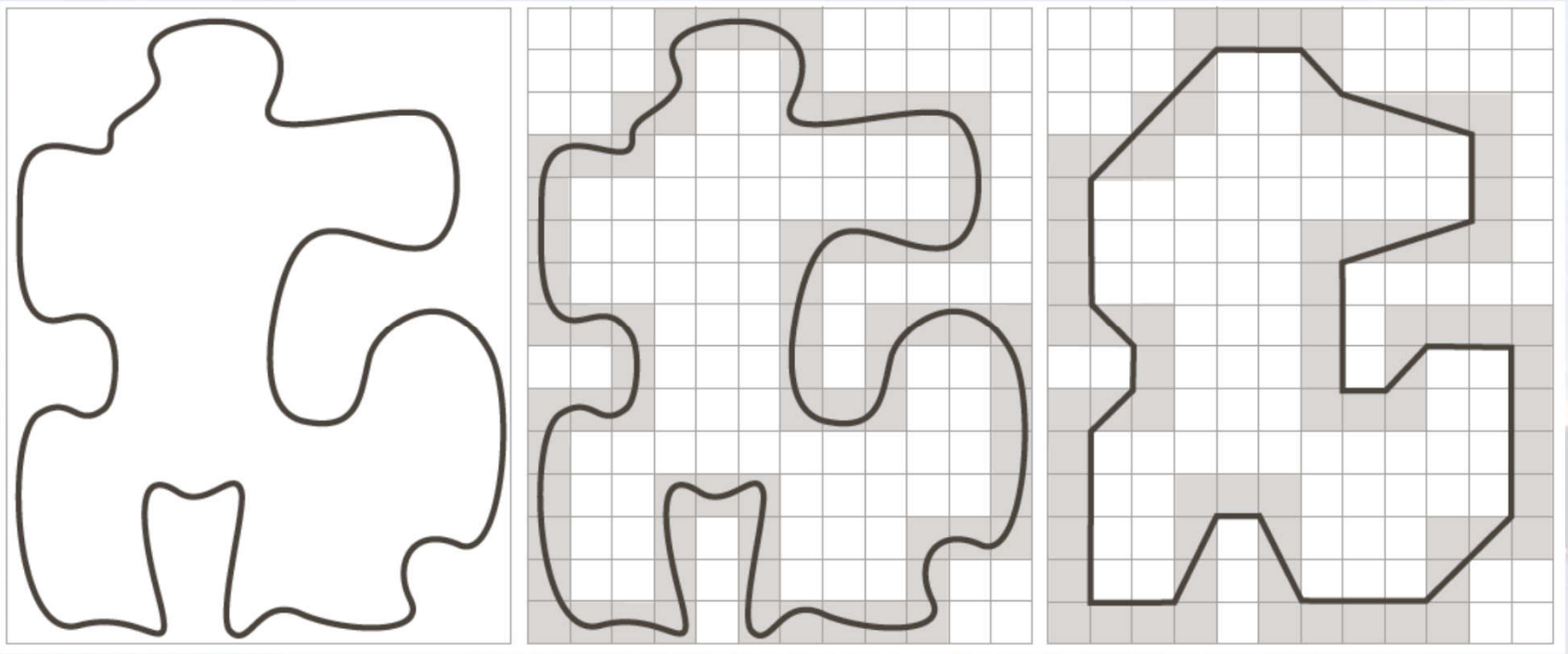
1) Enclose the boundary into a set of concatenated cells and compute the polygon of minimal perimeter that fits inside it.



a b

FIGURE 11.3 (a) Object boundary enclosed by cells. (b) Minimum perimeter polygon.

Image description



a b c

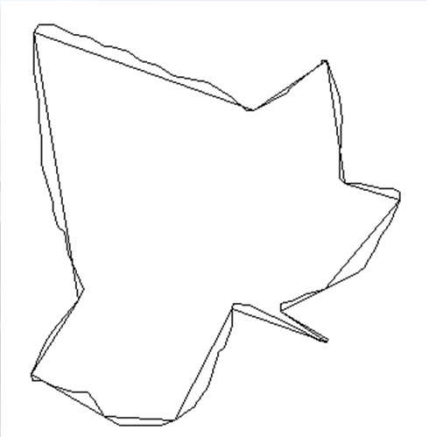
FIGURE 11.6 (a) An object boundary (black curve). (b) Boundary enclosed by cells (in gray). (c) Minimum-perimeter polygon obtained by allowing the boundary to shrink. The vertices of the polygon are created by the corners of the inner and outer walls of the gray region.

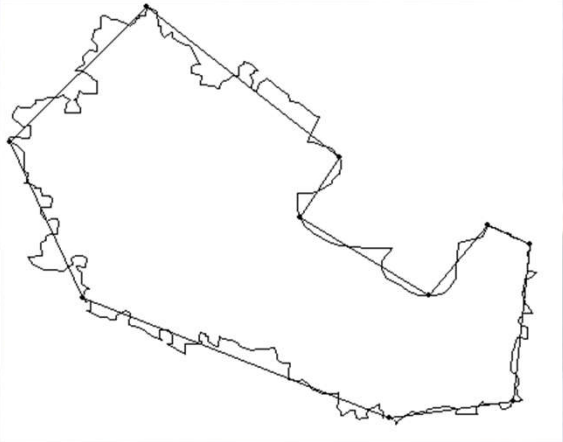
Image description

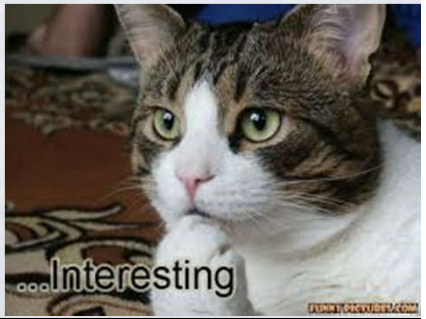
a b c
d e f
g h i

FIGURE 11.8
(a) 566 × 566 binary image. (b) 8-connected boundary. (c) through (i), MMPs obtained using square cells of sizes 2, 3, 4, 6, 8, 16, and 32, respectively (the vertices were joined by straight lines for display). The number of boundary points in (b) is 1900. The numbers of vertices in (c) through (i) are 206, 160, 127, 92, 66, 32, and 13, respectively.

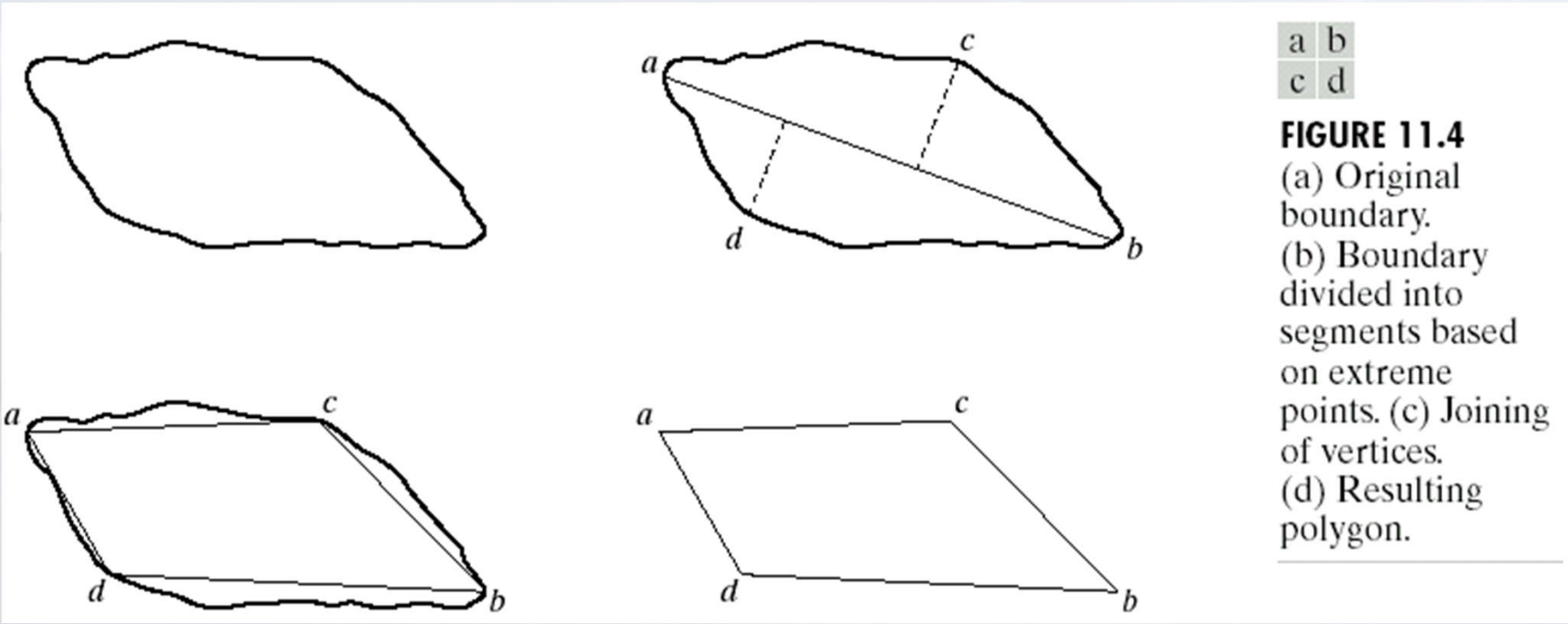
2) Alternatively, you can also merge points along the boundary, until the least square error exceeds a predefined threshold. The intersections of adjacent line segments will constitute the vertices of the polygon.







