Computer Vision

Module 4
Feature extraction

Krzysztof Krawiec
http://www.cs.put.poznan.pl/kkrawiec/

Poznan University of Technology, 2021

Module outline

1. Introduction
2. Geometric features
    a. Simple geometric features
    b. Signature
    c. Skeleton
    d. Chain codes and edge description in the frequency domain
    e. Polygonal approximations
    f. Moments
3. Non-geometric features
    a. Texture analysis
4. Related topics
    a. Hough transform
    b. Voronoi diagram
    c. Fractal dimension

The concept of image feature

The term ‘feature’ has at least two meanings/interpretations in computer vision:

1. A value (usually a scalar) calculated from an image (or field of view, region of interest, ROI)
   - E.g. object perimeter, average brightness, histogram kurtosis, ...

2. A fragment of an image (usually small) characterized by certain properties (usually predetermined).
   - May occur multiple times in the image.
   - Examples: edge, corner, "blob", ...

In this module we adopt interpretation 1. We will return to interpretation 2 in Module 5 (Feature Detection and Feature Descriptors).

Desirable characteristics of feature extraction

Design an image/object description that is:

1. relevant in a given context/application,
2. has high information content ("compactness"),
3. is invariant under various image transformations
    - most often due to translation (T), scaling (S), rotation (R),
    - independence from other distortions (e.g. resulting from transformations in the 3D space of the scene).

Implications:

- We tend to use only *certain features* in specific applications (although there are also universal features).

An example of an "anti-feature" (cf. the concept of *anti-patterns*): brightness of a single pixel.

Geometric and non-geometric features

Geometric features: capture the shape of the object (e.g., area, perimeter, etc.).

- Can be calculated from the contour of the object (e.g., including holes in the object).

Non-geometric features: depend on the interior characteristics of objects. Most often reflect the following image properties:

- brightness,
- color, hue,
- texture.

Discrete geometry

More specifically: digital geometry.
A branch of mathematics and computer science concerned with geometric objects represented on rasters.

Main challenge: fundamental differences between continuous and discrete geometry, e.g.

- different definition of a point (a pixel has finite spatial dimensions),
- no classical concept of edge (edges cannot be infinitely thin),
- limited set of possible distances between points.

Geometric features

Preliminary assumptions

In discrete geometry, we delimit the objects with contours.

- Contour = a discrete approximation of the object edge; a sequence of object points such that each point has at least one non-object neighbor.

Methods of determination:

- Morphological operations (e.g., top-hat)
- Specialized contour tracking algorithms

Object contour can be obtained with the findContour()[1] function in OpenCV.

We usually assume that contours are closed.

Preliminary assumptions

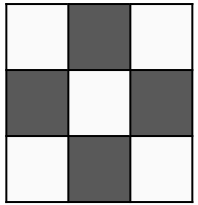
Contour = an ordered sequence of pixels representing the ‘silhouette’ of an object

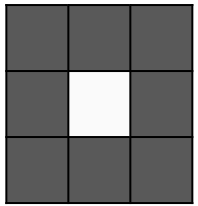
$$(x(t), y(t)),\ t = 1, \ldots, L$$

where L is the contour length, and each pair of points

$$(x(t), y(t)) \quad (x(t+1), y(t+1))$$

remains in some neighborhood relation. The variable t is the iterator in this sequence.





The type of a contour depends on the neighborhood type:

- N4 neighborhood: direct neighbors only.
- N8 neighborhood: direct neighbors and diagonal neighbors.

Diagonal neighbors = N8 \ N4.

[1] Satoshi Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985.

Perimeter

In continuous geometry:

$$L = \int \sqrt{x^2(t) + y^2(t)} dt$$

where t is a rolling variable (the iterator).

In discrete geometry, we approximate the perimeter by the length L of the contour.
Two basic variants:

1. Simply assume that L is the length (in pixels).
    a. The downside: will underestimate the contribution to length for diagonal neighbors.
2. Assume the following contributions to the length:
    a. For direct neighbors (N4): +1
    b. For diagonal neighbors (N8 \ N4): +√2

Area

In continuous geometry:

$$A = \int \int dxdy$$

In discrete geometry: the number of points belonging to an object.

$$A = |R| = \sum_{p \in R} 1$$

where R is the object (region) and p is the pixel.

- Note: in this case the area is expressed in *pixels*.
- Sometimes it is necessary to switch to *physical units*.

For a non-binary (monochromatic) image, we can consider interpreting brightness as the degree of object membership:

$$A = \sum_{p \in R} f(p)$$

where f(p) is the brightness of pixel p.
A is in this case a "fuzzy" (probabilistic) area.

Calculating area from the contour

Continuous variant (signed area):

$$A = \frac{1}{2}\left(\int y(t)\frac{dx(t)}{dt}dt - \int x(t)\frac{dy(t)}{dt}dt\right)$$

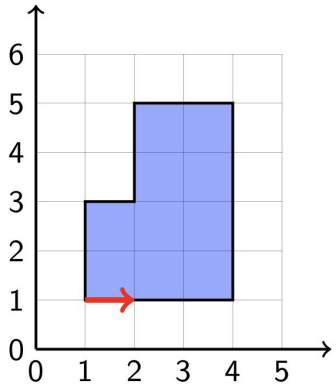
Discrete variant, signed:

$$A = \frac{1}{2}\sum_t \left(y(t)\Delta x(t) - x(t)\Delta y(t)\right)$$

Discrete variant, unsigned:

$$A = \frac{1}{2}\sum_t \left|y(t)\Delta x(t) - x(t)\Delta y(t)\right|$$

Notice:
We iterate over the boundaries between pixels,
rather than over pixels.



t	x	y	dx	dy	$ydx - xdy$
1	1	1			
2	2	1	1	0	1
3	3	1	1	0	1
4	4	1	1	0	1
5	4	2	0	1	-4
6	4	3	0	1	-4
7	4	4	0	1	-4
8	4	5	0	1	-4
9	3	5	-1	0	-5
10	2	5	-1	0	-5
11	2	4	0	-1	2
12	2	3	0	-1	2
13	1	3	-1	0	-3
14	1	2	0	-1	1
15	1	1	0	-1	1
					$\sum = -20$

Calculating area from the contour

Assuming that each displacement is parallel to the X or Y axis, at each step either Δx(t)=0 or Δy(t)=0, which allows the formula to be simplified to:

$$A = |\sum_t y(t)\Delta x(t)|$$

or complementary:

$$A = |\sum_t x(t)\Delta y(t)|$$

The formula from the previous also works for polygons.

- For example, polygons obtained by approximating the contour of an object with segments (see later sections of this module).

Local curvature

Local feature of the contour; based on second derivatives (for rasters: second-order differences/increments).

Continuous variant:

$$|\kappa(t)|^2 = \left(\frac{d^2x}{dt^2}\right)^2 + \left(\frac{d^2y}{dt^2}\right)^2$$

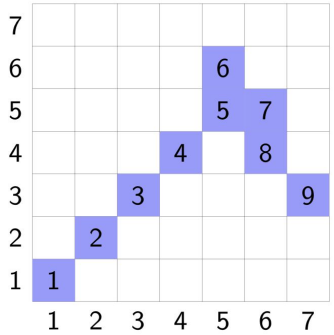
Discrete variant:

$$|\kappa(t)|^2 = \left(\Delta^2 x\right)^2 + \left(\Delta^2 y\right)^2$$

Applications:

- determining locations of corners,
- testing of object symmetry,

When is the curvature maximum?



t	x	y	dx	dy	d^2x	d^2y	$\kappa(t)^2$
1	1	1					
2	2	2	1	1			
3	3	3	1	1	0	0	0
4	4	4	1	1	0	0	0
5	5	5	1	1	0	0	0
6	5	6	0	1	-1	0	1
7	6	5	1	-1	1	-2	5
8	6	4	0	-1	-1	0	1
9	7	3	1	-1	1	0	1

Total curvature (of a contour)

Global aggregate of curvature: so-called *deflection energy*, *bending energy*.

$$\sum_t |\kappa(t)|^2$$

Summed or averaged over contour points.

General remark:
Area, perimeter, curvature, and other geometric features can be successfully applied not only to contours, but also to more 'coarse' representations of object shape, e.g., contour approximations:

- polygonal approximation,
- splines.

Signature

Question: We have many mathematical concepts and tools for analyzing functions of one variable. Can we convert the contour of an object (i.e., a pair of functions (x(t), y(t)) into such a function while retaining all of the information about the contour?

Answer: Yes; there are many possibilities; one of them is a *signature*.

Definition: a signature is the distance of a contour point from some reference point (i.e., the length of the trailing radius), expressed as a function of some independent variable, such as angle or the rolling variable t.

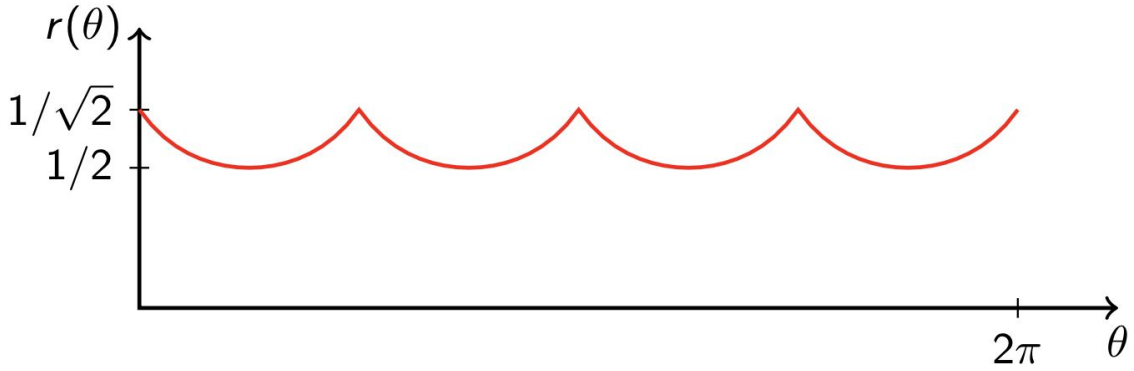
Typical reference point: the centroid of an object (center of gravity).

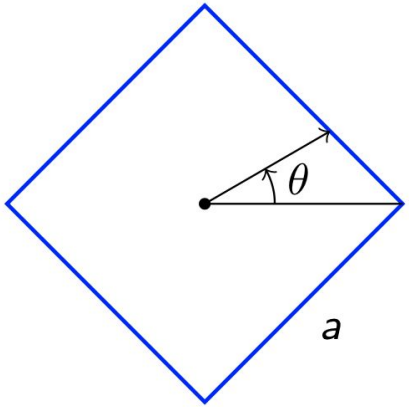
Properties: Invariance w.r.t. T, but not of R and S.

Signature: example

Independent variable: the angle between the ray and the positive direction of the X axis.

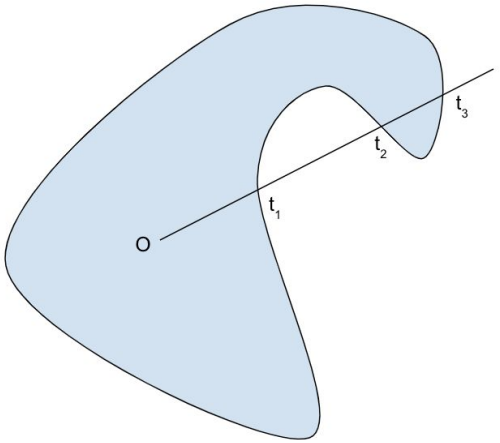
Dependent variable: the Euclidean distance of the contour point from the reference point (which in this example is the geometric center of the object).





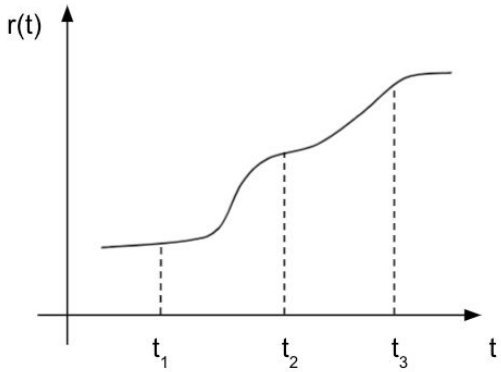
Signature dependent on iterator (running variable)

A signature defined with respect to an angle cannot be applied in general to nonconvex shapes. See the example on the right: the trailing arm intersects the object contour at many points; the signature is ambiguous.