3) Or you can start with a segment connecting the farthest points on the boundary. If any point on the boundary has a perpendicular distance to the segment exceeding some threshold it forms a subdivision point; and repeat for every segment, until the threshold is no longer exceeded anywhere.

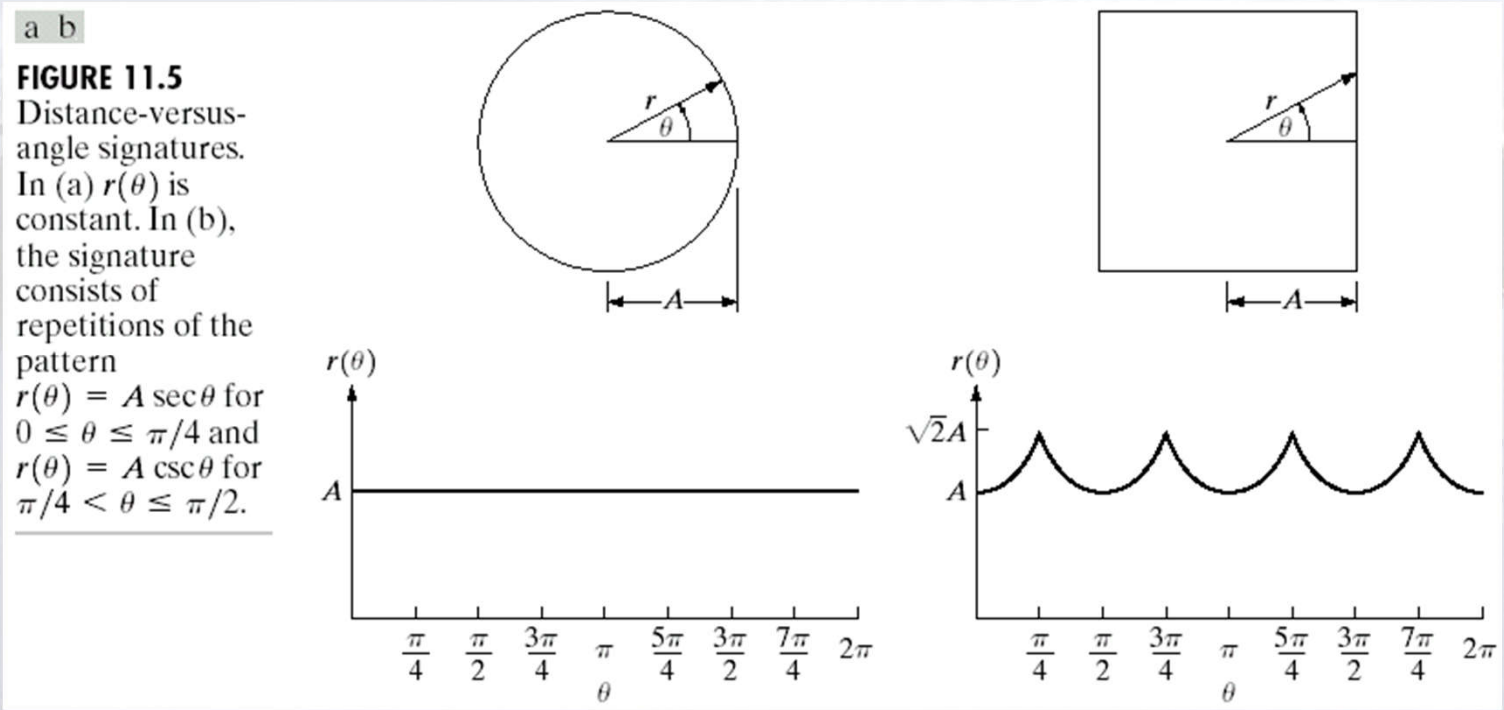


a b
c d

FIGURE 11.4 (a) Original boundary. (b) Boundary divided into segments based on extreme points. (c) Joining of vertices. (d) Resulting polygon.

Boundary signature

The boundary signature is a 1D function, formed by measuring the distance of the shape centroid to the boundary pixels. Scale and rotation invariance can be easily achieved by normalizing the distances w.r.t. the size of the shape and shifting the function w.r.t. some criterion. You may then exploit this function for calculating all kinds of statistical measures.



Moment invariants

They are statistical properties of connected regions in images, that are invariant to translation, rotation and scale.

Raw moment of order (i+j): $$M_{ij} = \sum_x \sum_y x^i y^j I(x,y)$$

Central moment of order (p+q): $$\mu_{pq} = \sum_x \sum_y (x-\bar{x})^p (y-\bar{y})^q f(x,y)$$

where $\bar{x} = M_{10}/M_{00}$ and $\bar{y} = M_{01}/M_{00}$. Central moments are translation invariant.

Translation and scale invariance is achieved through: $$\eta_{ij} = \frac{\mu_{ij}}{\mu_{00}^{\left(1+\frac{i+j}{2}\right)}}$$

for $(i+j) \geq 2$.

Moment invariants

As shown by Hu *et al*. (1962), translation, scale and rotation invariance can be obtained through:

$$I_1 = \eta_{20} + \eta_{02}$$

$$I_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

$$I_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

$$I_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

$$I_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

$$I_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$$

$$I_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

$$I_8 = \eta_{11}[(\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{21})^2] - (\eta_{20} - \eta_{02})(\eta_{30} + \eta_{12})(\eta_{03} + \eta_{21})$$

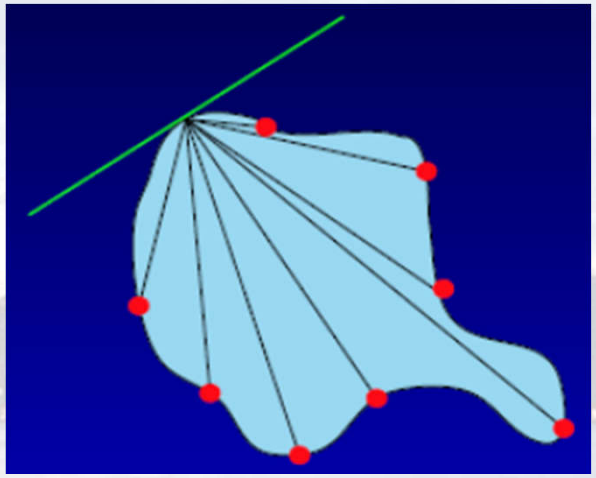
that are known as *Hu's moments*. There are many more moment types; e.g. Zernike (effective with face recognition).

Shape Context (2000)

A powerful descriptor used for finding corresponding points between the boundaries of two shapes.

Assume you have a boundary with N points.

Pick one reference point and calculate the distances to the remaining N-1 points.

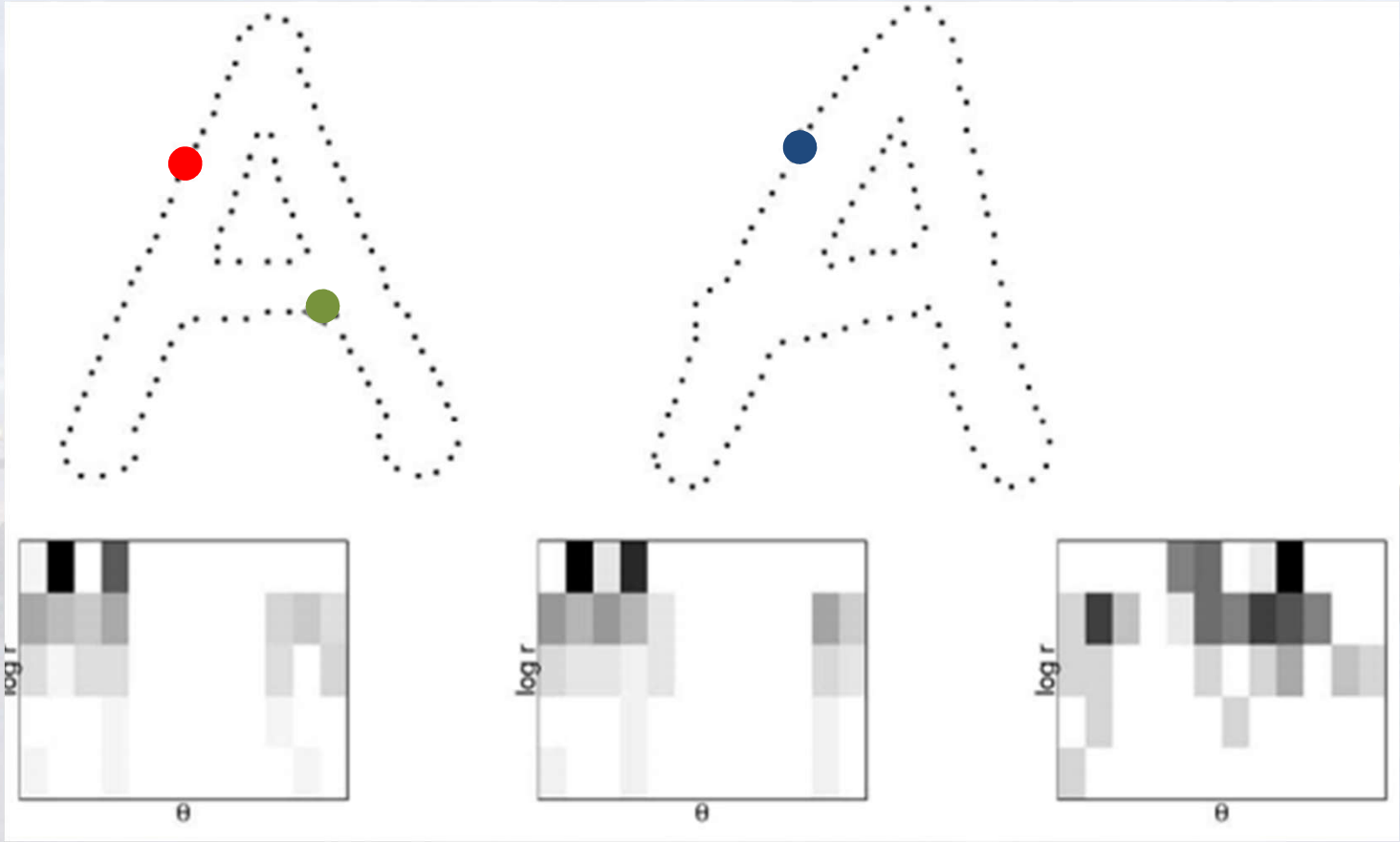


The histogram of distances describes that point.

$$h_i(k) = \#\{q \neq p_i \quad : \quad (q - p_i) \in \text{bin}(k)\}$$

In more detail you use the logarithm of distances, and calculate the histogram w.r.t. the angles between the tangent of the reference point and the boundary point.