Remedy: signature with respect to a running variable (point number, 1,...,L):

- The distance from point O is unambiguous for successive points t1, t2, t3, visited in the process of creating the signature.



Signature: invariance

The signature ensures the invariance w.r.t. T, but not of R and S.

Ensuring invariance of R: standardizing the choice of a starting point, e.g.

- the point furthest from the centroid,
- the point on the principal axis that is most distant from the centroid.

Ensuring S-invariance:

- Normalization (min-max scaling) or (better) standardization* of the signature.

*Meaning: subtracting the mean value of r and dividing by the standard deviation of r.

Skeleton

Also known as *thinning* or *skeletonization*.

A **skeleton point** in a set of points (area, region) R is a point that has more than one nearest neighbor on the edge B of R.

The **skeleton** of a set R (medial axis, [Blum 1967]) is the set of all skeleton points of R.

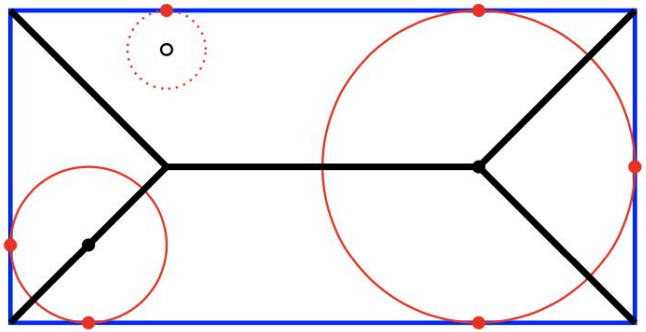
Property: skeleton points are points that are the centers of maximal circles contained in R (i.e., there is no circle with the same center and larger radius contained in R).

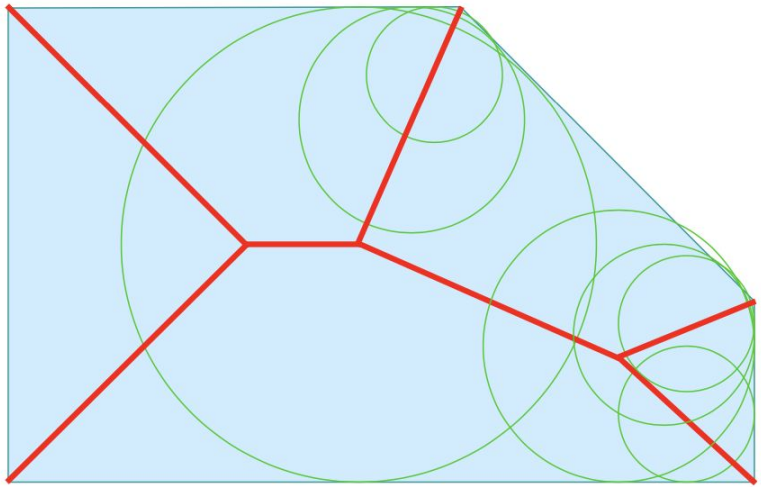
Blum H. A transformation for extracting new descriptors of shape. W. Wathen-Dunn (Ed.), Models for the Perception of Speech and Visual Form, M.I.T. Press, New York (1967), pp. 362-380

Skeleton: Examples

The result: a "graph", or more precisely: a branching curve.

- Right: the figure in blue, the skeleton in black.
- Below: the figure in blue, the skeleton in red.

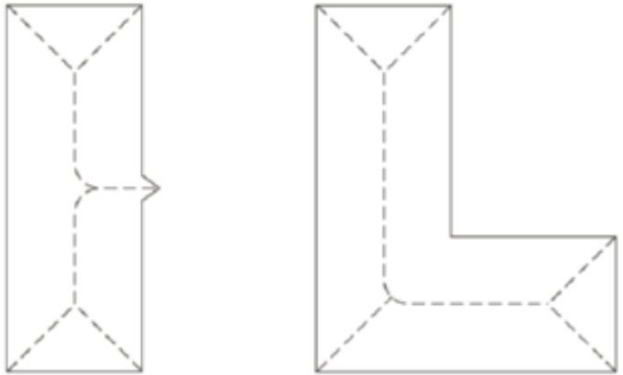




Skeleton: Examples

Characteristics:

- Susceptibility to small edge distortions:
  - Theorem: If the point P on the edge B of the set/object/region R is the point where the curvature of the edge has a local maximum, then there exists a branch of the skeleton that ends at P.

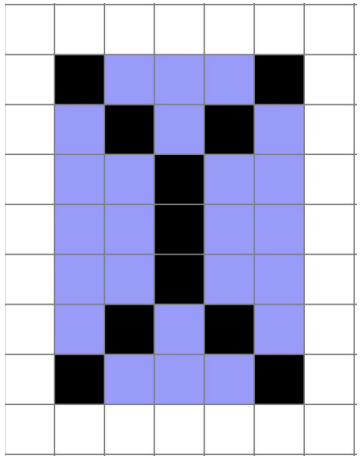


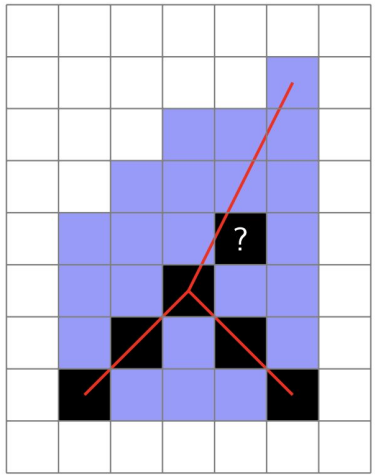
Implication: we use skeletons mainly for analyzing thin objects with smooth contours.

Defining skeletons on rasters

Requires generalizing the notion of equality of distance.

- Left: There is exact distance equality for the skeleton pixels, so it can be unambiguously determined that they are skeleton points.
- Right: discrete distance values do not allow to define the skeletons in an unambiguous way.





Defining skeletons on rasters

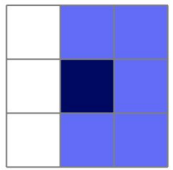
Practical skeletonization algorithms work by "selective erosion," i.e., iteratively removing the contour points that are not skeleton points.

- Iterations are repeated until the next step makes no changes (all points are both contour and skeleton at the same time).

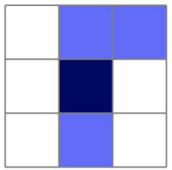
How to check if a point belongs to a skeleton branch?

A common approach: counting background-object transitions:

```
1: function N8Trans(p)
2:     n ← 0
3:     for i ← 0 . . . 7 do
4:         if N8(p,i)=0 ∧ N8(p,(i + 1) mod 8)=1 then
5:             n ← n + 1
6:         end if
7:     end for
8:     return n
9: end function
```



N8Trans=1



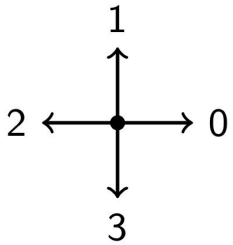
N8Trans=2

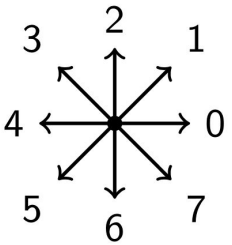
Shape approximation

Chain codes (absolute)

A chain code represents the contour of an object by a sequence of vectors – usually unit steps in an N4 or N8 neighborhood.

- Right: 4-way and 8-way codes, along with examples of contours.

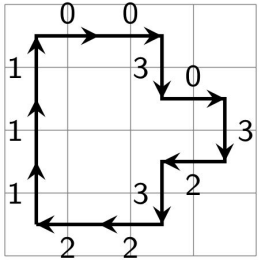




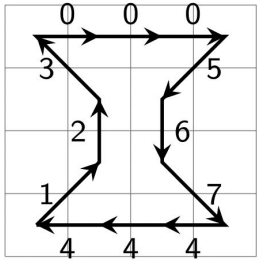
- Allows for very memory-efficient storage of information about the shape of a of the contour.
- Invariant w.r.t. translations.

Often used in combination with run-length encoding:

- Storing the number of repeating elements. For the shapes on the right:
  - $0_2303232_21_3$
  - 0_35674_3123



003032322111

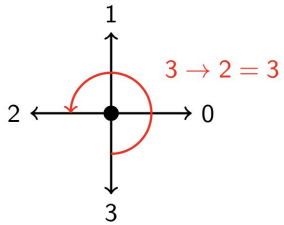


000567444123

Relative chain codes

Motivation: provide independence from absolute directions.
Idea: The next code (for pixel t+1) is determined relative to the current direction (determined by the transition from point t-1 to point t).

- Example (on the right):
  - previous direction: 3
  - current direction: 2 (turning 90° right)
  - relative chain code: 3
    - (because that's the absolute chain code for 'right' assuming 0 being the current direction)



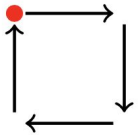
Provides invariance w.r.t. rotations by multiples of 90°.
Another formalism often used in combination with chain codes (not discussed further in detail): Huffman encoding: frequent directions (e.g., "straight ahead") encoded with short bit sequences; less frequent ones with longer ones.

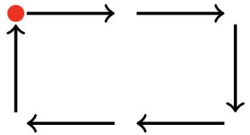
Shape number: feature based on chain codes

Calculated as follows:

1. Calculate the chain code
2. Convert to a relative chain code (differential)
3. "Normalize" it by rotating the digits (modulo) in such a way that the number represented by the codes is the smallest.

In other words: the first difference (derivative) with the smallest magnitude.





Chain code:	0321	003221
Diff:	3333	303303
Shape number:	3333	033033

Polygonal approximations

Polygonal approximation

Motivations:

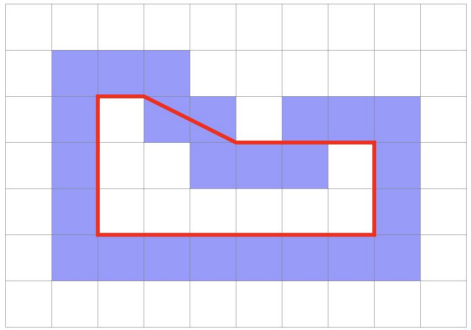
- obtaining an approximation of the shape of the object,
- while eliminating non-essential contour elements and
- forcing the shape of an object into a certain class of geometric objects (polygons).

The methods discussed in the following slides differ in

1. the approximation method, and
2. the criteria for evaluating approximation.

Minimum perimeter polygon

Imagine that the contour pixels form a "channel" carved into a block of some material. Take a rubber band and insert it into this channel; the rubber band will take the shape of a polygon with a minimum perimeter.

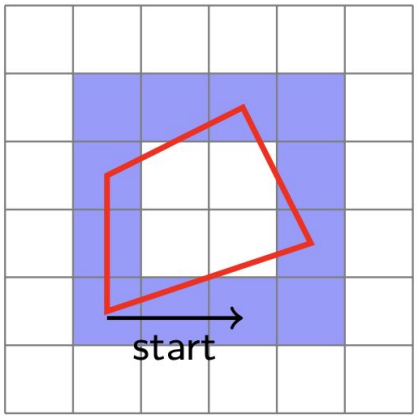


Properties:

- well-defined upper bound on approximation error ($\sqrt{2}$ per pixel),
- sensitive to noise and distortions,
- coordinates of points describing the polygonal chain do not correspond to raster nodes (centers of pixels).

Merging algorithm

1. Iterate over contour pixels, in each iteration creating a segment that connects the current point with the starting point (‘running segment’).
   a. … until the [mean square] error of approximating the contour points with the segment exceeds a certain threshold.
2. Add the resulting segment to the broken line.
3. If the starting point is not reached, go to 1.
4. Analyze pairs of consecutive lines, finding their points of intersection. The resulting segments constitute an approximating polygon.



Disadvantages:

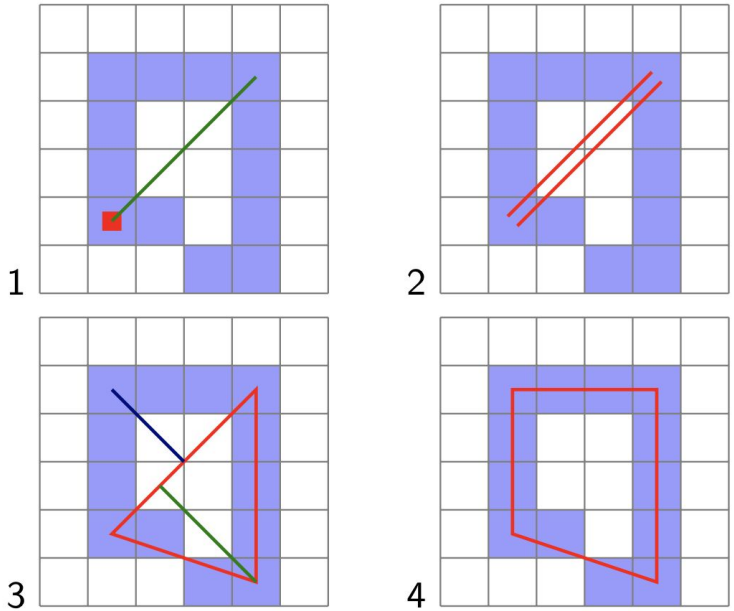
- Delay in making the decision in step 1a.
- Decisions made strictly locally (for a given segment).

Advantages:

- Very fast, single-pass.

Divide-and-conquer algorithm

1. Start with a degenerate polygonal chain (a single contour point).
2. Find the contour point furthest from the polygonal chain and add it to the chain.
3. If the stopping conditions are not satisfied, go to point 2.



Typical stop conditions:

- Number of points/segments in the chain.
- Quality of approximation (mean/maximum distance of contour points from the chain)

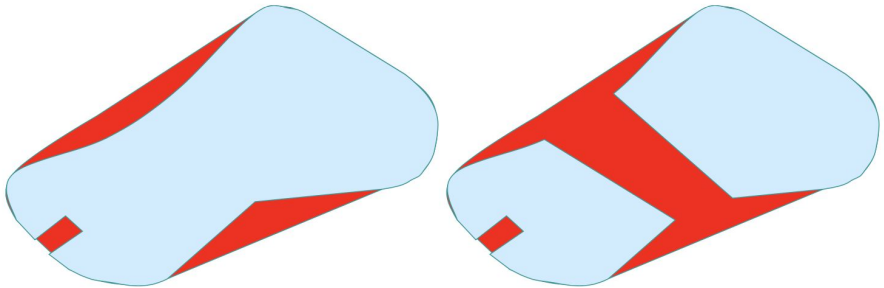
Disadvantages: multi-pass (however, significant savings can be achieved by storing data from previous passes).

Advantages: better reflects the shape of the object at the global level.

Convex hull

Convex hull H of a set of points S: the smallest convex set of points containing S.

The set H \ S is the so-called **convex deficiency**.



Left: an object (blue) and its convex deficiency (red).
Right: an set composed of two objects and its convex deficiency.

Convex hull

Features that can be extracted from a convex hull:

- areas of H, H \ S,
- number of connected components in H \ S
- spatial distribution of connected components in H-S

Classes of algorithms for calculating the convex hull:

1. Using controlled polygon approximation.
2. Using morphological operations; controlled dilation (drawback: produces only coarse approximation of the hull).

Other applications:

- Can be used for edge segmentation:
  - Edge segmentation points: points where the contours of H and S diverge and rejoin.

Fourier Descriptors

A sequence of contour points

$$((x_0, y_0), (x_1, y_1), (x_2, y_2), \ldots, (x_{N-1}, y_{N-1}))$$

can be represented as a complex time series:

$$s(k) = x(k) + jy(k)$$

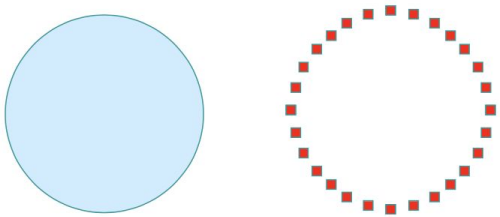
and subject to a discrete (one-dimensional) Fourier transform:

$$a(u) = \frac{1}{N} \sum_{k=0}^{N-1} s(k) \exp\left(-\frac{j2\pi k}{N}\right) \qquad u = 0, \ldots, N-1$$

The results: N coefficients (complex) characterizing the 'torsion'/curvature of the contour in each frequency band.

Fourier Descriptors

The object and its contour:



Power spectrum of the contour:

Characteristics:

- the overall shape of the object determines the low frequencies,
- the fine fluctuations of the contour ('fringes') determine the high frequencies.

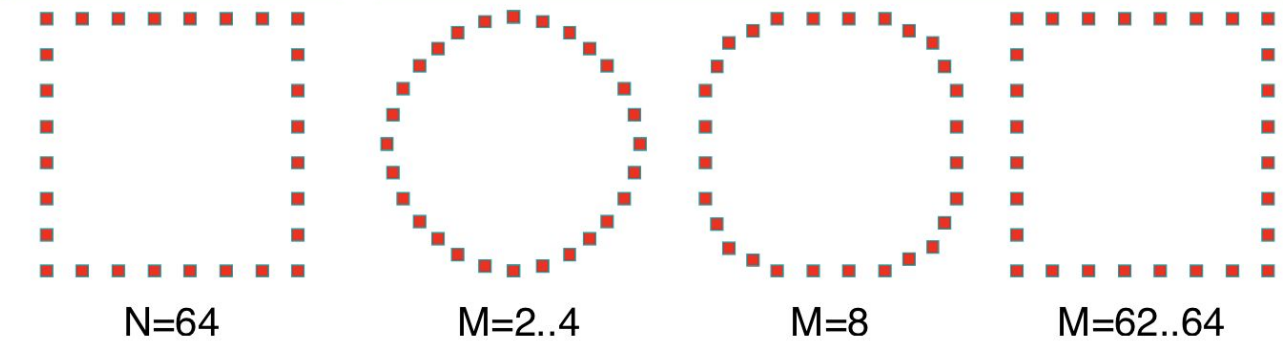
Question: What determines a(0)?

Fourier Descriptors for contour processing

Other applications: contour processing (e.g., smoothing).

- Essentially: smoothing in the frequency domain.

Example: a contour comprising N=64 points (left) and its reconstructions (right) obtained using the inverse Fourier transform applied to the spectrum band-limited ('clipped') to the M lowest frequencies.



Fourier Descriptors

Disadvantages:

- Dependence of spectrum size on contour length (nevertheless, in practice we are mostly interested in low frequencies anyway).

Advantages:

- Although FDs do not offer TSR invariance, the transforms translate into predictable changes in the spectrum, according to the properties of the Fourier transform:

Transform	Contour	FD
Identity	$s(k)$	$a(u)$
Rotation by angle θ	$s(k)e^{j\theta}$	$a(u)e^{j\theta}$
Translation by Δ_{xy}	$s(k) + \Delta_{xy}$	$a(u) + \Delta_{xy}\delta(u)$
Scaling	$\alpha s(k)$	$\alpha a(u)$
Changing the starting point	$s(k - k_0)$	$a(u)e^{-j2\pi k_0 u/N}$

Moments

Motivations

- The concept of moments (used commonly calculus and statistics) gives us the ability to characterize a function by certain ‘metrics’.
  - Moments are well-known and have been studied in-depth for centuries.
- A two-dimensional raster image is a function f(x,y) of two variables.
  - Using moments to characterize images is therefore natural.

In the context of computer vision, moments are quite often referred to as **geometrical moments**.

Moments

One-dimensional moment of order p of function f (assumption: f is continuous)

$$m_p = \int_{-\infty}^{+\infty} x^p f(x) dx$$

Two-dimensional moment of order p+q of function/image f

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

Discrete two-dimensional moment of order p+q

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

Disadvantage: no TSR invariance

Central moments

Central two-dimensional moment of order p+q

$$\mu_{pq} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} (x - \bar{x})^p (y - \bar{y})^q f(x, y) dx dy$$

where the coordinates of the center of gravity can be calculated using moments:

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

Question: what is the interpretation of m_{00}?

Central moments provide the invariance w.r.t. translation.

Central normalized moments

Central normalized two-dimensional moment of order p+q

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

where

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

for p+q > 1.

Provides invariance w.r.t. T and S.

Hu moments

$$\phi_1 = \eta_{20} + \eta_{02}$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

$$\phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

$$\phi_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$$

$$\phi_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2].$$

Provide the TSR invariance.

Since moments depend on each other, they are often calculated 'in bulk'.

- See, for example, the HuMoments() function in OpenCV.

Moments: A theoretical note

Uniqueness theorem [Papoulis 1965].

For a function f(x,y) continuous and taking nonzero values only in a finite part of the XY plane:

- all moments exist (for any $p,q \geq 0$),
- the series of moments m_{pq} is uniquely determined by f(x,y),
- and vice versa: f(x,y) is uniquely determined by the series of moments m_{pq}.

Therefore: Geometric moments (all taken together) define an **image transform.**

- However, this observation is rarely used in practice
  - In image analysis, we use moments almost exclusively as features of images/objects.

Moments: applications

- Applicable to both monochrome and binary images.
- Applicable to functions of any number of variables.
- Can be used to describe any features, not necessarily rasters.
  - For example contours, signatures, etc.
- Have strong connections with quantities used in statistics (mean, variance, kurtosis, covariance, etc.).

Caveat: Like for many other feature descriptors, the S (scale) and R (rotational) invariance is only approximate, i.e. conditioned on the quality of interpolation incurred by those transformations.

Shape coefficients

Shape coefficients

Quantities/algorithms designed with the intent of reflecting a specific feature of an object, usually such that has a natural perceptual and/or linguistic interpretation, e.g.:

- circularity of an object,
- the elongation of an object,
- degree of hollowness of an object,
- and more (BYOSC – Bring Your Own Shape Coefficient 😄)

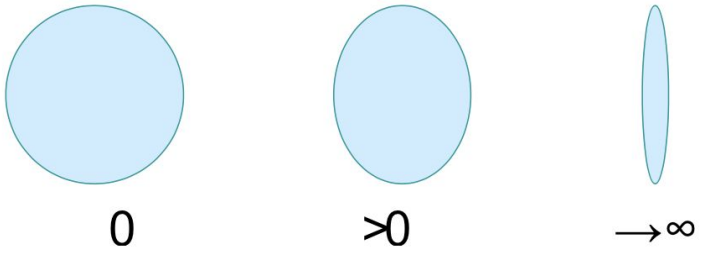
Notes:

- can be normalized (bounded) or unnormalized (unbounded),
- for small objects, quite prone to errors resulting from interpolation/spatial discretization.

Shape coefficients

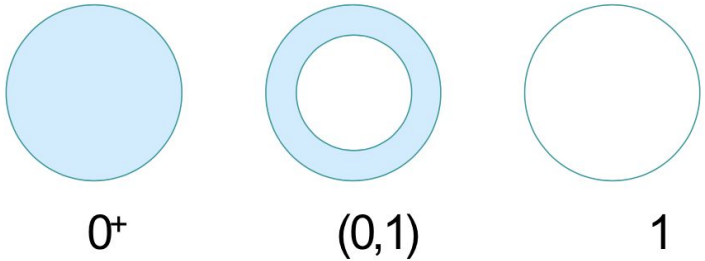
Circularity/roundness coefficient

$$\frac{L}{2\sqrt{\pi S}} - 1$$



‘Hollowness’ (Blair-Bliss coefficient)

$$\frac{S}{\sqrt{2\pi \iint_S r^2 ds}}$$



- L: circumference (perimeter),
- S: area,
- ds: an infinitesimally small element of the object (in practice: a pixel),
- r: length of the segment/ray connecting the center of the object with the pixel.

Shape coefficients

Danielson: $$\frac{S^3}{\left(\iint_S l ds\right)^2}$$

Haralick: $$\sqrt{\frac{\left(\sum d\right)^2}{n\sum d^2 - 1}}$$

Other: $$\frac{L_{max}}{L} \qquad \frac{r_{min}}{R_{max}}$$

Where:

- n: number of contour points,
- d: distance of the contour point from the object's center of gravity
- l: minimum distance of the object from the contour
- r_{min}: minimum distance of the contour from its center of gravity
- R_{max}: maximum distance of the contour from its center of gravity
- L_{max}: maximum extent of object (*major axis*); *minor axis*: perpendicular to the major axis and such that the resulting rectangle includes the shape.