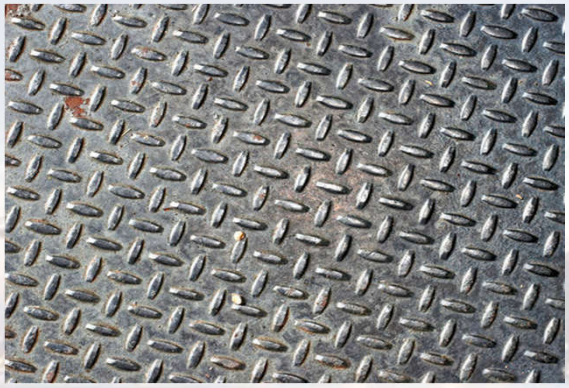
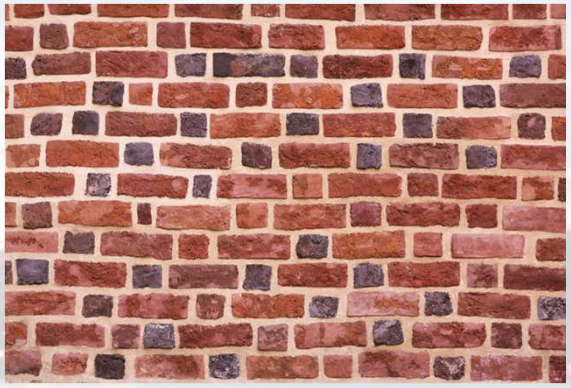
Shape context (2000)



Shape contexts for three points; with 12 bins for angles, and 5 bins for log of distances (dark denotes larger values). Notice the similarity between the first two

Texture: spatially adjacent basic structures, the placement and orientation of which are governed by certain generation rules.

No formal mathematical definition is available. It is however a fundamental property of objects.

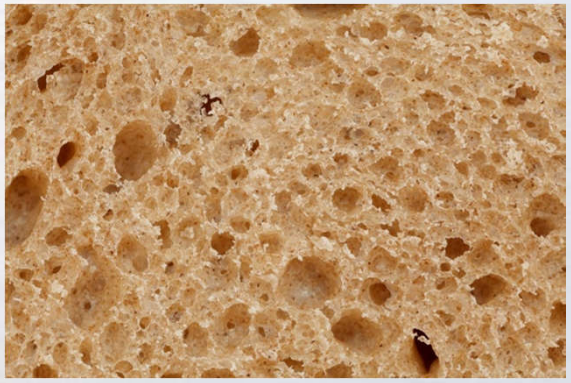
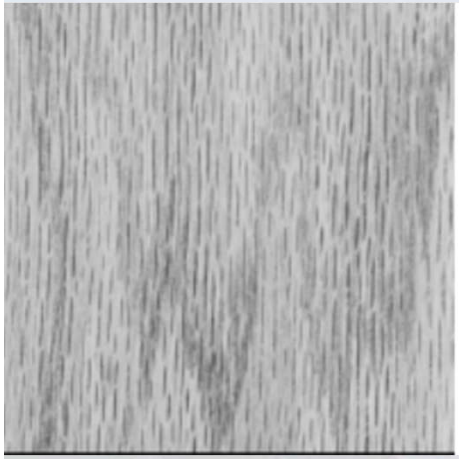



Image description

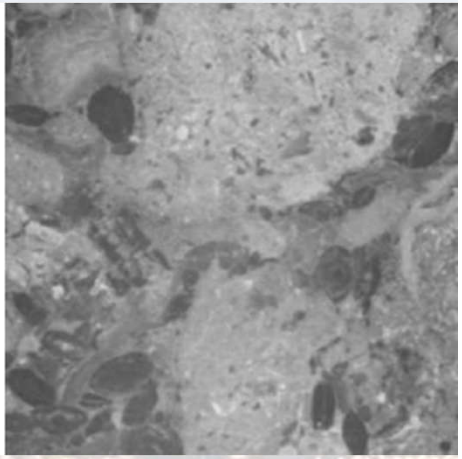
Textures can be classified according to the spatial distribution of their details into 4 categories:



Strongly ordered

Weakly ordered

Disordered



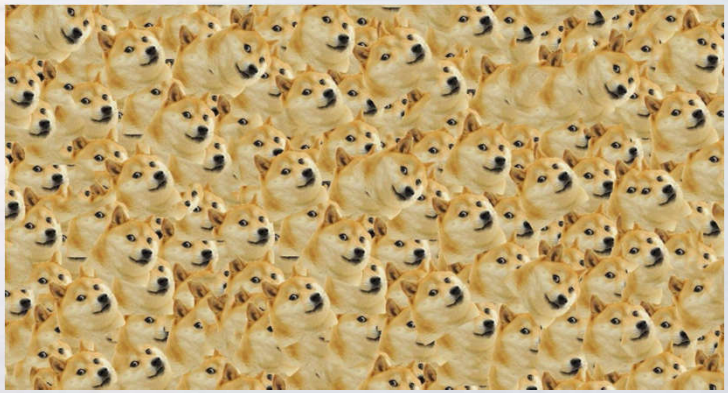
Compositional

It is claimed that the most distinguishing properties of textures are:

- Regularity (periodicity)
- Directionality
- Complexity
- Overall color and color purity

There exists a rich variety of **texture descriptors**, that aim to capture these properties. Texture descriptors can be categorized as:

- Structural
- Statistical
- Model based
- and Transform based



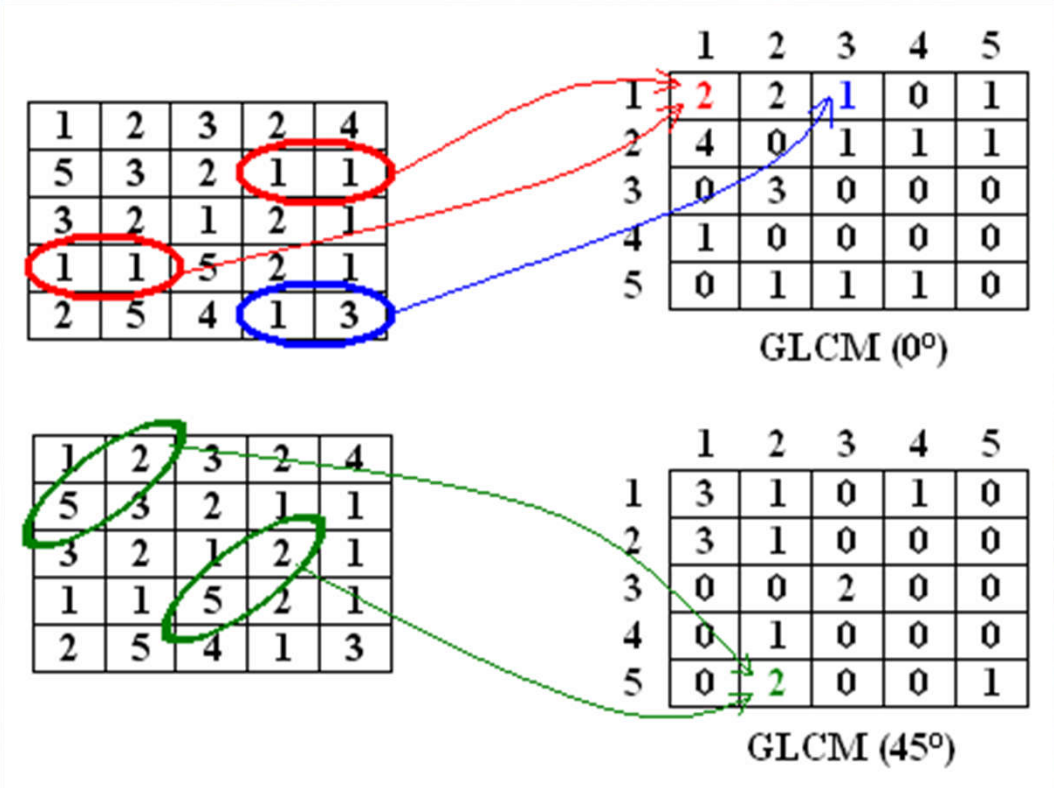
The **graylevel co-occurrence matrix** (GLCM) (1979)

More precisely, for an image f and a displacement vector $\boldsymbol{d} = [\Delta x, \Delta y]^T$, each element $C_{\Delta x,\Delta y}(i,j)$ of a GLCM represents the occurrence frequency of a pair of pixels with gray values i and j, separated by $\boldsymbol{d}$:

$$C_{\Delta x,\Delta y}(i,j) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\begin{cases}1, \text{if } f(x,y) = i \text{ and } f(x+\Delta x, y+\Delta y) = j \\ 0, \text{otherwise}\end{cases}$$

Beware that for an r-bit image, the matrix will have dimensions $2^r \times 2^r$. Which is why various statistical measures are then computed from it, in order to characterize the resulting matrix.

You are going to need one matrix per displacement vector.



Energy

$$\sum_{i,j} C(i,j)^2$$

Contrast

$$\sum_{i,j} (i-j)^2 C(i,j)$$

Entropy

$$-\sum_{i,j} C(i,j) \log_2 C(i,j)$$

Homogeneity

$$\sum_{i,j} \frac{C(i,j)}{1+|i-j|}$$

Correlation

$$\frac{\sum_{i,j}(i-\mu_x)(j-\mu_y)C(i,j)}{\sigma_x \sigma_y}$$

Autocorrelation

The autocorrelation function ϱ of an image I computes the dot product of the original image with its shifted self, for various shifts $[dr, dc]$

$$\rho(dr,dc) = \frac{\sum_{r=0}^{N}\sum_{c=0}^{N} I[r,c]I[r+dr,c+dc]}{\sum_{r=0}^{N}\sum_{c=0}^{N} I^2[r,c]} = \frac{I[r,c] \circ I_d[r,c]}{I[r,c] \circ I[r,c]}$$

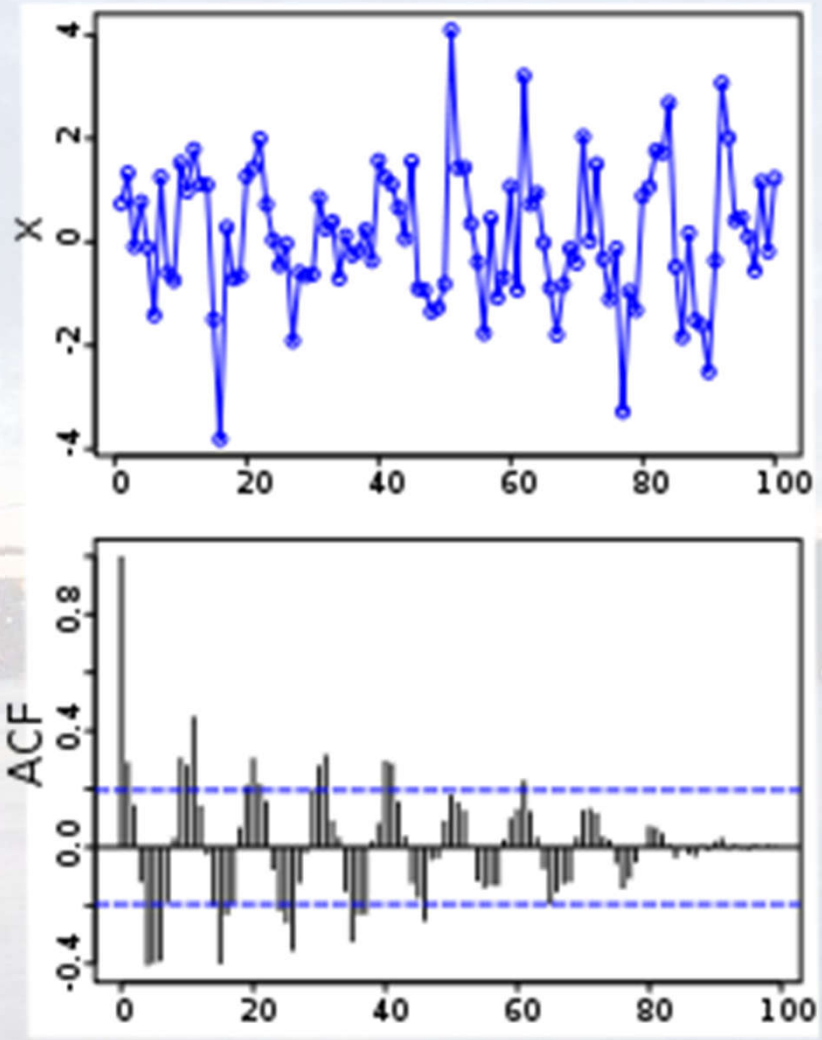
Strongly ordered texture: the function will have peaks and valleys at regular intervals.

Coarse texture: the function will drop slowly

Fine texture: the function will drop rapidly

Autocorrelation

1D example: a noisy sine wave with a wavelength of 10 units.



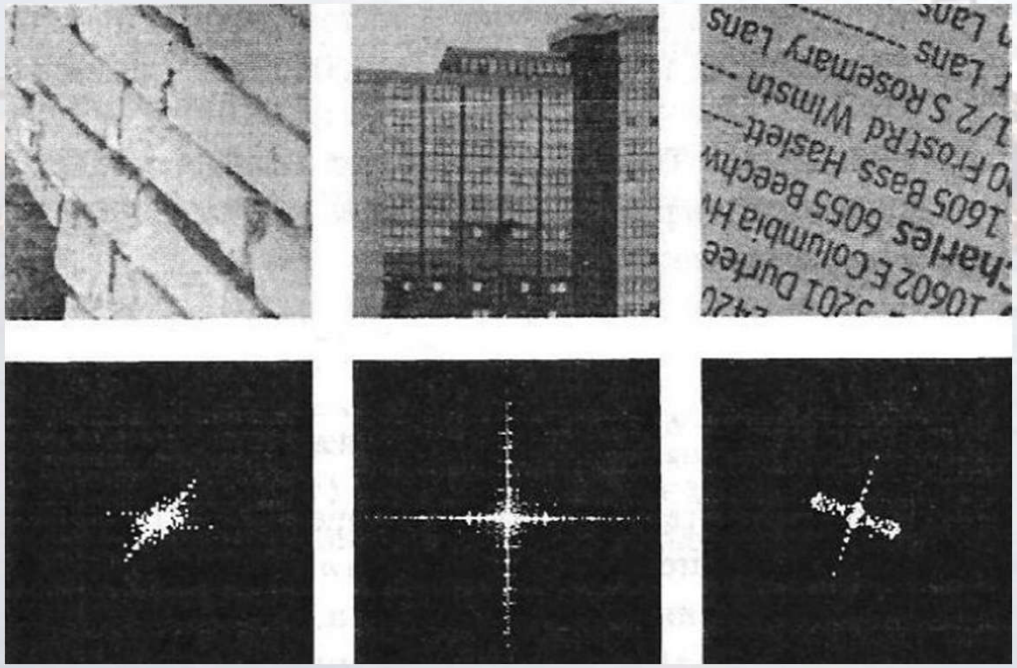
Notice the regular intervals between peaks.

Another classic tool for texture description is the **Fourier power spectrum**.

Since we are interested in capturing the input's periodicity, it's the ideal tool!

Think about it;

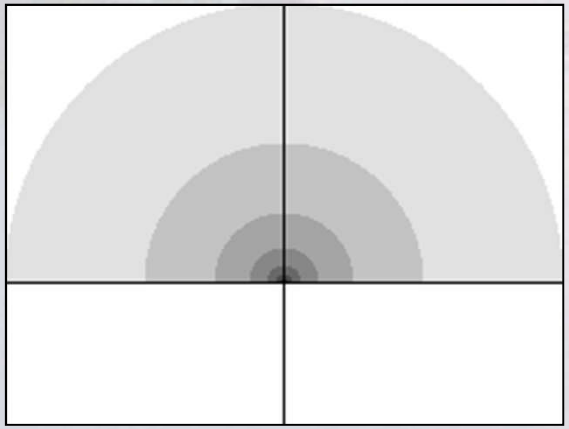
- Presence of peaks: regularity!
- Directionality: directional texture!
- Presence of high frequencies: fine texture!
- Location of peaks: spatial period of texture!

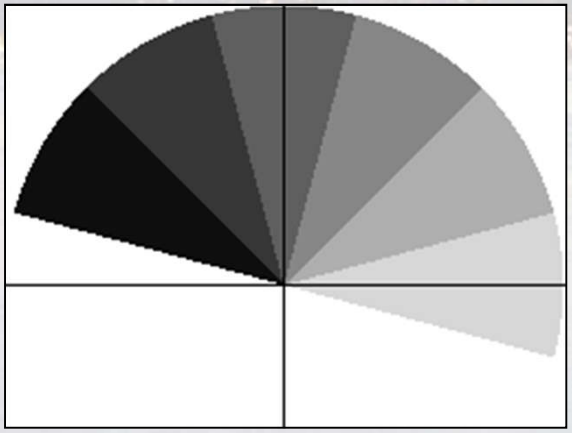


From a practical perspective, the power spectrum can be summarized through its intensities in circular rings and wedges.

$$ring_i = \sum_{r=r_i}^{r_{i+1}} \sum_{\theta=0}^{\pi} S(r,\theta) \qquad wedge_i = \sum_{\theta=\theta_i}^{\theta_{i+1}} \sum_{r=0}^{r_{max}} S(r,\theta)$$

Note that since images have real values the spectrum is symmetrical.