Texture analysis

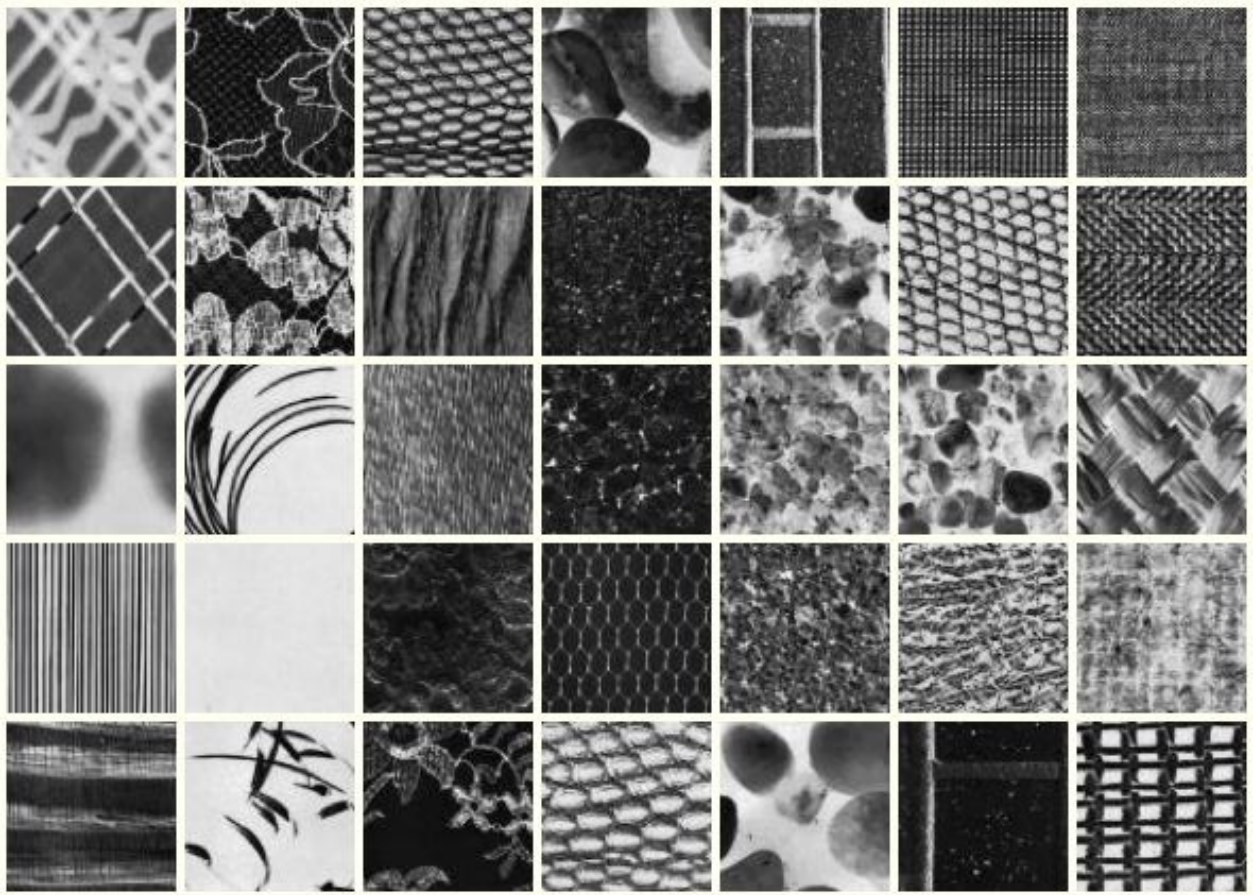
The concept of texture

Definition 1: A cyclic (in the spatial sense), but often imprecise, repetition of similar image elements called *texems*.

Definition 2: A combination (superposition) of a relatively large number of objects, where the characteristics of the individual objects are unimportant, but the structural properties of the entire set of objects are important.

A texture may reflect certain physical characteristics of the objects being imaged, e.g., smoothness/roughness, directionality, being fabric-like, etc.

Examples of textures:



P. Brodatz, "Textures: A Photographic Album for Artists and Designers", Dover Publications, New York, 1966.
See e.g. http://sipi.usc.edu/database/database.php?volume=textures

Texture analysis methods

Selected texture analysis techniques:

- structural:
  - including syntactic,
- statistical:
  - autocorrelation,
  - histogram analysis,
  - co-occurrence matrix
- spectral methods,
- filter banks,
- edge density,
- …

Structural methods

An attempt to reconstruct/generate a texture using some algorithm, or grammar (in syntactic approaches).

- Texems correspond to terminal symbols of the grammar,
- Grammar productions define possible relationships between the texems.

Advantages:

- strong and elegant formalism,

Disadvantages:

- problems with the description of real textures (typically, the periodicity/repetition of texems is only approximate)
  - This observation motivates turning to statistical methods.

Autocorrelation

$$C(p,q) = \frac{MN}{(M-p)(N-q)} \frac{\sum_i \sum_j f(i,j) f(i+p,j+q)}{\sum_i \sum_j f^2(i,j)}$$

where:

- f: the image (we assume that the mean value has already been subtracted, i.e. mean = 0)
- M,N: window size (width, height)
- p,q: considered periods of texture periodicity in each dimension.

C(p,q) gives the intensity of occurrence of a texture with a **period** determined by the length of the vector (p,q) and the **orientation** determined by p and q.

Provides invariance T; no invariance w.r.t. S, R.
Disadvantage: does not reflect texture characteristics at specific brightness levels.

Co-occurrence matrix

Also known as *gray-level cooccurrence matrix* (GLCM)

Let P be a *position operator* (the relation in which two image points can be located); typically expressed as an *offset vector* or simply neighbour number.

Let us define the matrix A of size kxk, where k is the number of brightness levels, whose elements for image f are defined as follows:

$$a_{ij} = |\{p : f(p) = i \wedge P(p,r) \wedge f(r) = j\}|$$

where p and r are image points, and P(p,r) means that points p and r are in the spatial relation P.

Co-occurrence matrix: Example

Definition of relation P: neighbor #7 in the N8 neighborhood:

$$P(p,r) \iff x_r = x_p + 1 \wedge y_r = y_p + 1$$

p	
	r

The input image f:

0	0	1	1	2
1	1	0	1	1
2	2	1	0	0
1	1	0	2	0
0	0	1	0	1

Co-occurrence matrix (raw, not normalized yet):

- rows: brightness of pixel p
- columns: brightness of pixel r

i \ j	*0*	*1*	*2*
0	4	1	0
1	2	4	2
2	1	2	0

Co-occurrence matrix

The co-ocurrence matrix C is the normalized matrix A (elements divided by the number of pixel pairs sampled by P, or simply by the sum of elements of A):

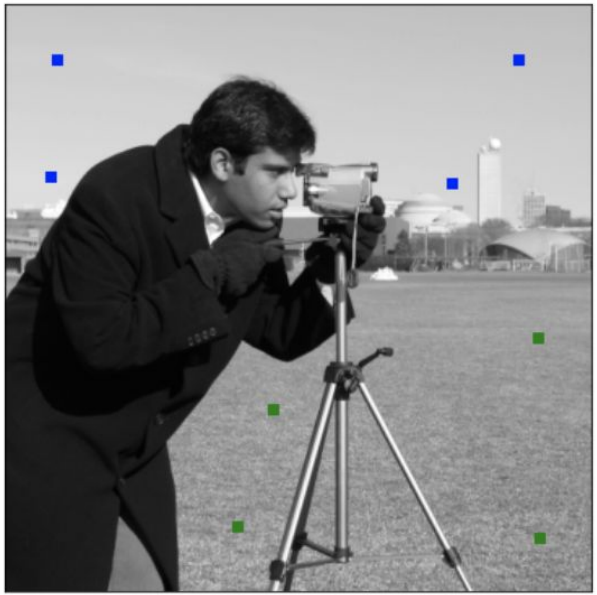
i \ j	*0*	*1*	*2*
0	0.25	0.0625	0
1	0.125	0.25	0.125
2	0.0625	0.125	0

Interpretation: an estimate of the joint probability distribution of sampling a pair of points that are in relation P *and* have specific brightness values.

- The co-occurrence matrix is a *second-order histogram* (or *conditional histogram*).
- The matrix can be a feature in its own right, or be the basis for computing more concise features (e.g., moments).

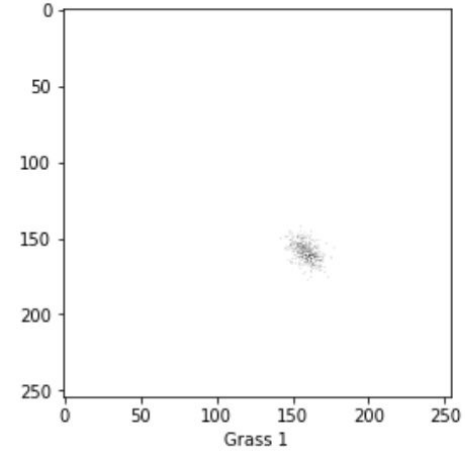
Example

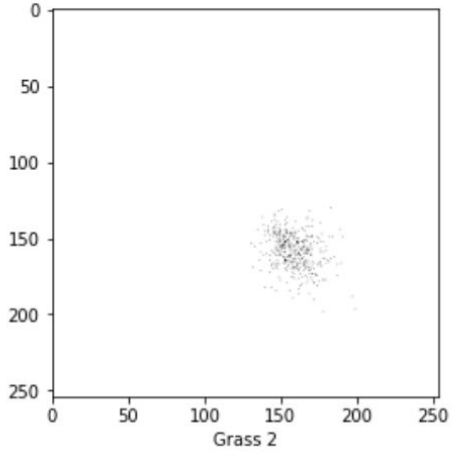
Input image and 8 locations for which the co-occurrence matrices will be determined:

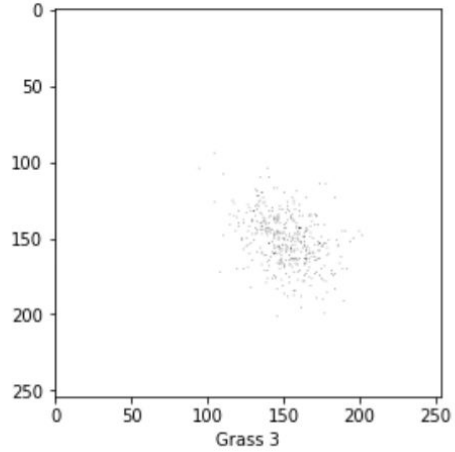


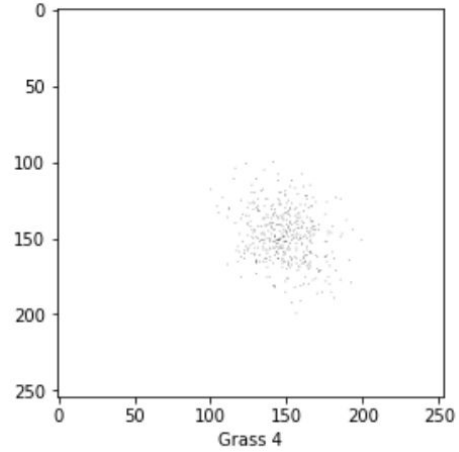
Based on: https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_glcm.html

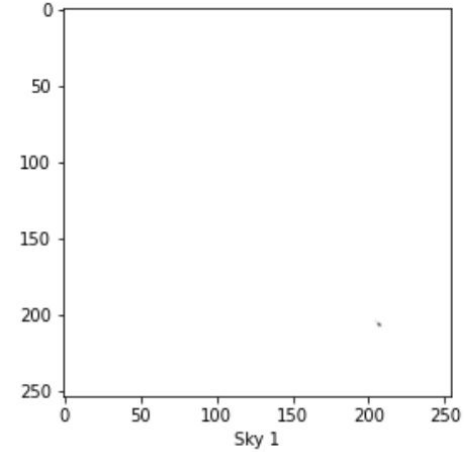
Co-occurrence matrices for particular locations (4x grass, 4x sky)

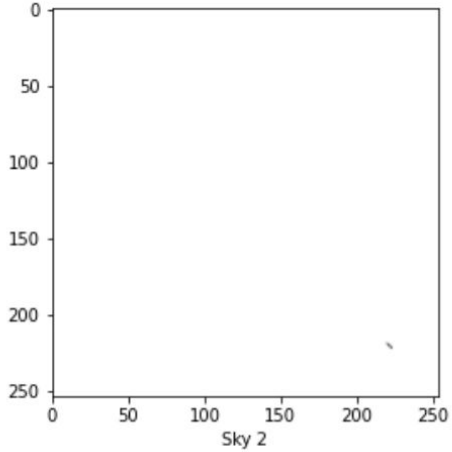


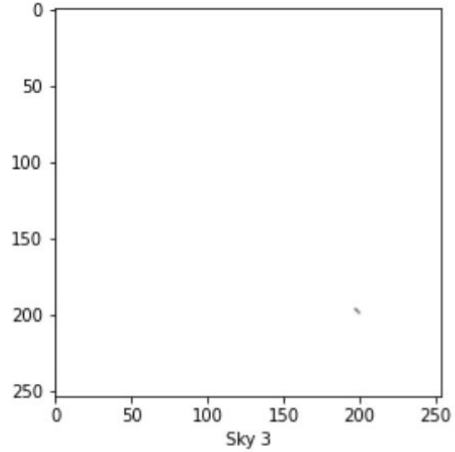


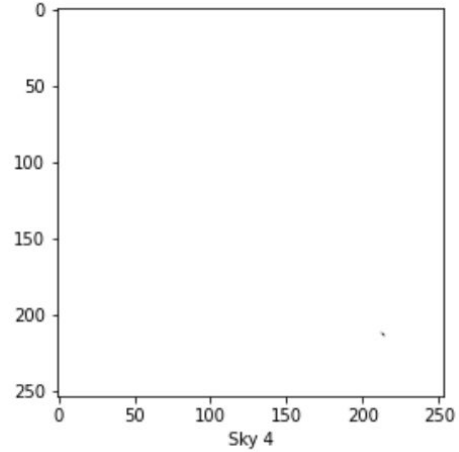






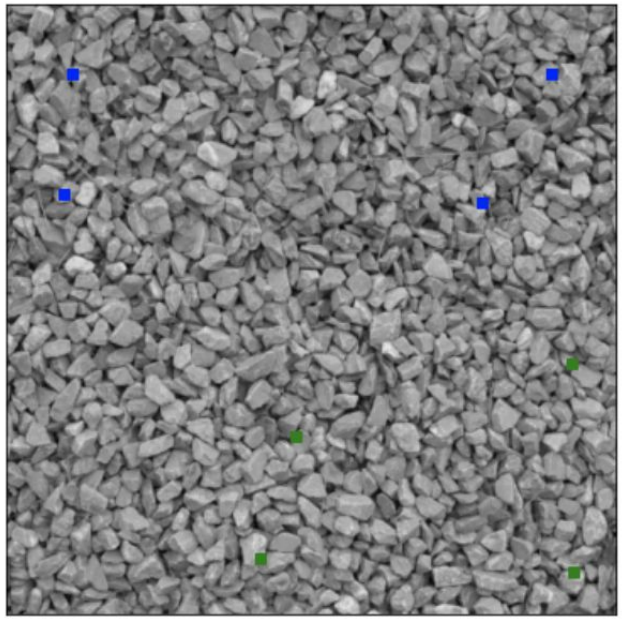




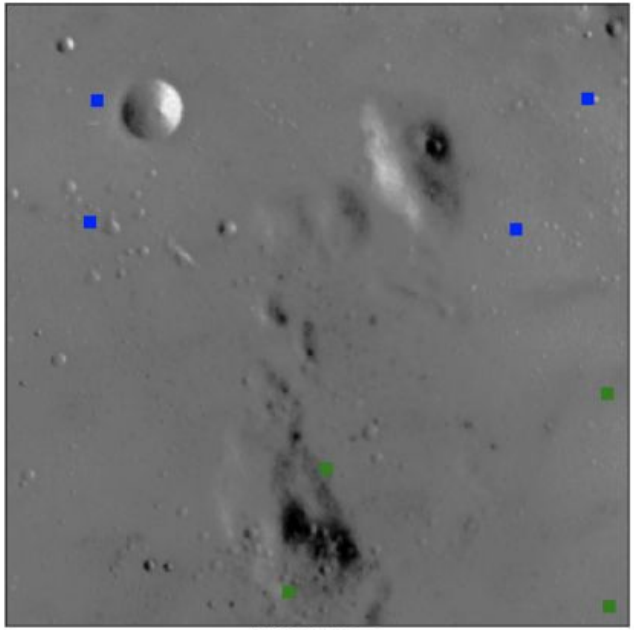


Other examples

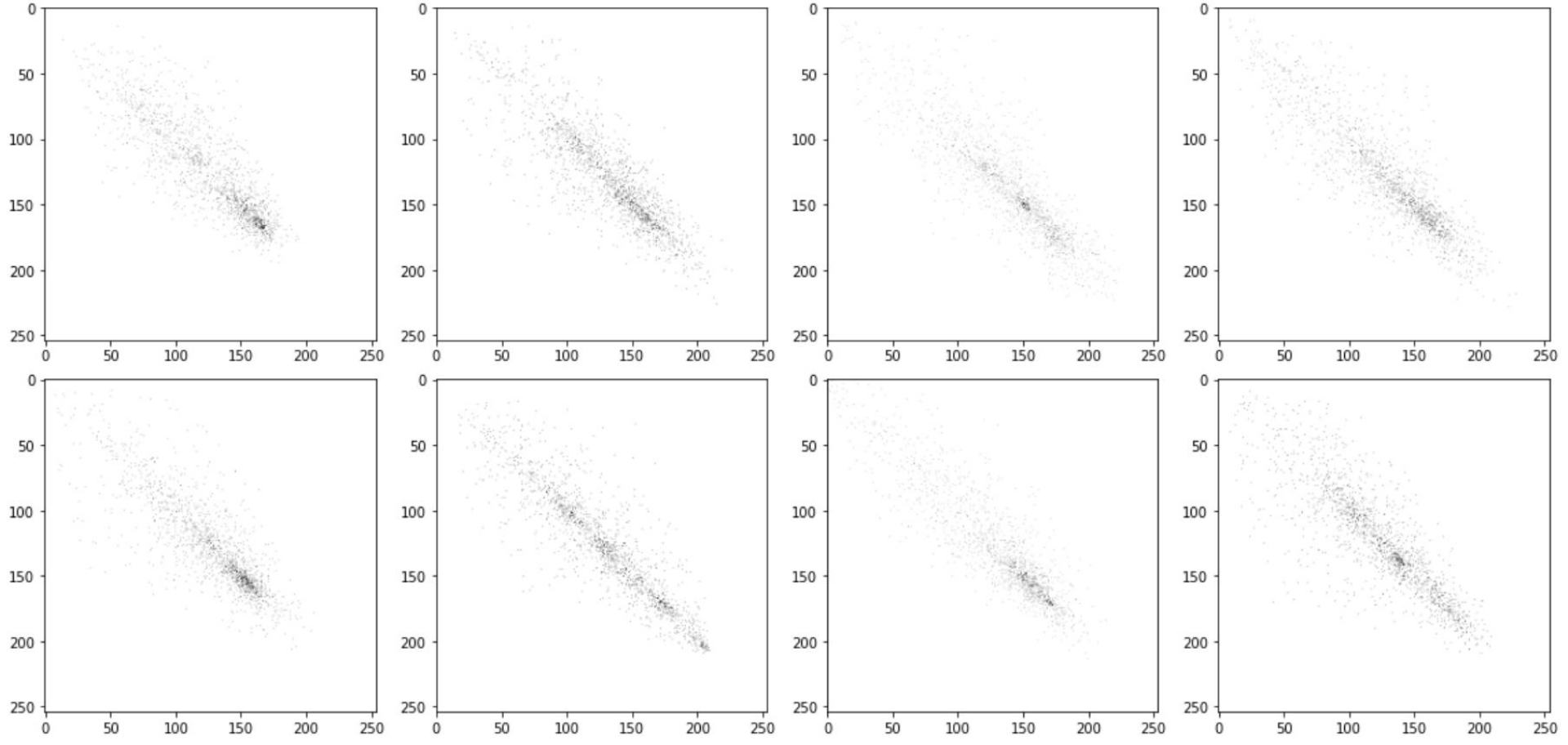
Gravel, PATCH_SIZE=41



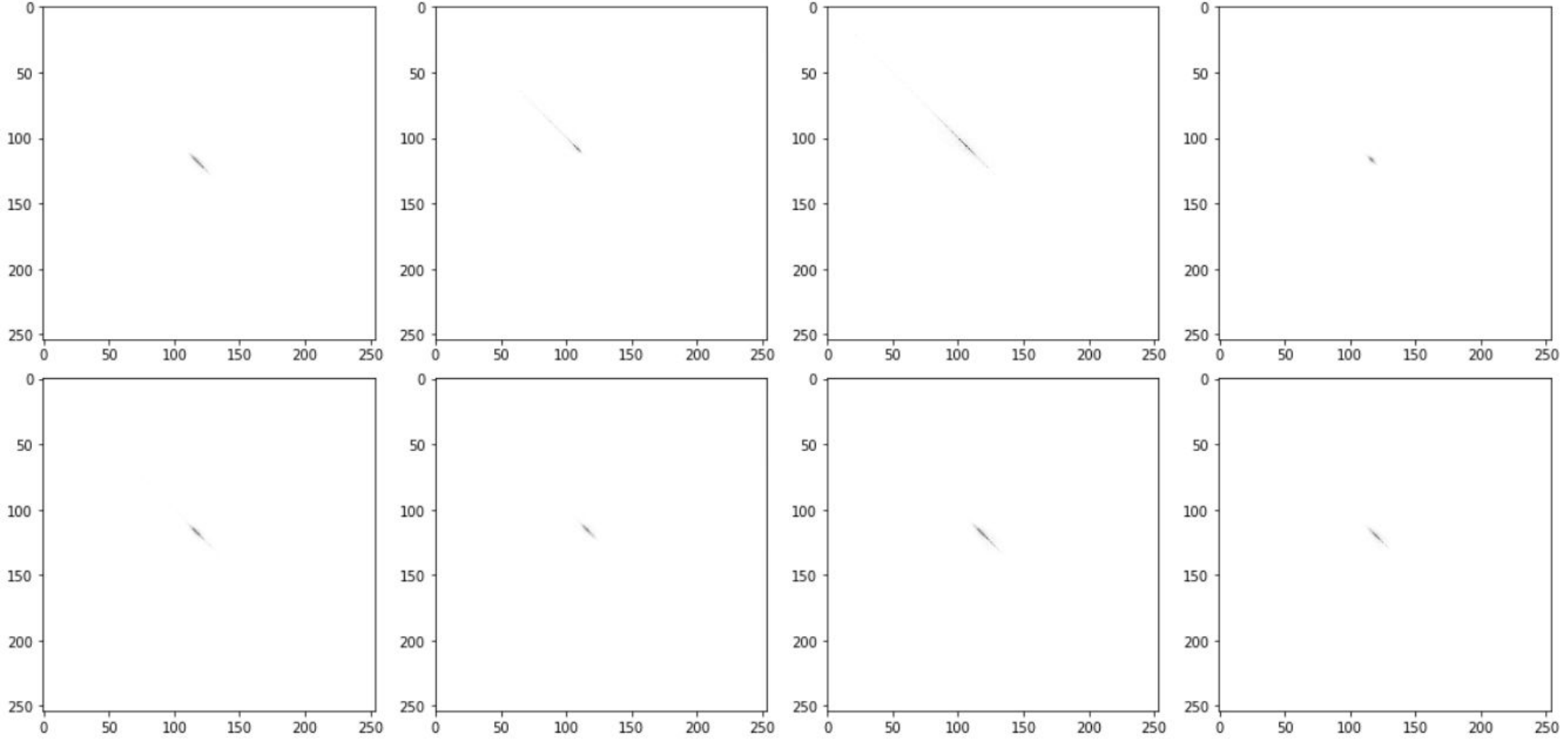
Moon, PATCH_SIZE=81



Gravel



Moon



Co-occurrence matrix: Remarks

Popular preprocessing for co-occurrence matrices: Discretization of brightness levels, for brevity and representativeness.

Texture features can be computed from co-occurrence matrices [Haralick 1979], e.g., difference moment of order k, inverse difference moment of order k, entropy (a measure of texture randomness), uniformity, and others.

$$\sum_i \sum_j c_{ij}(i-j)^k$$

Possible generalizations:

- Relations P that are invariant w.r.t. rotation
  - E.g., P defines a neighbor at a certain *distance* (disregard for orientation).
- Using multiple P relations simultaneously:
  - Results in multiple co-occurrence matrices of the same size, which can be stack into a tensor.
- Applying the CMs to multiscale representation of images.