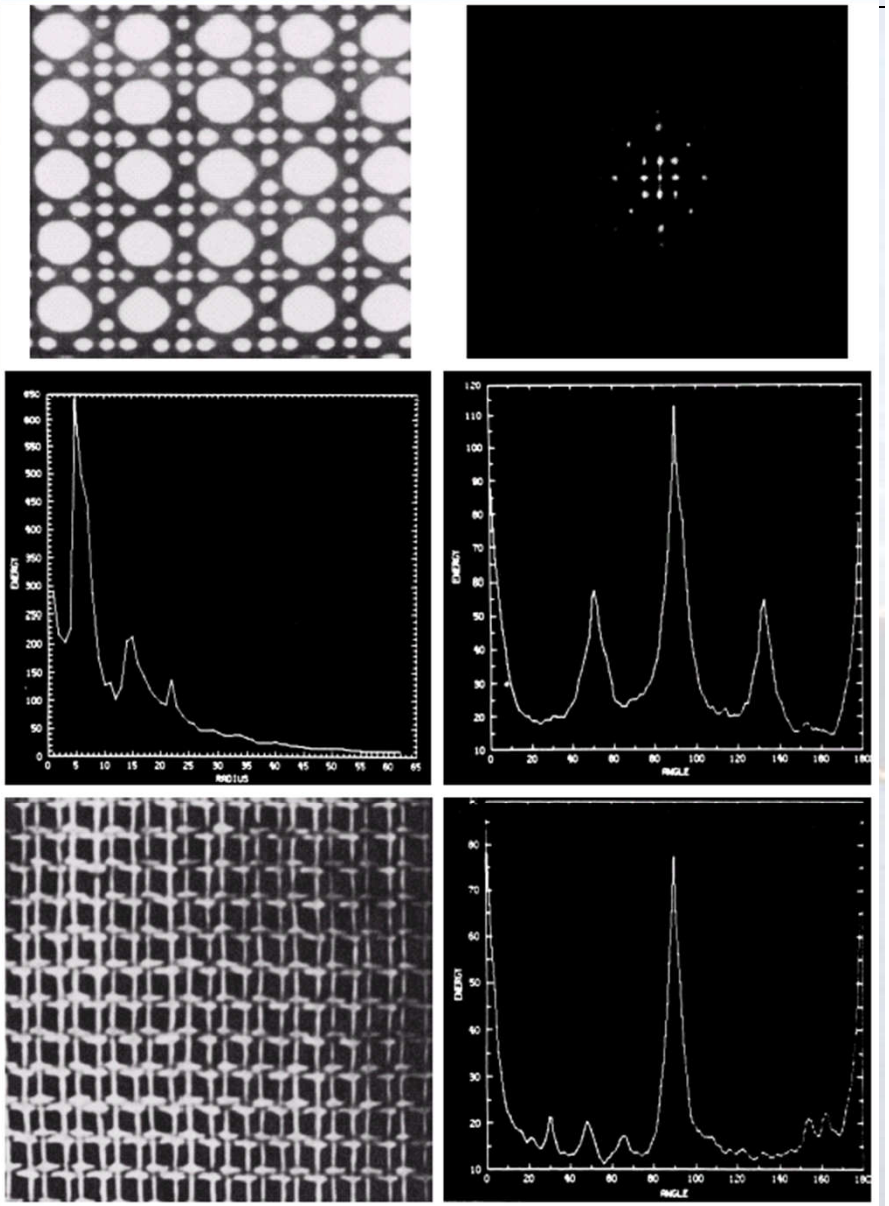




The angles are tilted in accordance with the HVS's sensitivities

Adapted from Selim Aksoy, Bilkent U.

Image description



a b
c d
e f

FIGURE 11.24 (a) Image showing periodic texture. (b) Spectrum. (c) Plot of $S(r)$. (d) Plot of $S(\theta)$. (e) Another image with a different type of periodic texture. (f) Plot of $S(\theta)$. (Courtesy of Dr. Dragana Brzakovic, University of Tennessee.)

Gabor features are …wait a second, first we need to talk about **Gabor filters**!

- A Gabor filter is basically a linear filter for edge detection.
- Its scale and orientation are tunable.
- Its response is similar to the brain's (first few layers of it at least) HVS response.
- It consists of a sinusoidal plane wave (in 2D) modulated by a Gaussian envelope of frequency W.

At 1D:

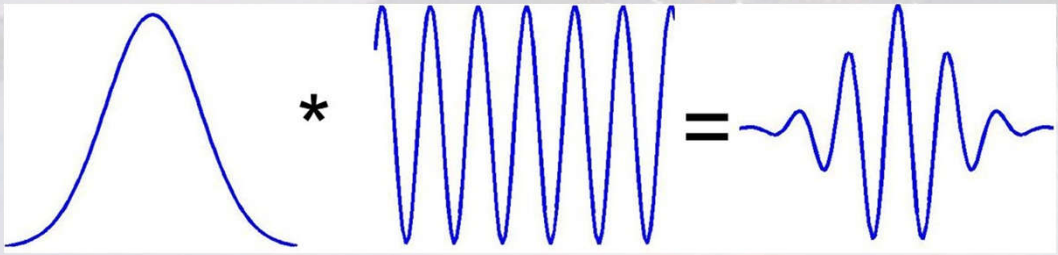
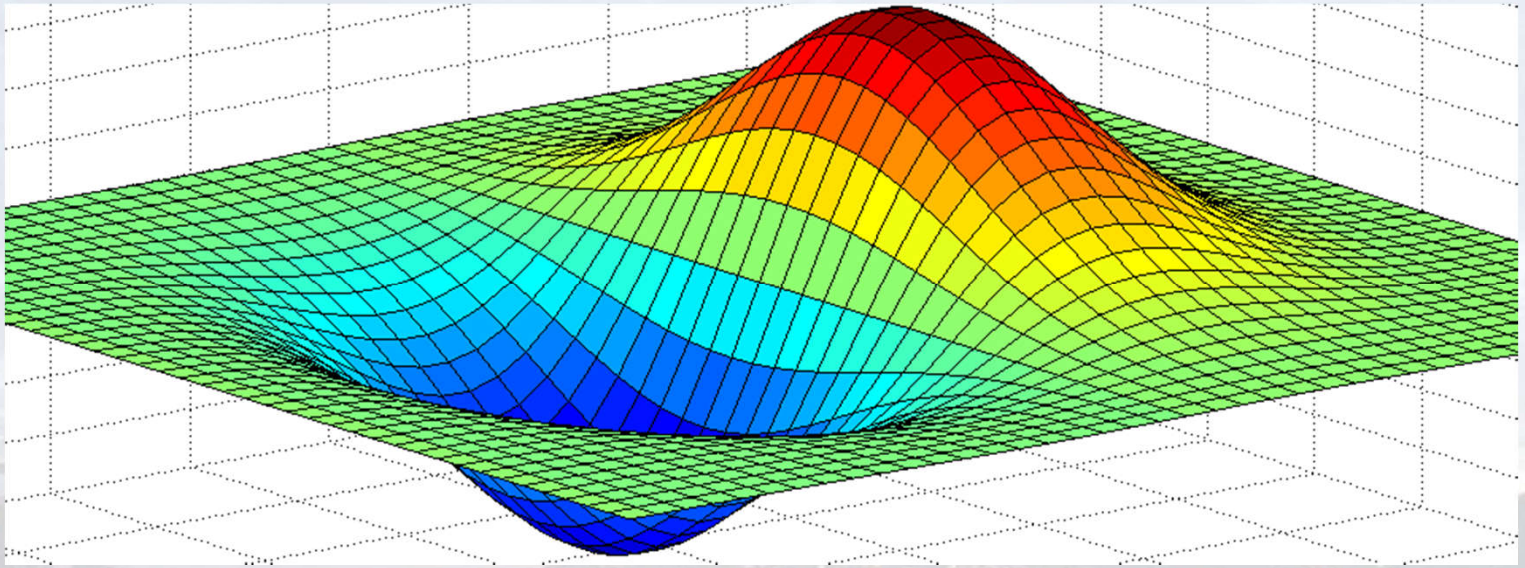


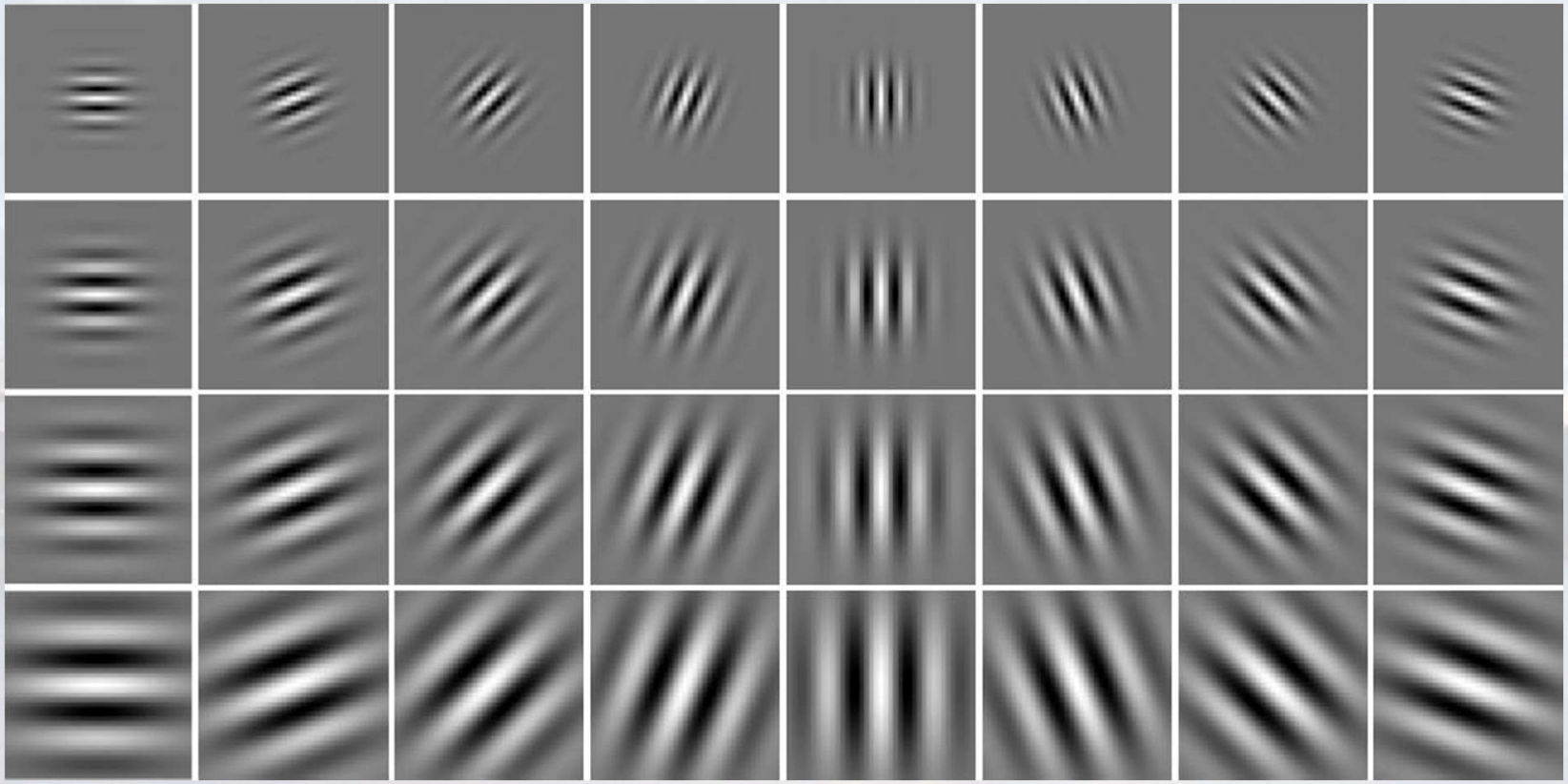
Image description



The 2D Gabor function:

$$g(x,y) = \frac{1}{2\pi\sigma_x\sigma_y}\exp\left[-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2}\right)+2\pi jWx\right]$$

Some Gabor masks at various scales and orientations (filter banks)

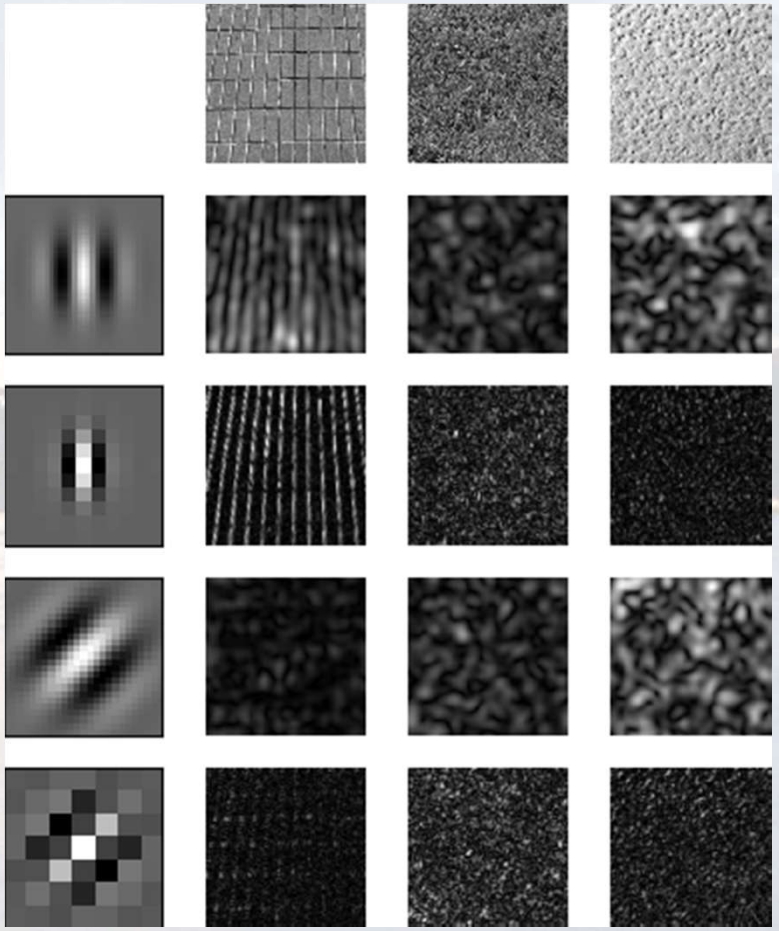


Gabor features

Gabor features are calculated through the Gabor filters by modifying the filter's scales and orientations, and recording the mean and standard deviation of the resulting filtered images:

$$gabor(f) = \{\mu_{11}, \sigma_{11}, \mu_{12}, \sigma_{12} \dots, \mu_{RS}, \sigma_{RS}\}$$

The question is often how to set the filter parameters; which scales, and which orientations to use?

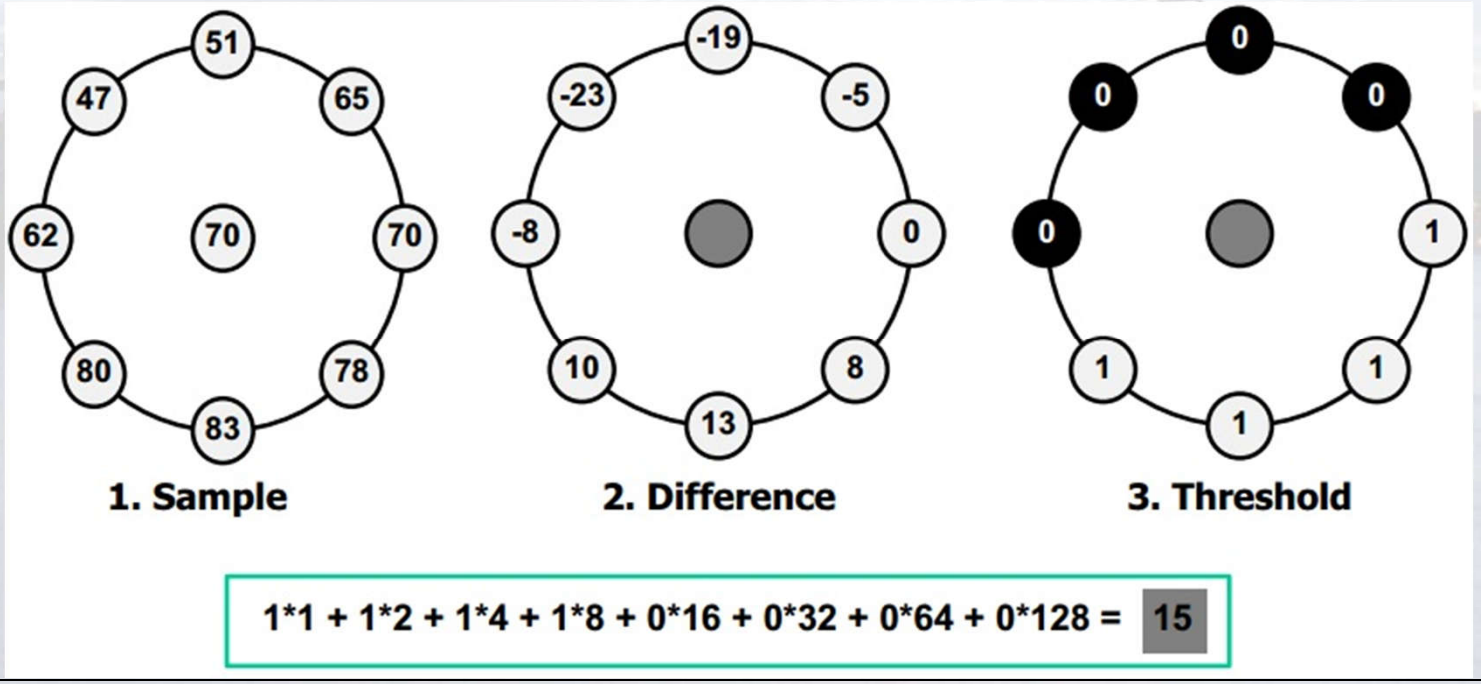


Local Binary Patterns (LBP) (1994)

A very popular texture descriptor; simple, efficient and effective.