Texture analysis: Other approaches

- Julesz features (capturing texture as a Markov process)
- Gabor filters: filters in the form of a two-dimensional/complex sinusoid modulated by a two-dimensional Gaussian distribution
  - Convolving the image with multiple differently parameterized Gabor filters captures the texture characteristics at different levels of detail.
- Wavelet transform
- Singular value decomposition (SVD)

Generalizations aimed at other types of images:

- color and multimodal images
- 3D images

Applications of texture analysis

- Description of the entire image
- Description of regions and objects
- Image segmentation

Texture analysis for depth perception

- In 3D scene analysis, textures undergo similar distortions as the objects themselves (projection, perspective, etc.); see examples below.
- These distortions convey additional information about the scene.
- Hence, many research works and approaches follow the "X from texture" scheme, e.g.
  - shape from texture,
  - depth from texture.

Hough Transform

Hough Transform

- A method for finding straight lines in an image by gradual accumulation of evidence.
- Input: Binary image representing contour points.
- Output: Equations of the most prominent lines (segments in the input image).

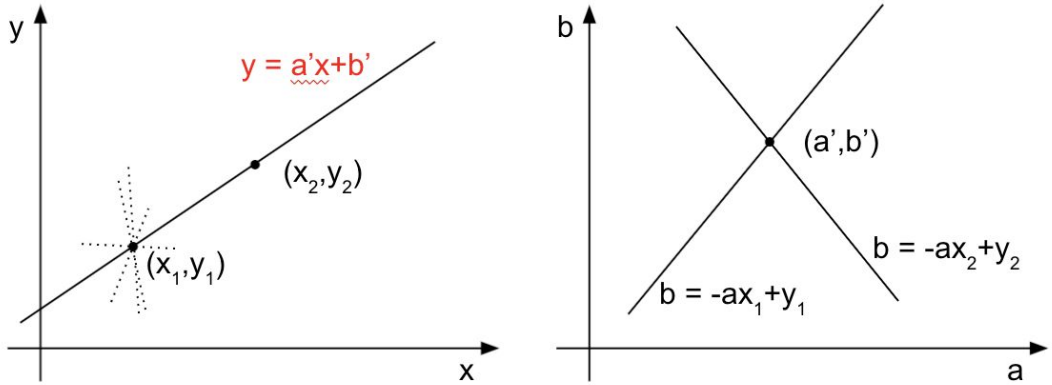
The essence of the method: transition from the **primal coordinate space** (x,y) to the **dual parameter space of lines** (a,b).

- Line equation $y = ax + b$
- Equation of the concurrent lines passing through (x,y) $b = y - xa$
- Observation: a contour point with coordinates (x,y) increases the probability of existence of lines with parameters (a,b) satisfying the equation of the concurrent lines.

Hough Transform

Original space: X × Y image

Dual space: A × B



- A point in the primal space corresponds to a line in the dual space.
- A point in the dual space corresponds to a line in the primal space.
- Co-linear points in the primal space correspond to lines intersecting at a single point (a',b') in dual space.

Hough Transform: The algorithm

Idea:

- Discretization of the dual space: an array of *accumulators* (initially zeros).
- Each considered point (x,y) in the original space increments the evidence in the accumulator cells located (approximately) on the straight line corresponding to (x,y) in the dual space.

Example: (x,y) = (1,2)
Equation of the dual line: b = 2 - 1a = -a + 2

Each considered point causes analogous increments.

Advantage: very simple schema of updating the array of accumulators.

b \ a	-3	-2	-1	0	1	2	3
3			+1				
2				+1			
1					+1		
0						+1	
-1							+1
-2							
-3							

Hough Transform: The algorithm

```
function hough(f)
      acc = 0           // initialize the accumulator array
      for x=1 to xMax   // loop over the image points
       for y=1 to yMax
        if f(x,y) > 0   // if the point is on (lit)
         for a = aMin to aMax   // iterate along a line in the dual space
          b = y - xa
          if bMin < b < bMax
            acc[a,b] += 1       // increment the element of the accumulator array
      return acc
```

Result: the accumulator array.

- An array element acc[a,b] that contains a particularly large values signals the presence of the straight line y = ax + b.

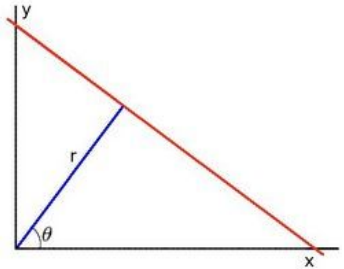
Hough Transform in polar coordinates

The constant step of a is inconvenient – 'steep' angles are more densely sampled.

Solution: the polar coordinate system, in which the line equation is

$$r = x\cos\theta + y\sin\theta$$

where: r – length of the normal vector, Θ – the angle of the normal vector with respect to the positive direction of the X axis.

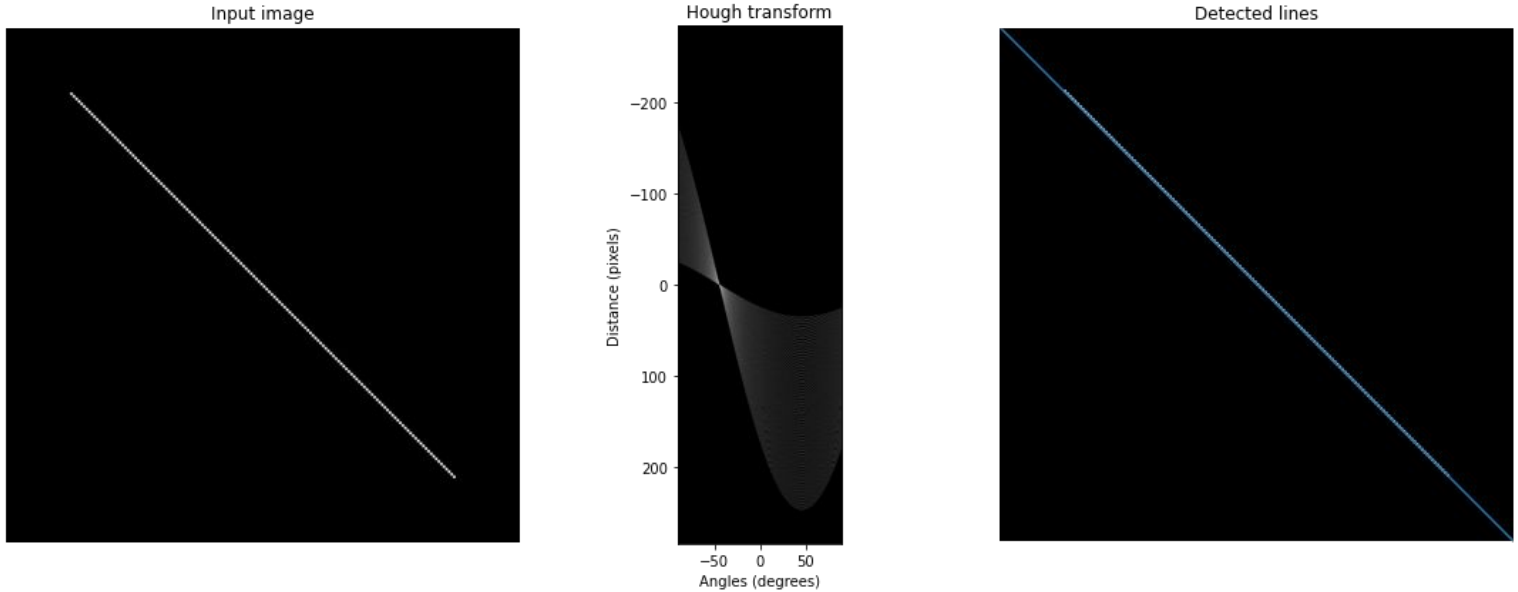


As a result, the dual (accumulator) space spans the coordinates (Θ,r).

- The values of Θ are discretized uniformly in the entire range [0, 180°).
- Addresses the problem of vertical lines.

Illustration for a single line

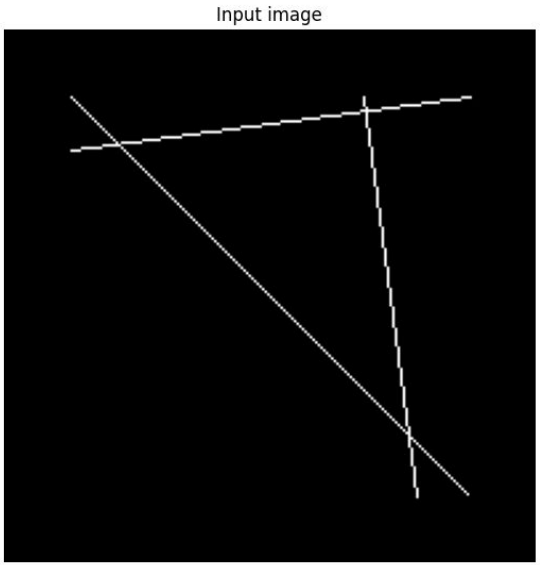
The accumulator array can be rendered as an image (in the middle).

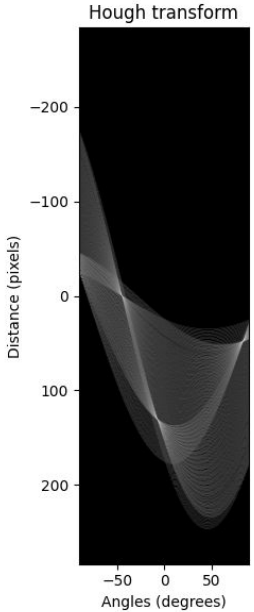


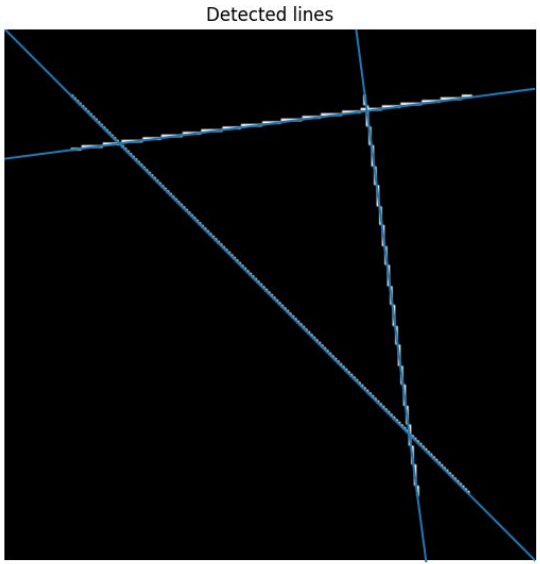
https://scikit-image.org/docs/dev/auto_examples/edges/plot_line_hough_transform.html

Illustration

- Three local maxima in the accumulator array determine three straight lines of the highest evidence.

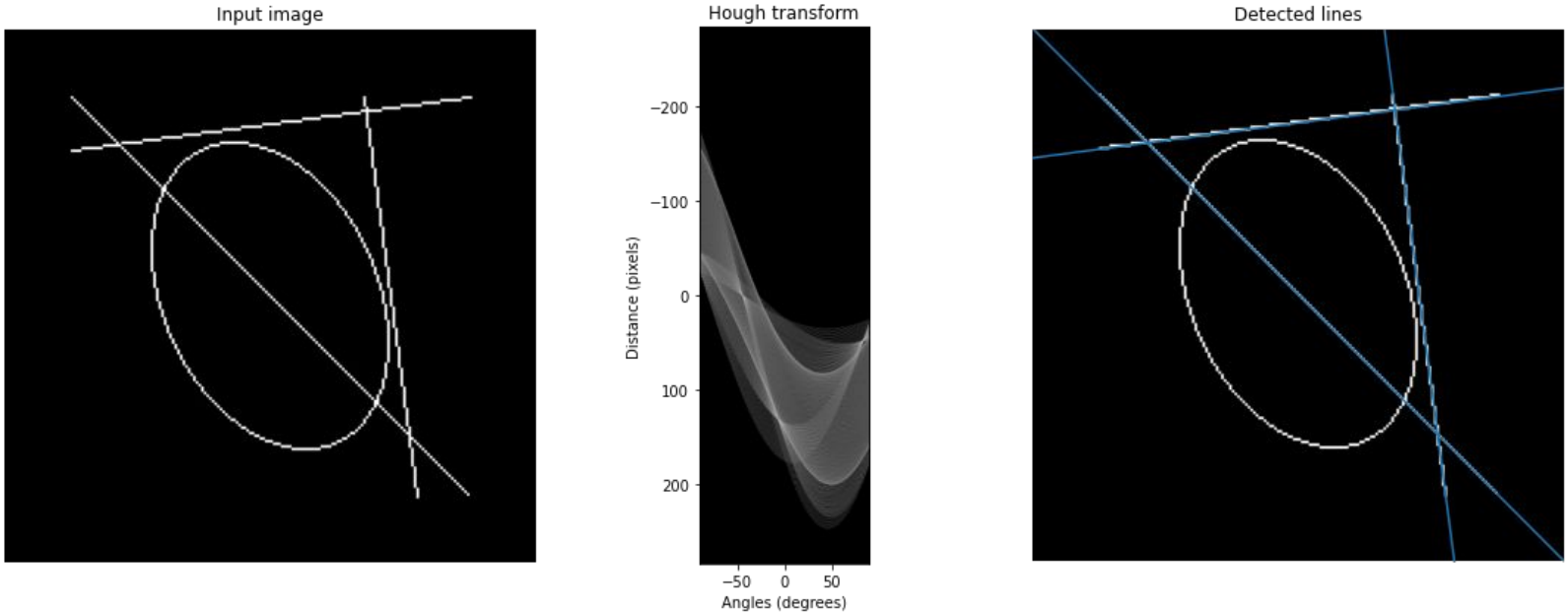






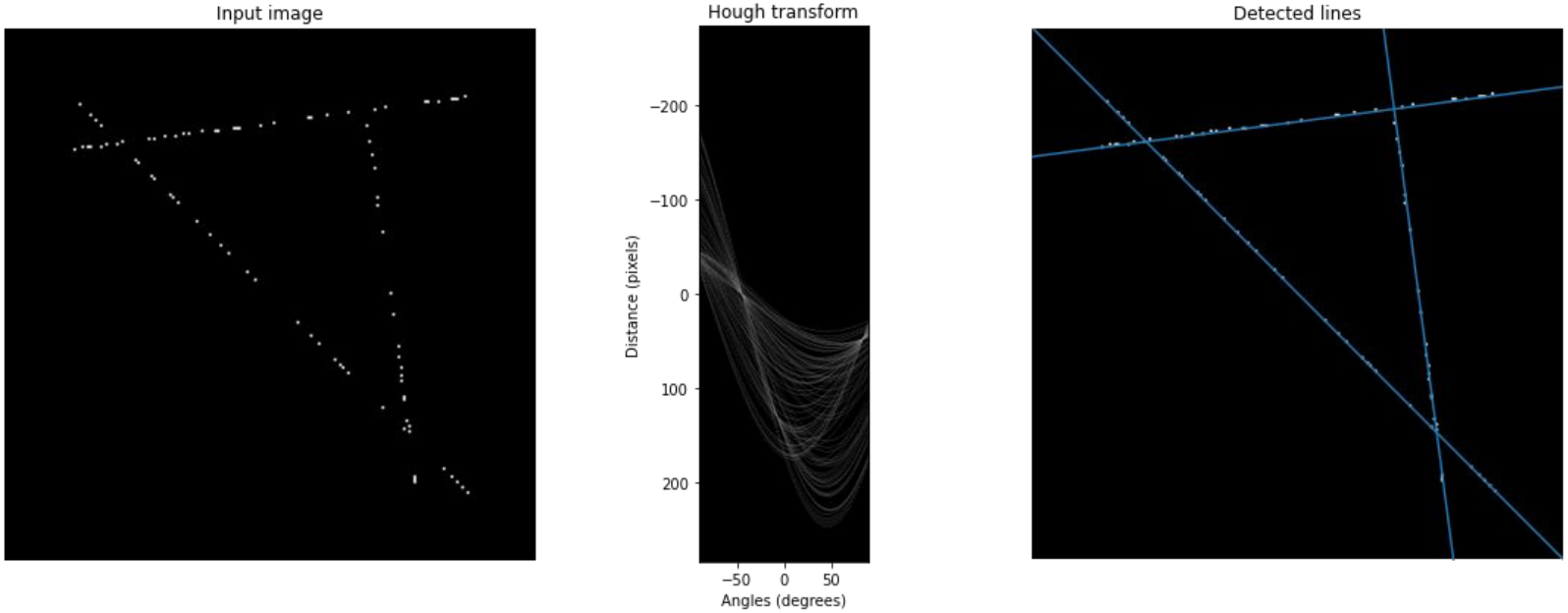
Robustness

The presence of other objects in the image impacts the accumulator array, but the locations of the maxima often remain unchanged.



Robustness

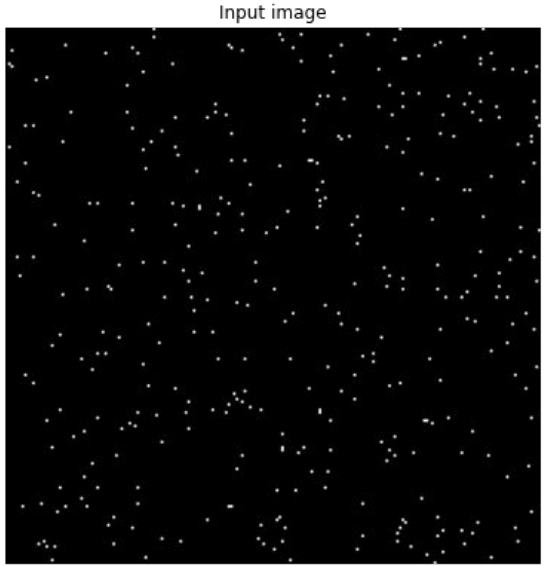
The continuity of the line doesn't matter much: all that matters is whether an equation is satisfied or not.
Below: 80% of randomly selected pixels replaced by pepper-type noise (zeroed).

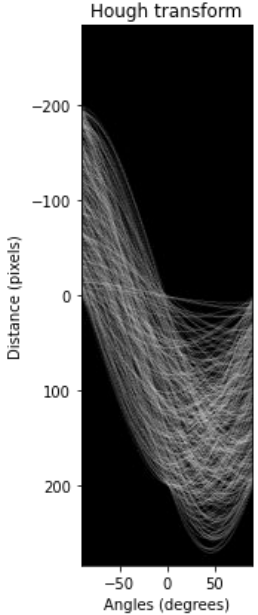


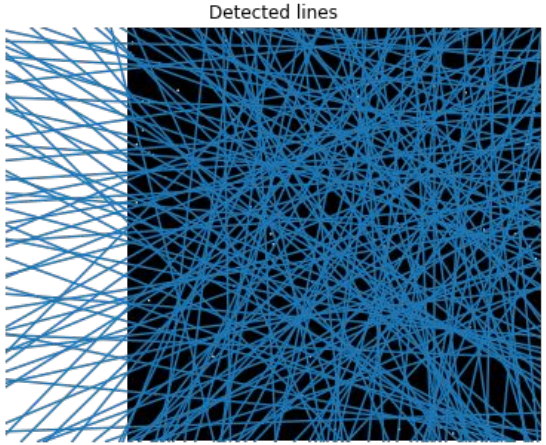
False positive detections

Points can accidentally line up.

Below: The input image contains 1% randomly selected lit pixels (salt noise).







Probabilistic variants of the Hough Transform

1. Probabilistic Hough Transform
   - Uses a random sample of points in the voting process.
   - In easy use cases, sampling even as little as 2% of image points may be sufficient for successful detection.
   - Requires determination of sample size, which strongly depends on the nature of the image, in particular the amount of noise.

N. Kiryati, Y. Eldar, and A. M. Bruckstein. 1991. A probabilistic Hough transform. Pattern Recogn. 24, 4 (1991), 303–316. DOI:https://doi.org/10.1016/0031-3203(91)90073-E

Probabilistic variants of the Hough Transform

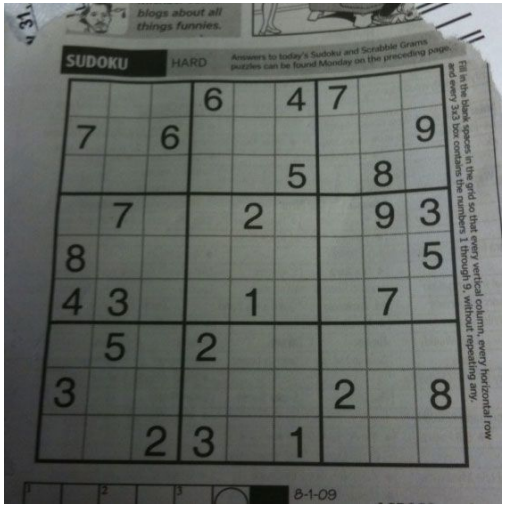
2. Progressive probabilistic Hough transform

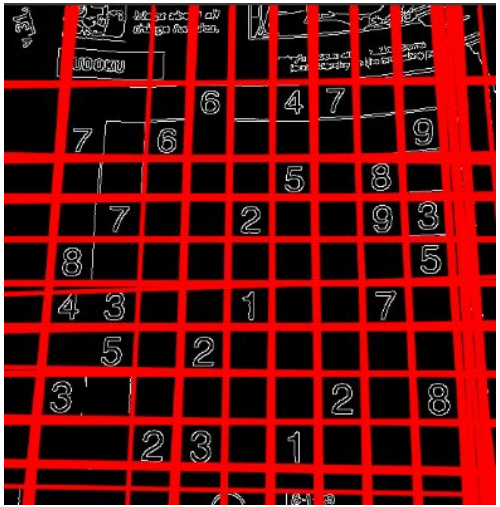
- Aims at reduction of the number of sampled image points, while maintaining possibly low false negative and false positive error rates.
- When the value of the accumulator cell updated in the last voting exceeds a certain threshold, the method analyzes the line around the current (last voting) point, and constructs a segment (a sequence of pixels with gaps no longer than a certain threshold).
  - The determined segments are removed from the input image.
- The method can determine the positions of extreme points on a given line.
- This makes it suitable for detection of *segments* (rather than entire lines).

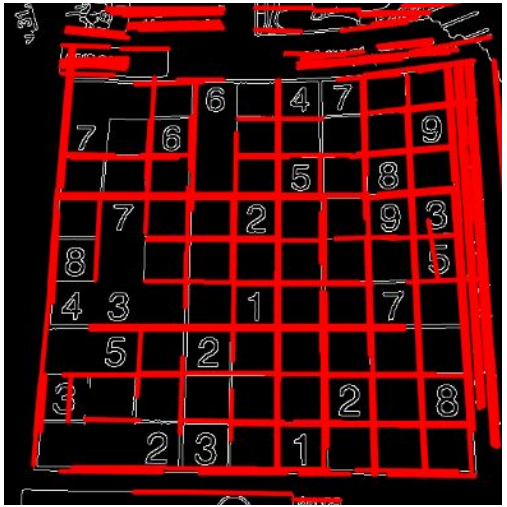
Jiri Matas, Charles Galambos, and Josef Kittler. Robust detection of lines using the progressive probabilistic hough transform. *Computer Vision and Image Understanding*, 78(1):119–137, 2000.