To describe pixel q, visit the 8 circular neighbors q_i at radius r, and associate each either with 0 or 1, depending on whether they have a lower or greater value than q, and use those bits to form a byte:

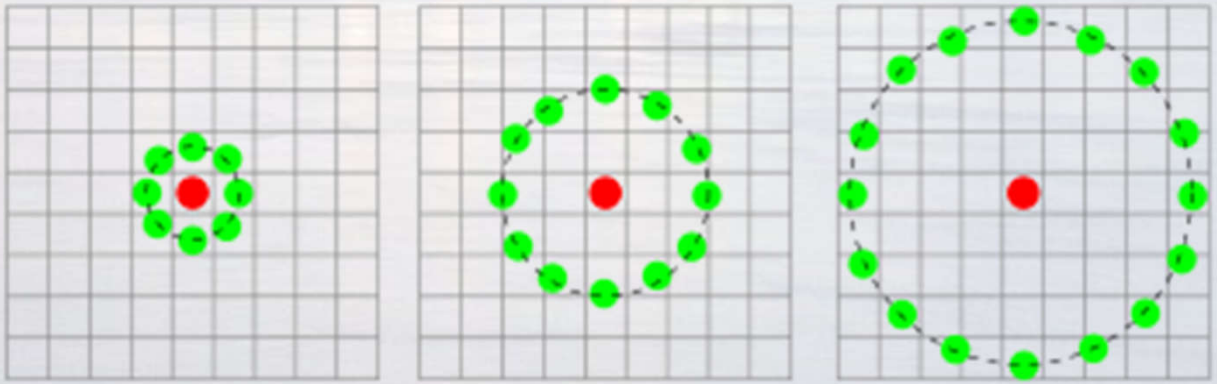
$$LBP_{p,r}(q) = \sum_{i=0}^{p-1} g(q - q_i)2^i \quad \text{and} \quad g(x) = \begin{cases} 0, x < 0 \\ 1, x \geq 0 \end{cases}$$



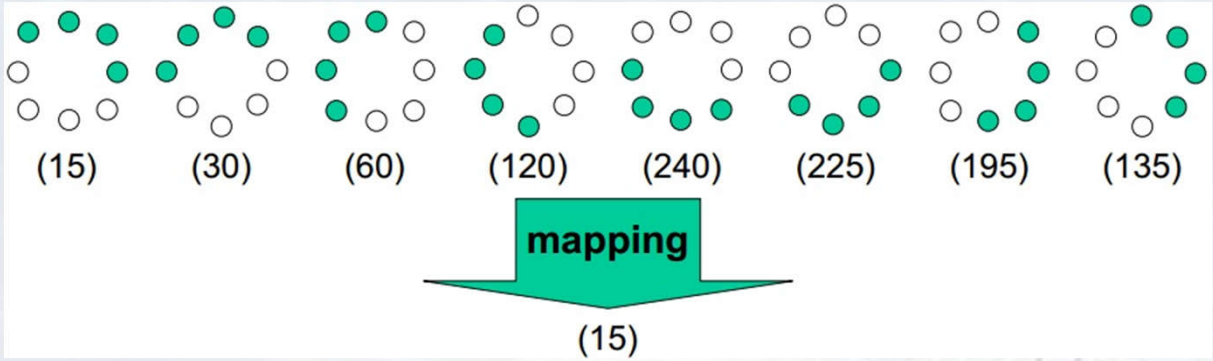
The image or region is then described through the histogram of the resulting 8bit numbers.

It can also be calculated for greater radii and neighbor numbers, and combined; thus becoming a multiscale descriptor.

Obviously there is some pixel interpolation to do.



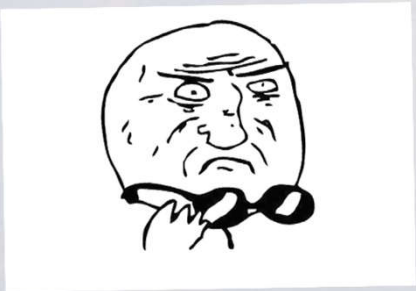
In order to equip it with rotation invariance, one way is to rotate every pattern into the lowest possible number:



It has A LOT of variants:

- Over-complete local binary patterns
- Transition local binary patterns
- Direction coded local binary patterns
- RGB local binary patterns
- Variance local binary patterns

etc.



Morphological texture description

The morphological arsenal has many powerful texture descriptors, two of which are: **granulometry** and **morphological covariance**.

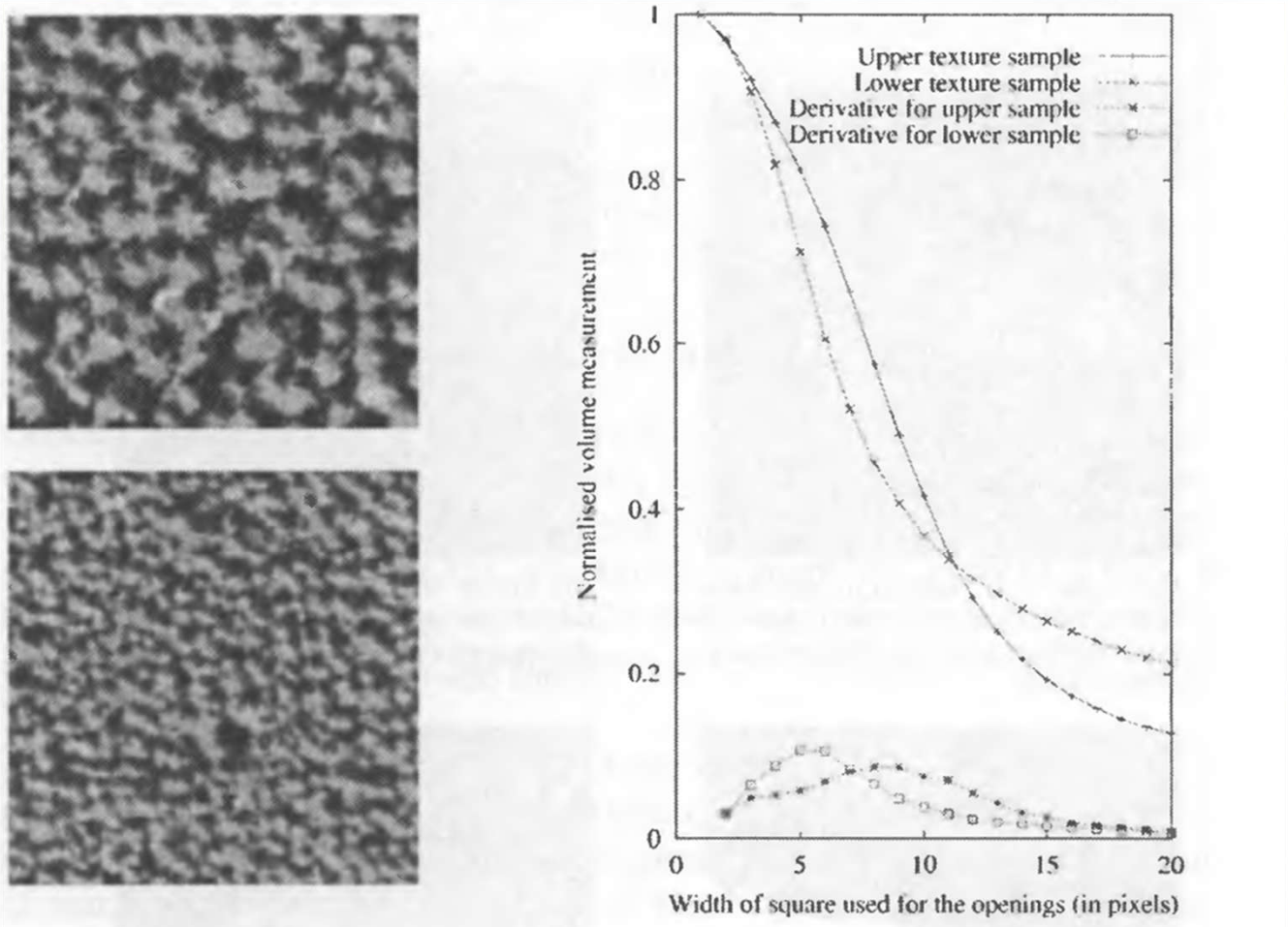


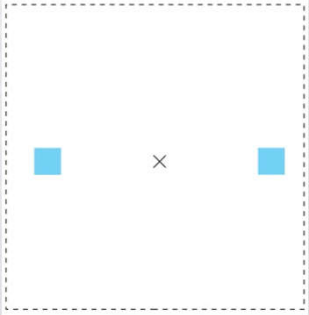
Fig. 11.4. *Left:* two samples of a 1 m resolution panchromatic IKONOS satellite image showing two forest stands of different age (crown size). *Right:* their granulometric curves by opening using squares of increasing width.

Morphological covariance is calculated by applying erosions with SEs($P_{\vec{v}}$) consisting of only two points, separated by a vector $\vec{v}$, for various directions and norms of $\vec{v}$. And every filtering result is characterized by the volume (Vol) of the image; i.e. sum of pixel values:

$$K(f, P_{\vec{v}}) = Vol(\varepsilon_{P_{\vec{v}}}(f))/Vol(f)$$

$$Vol(f) = \sum_{p} f(p)$$

It provides information on the periodicity, coarseness and directionality of textures.



Examples using a pair of points separated by horizontal vectors of various lengths:

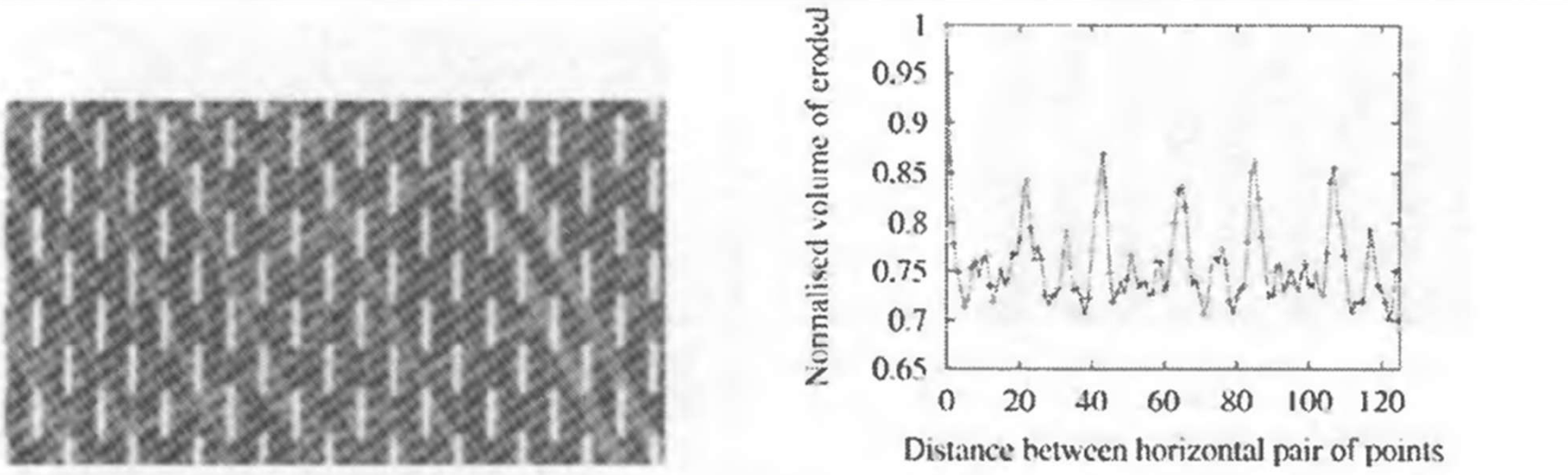


Fig. 11.10. Use of covariance for determining the period of periodic texture elements. *Left:* input image showing a textured region. *Right:* normalised sum of grey levels of erosions by the horizontal pair of points versus the distance separating the points.

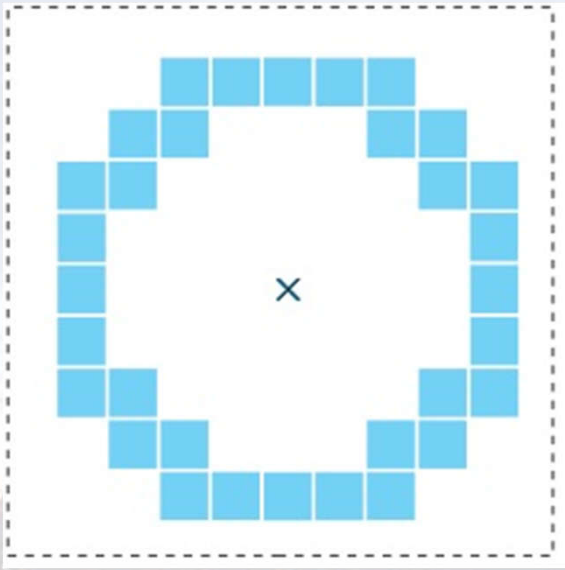
Adapted from P. Soille

Circular covariance (2012) equips morphological covariance with rotation invariance:

$$f \rightarrow \{T_{B_i}(f)\}_{1 \leq i \leq n}$$

$$\forall\, p, \quad L(p) = \arg \max_{1 \leq i \leq n} \{ \left| f(p) - [T_{B_i}(f)](p) \right| \}$$

$$\mathrm{CCH}_{T,n,\#1} = [p_1, p_2, \ldots, p_n]^T \in [0,1]^n,$$
$$p_i = P(L(p) = i), \quad i \in [1,n]$$



where T is a morphological operator or filter,
and B_i a circular SE of radius i.

Implemented often with dual operators (e.g. opening and closing), the outcomes of which are concatenated to form the final feature vector.

There are many more texture descriptors; e.g. fractals, Markov random fields, etc.

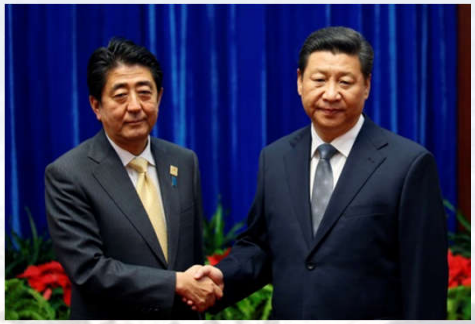
This was a short tour of global content descriptors. From this point on the tour continues with advanced contemporary descriptors:

- Object descriptors: e.g. HOG, etc.
- Local descriptors and visual vocabularies: SIFT, BoW, SURF, MSER, VLAD, etc.
- Deep descriptors: CNN, SAE, etc.

But they belong “mostly” to the realm of computer vision/pattern recognition and are out of our scope.

Content-based image retrieval (CBIR)

- You are working at a TV channel's control room, there's a news flash and your manager asks you to pull from the archive photos showing a handshake of the heads of state of China and Japan.



X. Jinping & Sh. Abe

- You come across an interesting piece of clothing in the street, in a movie, or maybe on TV, and you want the same or similar for yourself.

- Find on your laptop the video of “Mahmut amca dancing zeybek at Zeliha’s wedding”

More examples can be derived for:

- Medical imaging archives
- Art collections and museums
- The World Wide Web: google, flickr, and more.

Large visual databases are increasingly common. Yet, even the simplest search is far from trivial.

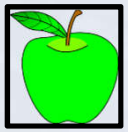
Content based image/video retrieval aims to provide the necessary tools for exploiting and managing effectively these large visual repositories.

The basic problem is how to retrieve the corresponding samples to a given query.

The query can in the form of

- text: “chair”
- an example
- or even a sketch

Bare minimum of a retrieval system:



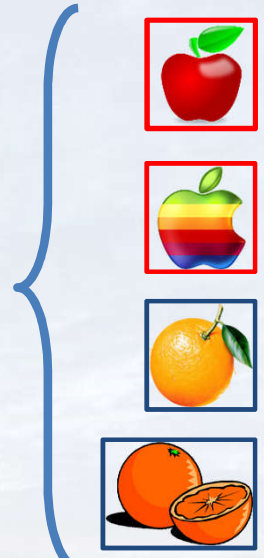
query

$A = [0.3,0.5,0.5]^T$

find the most similar feature vectors to A

$[0.3,0.4,0.5]^T$
$[0.2,0.5,0.4]^T$
$[1.0,2.6,3.8]^T$
$[1.2,2.4,4.1]^T$

feature vector database



show the top N most similar images

The core consists of two elements:

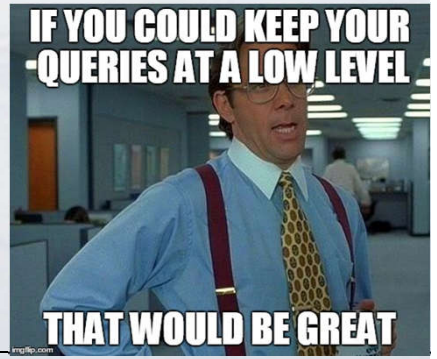
content description & **similarity measurement**

The content descriptors extract image characteristics describing various **low-level features**; e.g. edges, texture, color, spatial arrangement, etc.

If the user also formulated low-level queries such as “retrieve images with a lot of homogeneous blue on top and rough green texture in the center” that would be great!

BUT, the users most often formulate queries with high level concepts in mind; e.g. a fighter jet, blooming flowers, traditional Turkish paintings of mid-19th century, etc.

The gap between the low-level features and the high level concepts is called the **semantic gap**. Bridging this gap is a major issue.



There is a rich variety of similarity measures (metrics) that can be used among feature vectors…

$$d_{Bhattacharyya} = \sqrt{1 - \sum_{i=1}^{d} \frac{\sqrt{f1_i f2_i}}{\sqrt{\sum_{j=1}^{d} f1_j \sum_{k=1}^{d} f2_k}}}$$

$$d_{chi-square} = \sum_{i=1}^{d} \frac{(f1_i - f2_i)^2}{f1_i + f2_i}$$

$$d_{correlation} = \frac{\sum_{i=1}^{d} f1'_i f2'_i}{\sqrt{\sum_{j=1}^{d} f1'_j f2'_j}}$$

$$\text{where } f' = f - \frac{1}{d}\sum_{i=1}^{d} f$$

$$d_{cosine} = \frac{\sum_{i=1}^{d} f1_i f2_i}{\sqrt{\sum_{j=1}^{d} f1_j^2 \sum_{k=1}^{d} f2_k^2}}$$

$$d_{innerproduct} = \sum_{i=1}^{d} f1_i f2_i$$

$$d_{intersection} = \sum_{i=1}^{d} \min(f1_i, f2_i)$$

$$d_{L1} = \|f1 - f2\|_1 = \sum_{i=1}^{d} \|f1_i - f2_i\|$$

$$d_{L2} = \|f1 - f2\|_2 = \sqrt{\sum_{i=1}^{d} (f1_i - f2_i)^2}$$

…and more.

And how do you measure the performance of a retrieval systems?

There are many ways; two widely used measures are **precision** and **recall**.

$$precision = \frac{number\ of\ relevant\ samples\ among\ the\ retrieved}{number\ of\ retrieved\ samples} = \frac{TP}{TP+FP}$$

$$recall = \frac{number\ of\ relevant\ samples\ among\ the\ retrieved}{number\ of\ relevant\ samples} = \frac{TP}{TP+FN}$$

Say you have a dataset with 100 apples and 9 oranges. You query an apple and retrieve the top 8 images; out of which 6 are apples and 2 are oranges.

Your precision then would be $p = \frac{6}{8}$ and your recall $r = \frac{5}{100}$

If we go beyond a core CBIR system then we encounter additionally:

- Machine learning algorithms

If the number of image labels is limited, the CBIR can train a classifier to learn the different image classes.

- Relevance feedback

The user provides positive and/or negative feedback concerning the retrieval results, and the system exploits that feedback to improve its performance.

- Indexing systems

Finding the most similar feature vectors in a large data warehouse is no trivial task. Advanced hierarchical indexing systems are used to provide real-time retrieval performances.

….and more.