Examples

1. Input image (subsequently subject to contour detection using Canny detector; the result visible as the black and white component in the other images)
2. Result of the standard Hough transform
3. Result of the probabilistic Hough transform (segment detection)

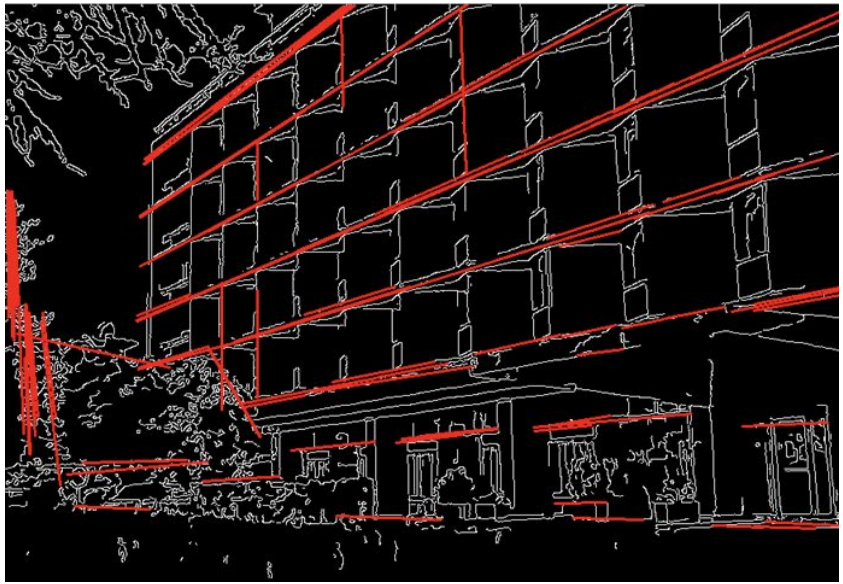






Examples

1. Input image
    a. Then subject to contour detection (Canny edge detector, white pixels in right image)
2. Result of probabilistic Hough transform



Pros and cons of the Hough Transform

Advantages:

- robustness to noise,
- reconstruction of lines based on points that do not have to be connected.

Disadvantages:

- False positive errors: 'illusory' lines can be produced,
- A countermeasure: a second phase of voting, in which each point examines the votes it has cast in the accumulator array, and selects from them only the one for which the accumulator value is maximum.

Possible extensions

Addressing the negative effects of discretization in the dual space:

- Interpolation of increments.

The basic algorithm assumes a binary input image. What if the input image is not binary?

- Incrementation by a value that depends on the brightness of the point, e.g. acc[a,b] += f(x,y)

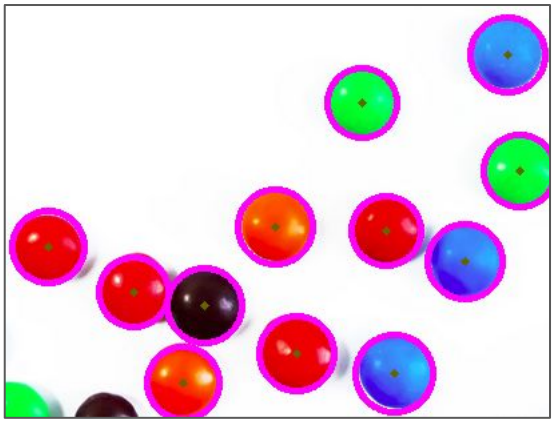
Hough Transform allows detection of arbitrary curves given in analytical form.

- However: the more parameters, the more dimensions of the accumulator array. Implications:
  - Higher memory and computational complexity,
  - Sparsity of increments.

Hough Transform for circles

Three-dimensional dual space: $X_o \times Y_o \times R$

$$(x - x_o)^2 + (y - y_o)^2 = r^2$$



Problem: requires:

- a three-dimensional array of accumulators,
- iteration over multiple (>1) parameters.

Implication: high computational complexity.

Hence, for example, the HoughCirles() method in OpenCV implements a two-phase heuristic (the so-called *Hough gradient*):

- Detection of candidate centers of circles.
- Estimation of the optimal radius for each candidate.

Generalized Hough Transform

Representation of a shape 'formula': similar to the signature, i.e., one of the following:

1. The list of displacements (vectors) of contour points relative to a reference point ('center'* of the object)
    - independent variable: the point number.
2. The list of distances of contour points from the reference point
    - independent variable: angle.

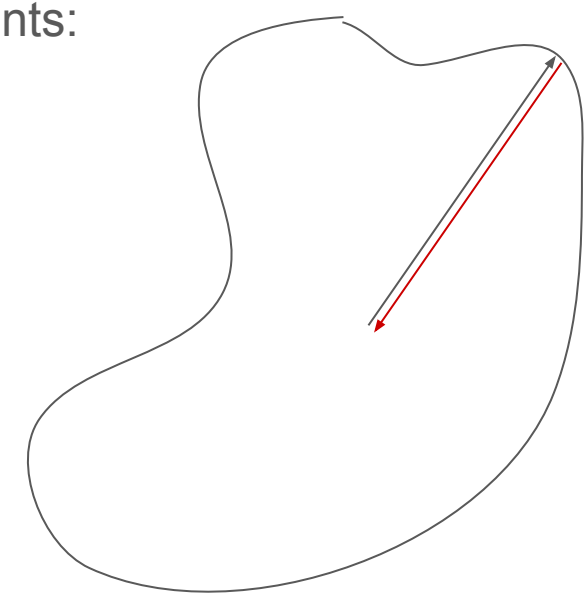
* e.g. center of gravity.

Can only be used to look for specific shapes in the image.

Generalized Hough Transform

Pseudocode for the variant with a list of displacements:

```
function hough(f, contour)
     acc = zeros(f)   // clear the accumulator array
     for x=1 to xMax  // loop through the image points
      for y=1 to yMax
       if f(x,y) > 0  // if point is on (lit)
        for i = 0 to len(contour)
         acc[ x-contour[i].x, y-contour[i].y ]++
     return acc
```



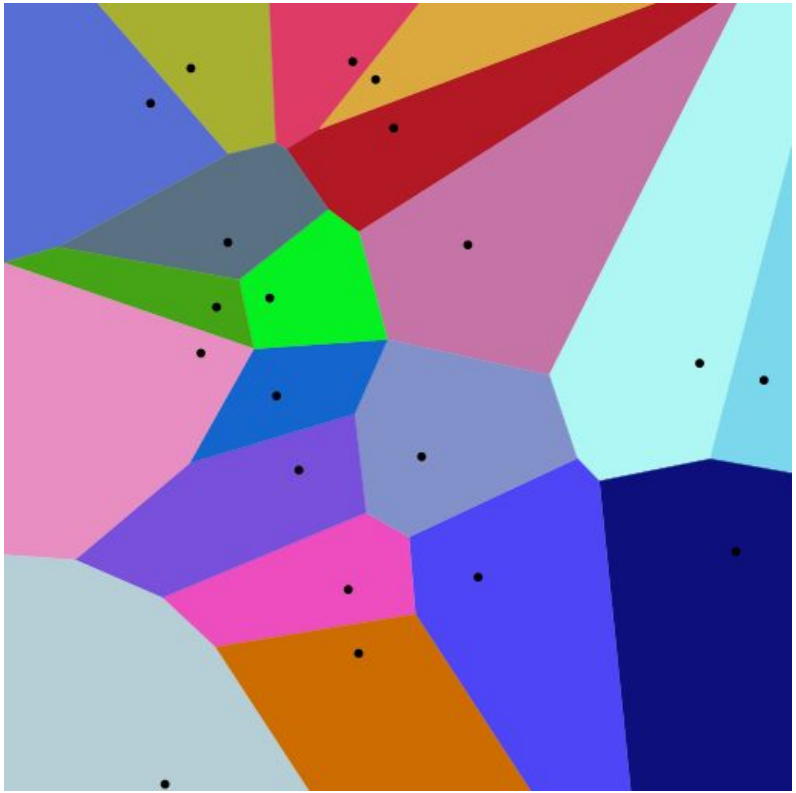
- Figure: The black arrow represents the vector [contour[i].x, contour[i].y]
- The contour point 'votes' for all len(contour) possible positions of the object's center.

Other concepts useful in image analysis and feature extraction

Voronoi Diagram

A Voronoi diagram (VD) is a decomposition of a metric space defined by a finite set of objects (usually points) S.

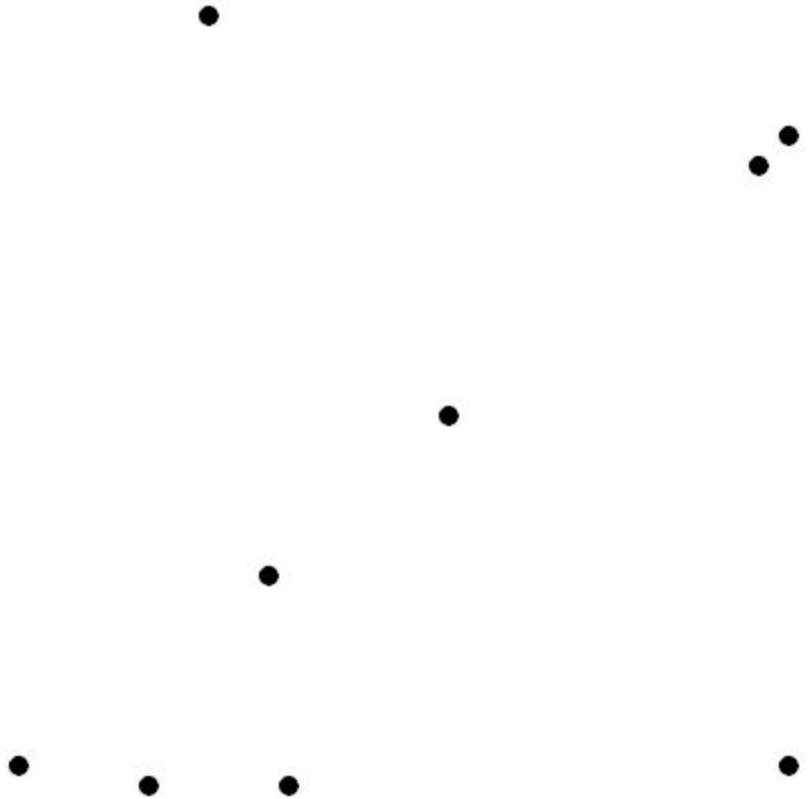
- Each point p of S uniquely determines a convex polyhedron consisting of such points x that, of all points in S, p is closest to x.
- The set of all such |S| polyhedra forms the Voronoi Diagram.



Example: Animation

The facets of the Voronoi diagram are defined by the points that are equidistant from the points of the set S.

- (and closest in the sense used in the previous slide)



Георгій Феодосійович Вороний

Georgiy Fieodosjevich Voronoy (Ukraine)

- Professor at the University of Warsaw
- Mentor of Wacław Sierpiński

Georgy Voronoy

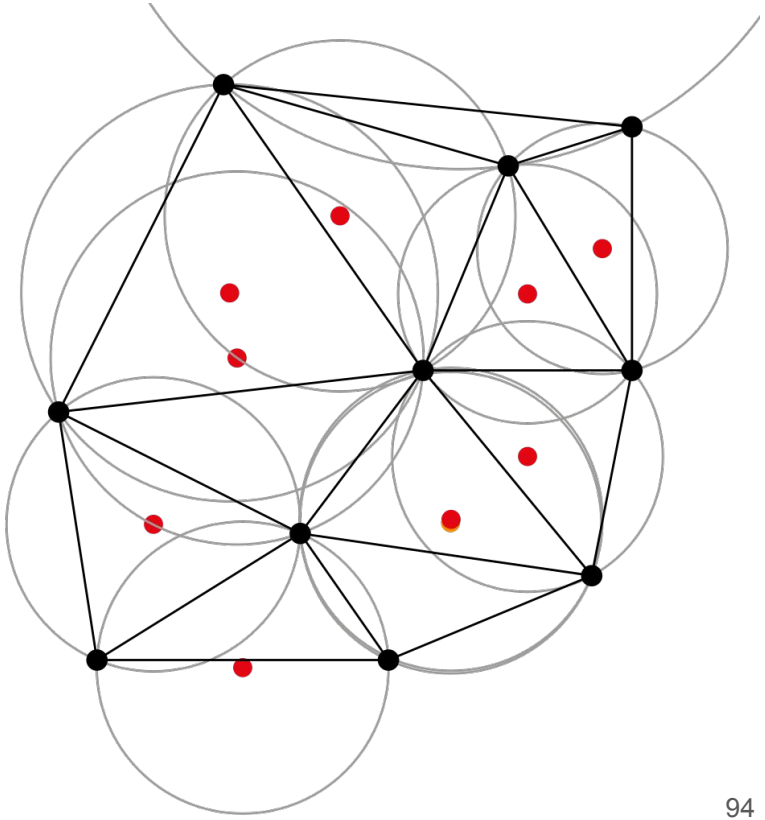
Born	Georgy Feodosevich Voronoy (Георгій Феодосійович Вороний) 28 April 1868 Zhuravka, Poltava Governorate, Russian Empire
Died	20 November 1908 (aged 40) Warsaw, Congress Poland, Russian Empire

Relation between VD and Delaunay Triangulation

Delaunay Triangulation: partitioning of a plane into triangles spanning points from S such that:

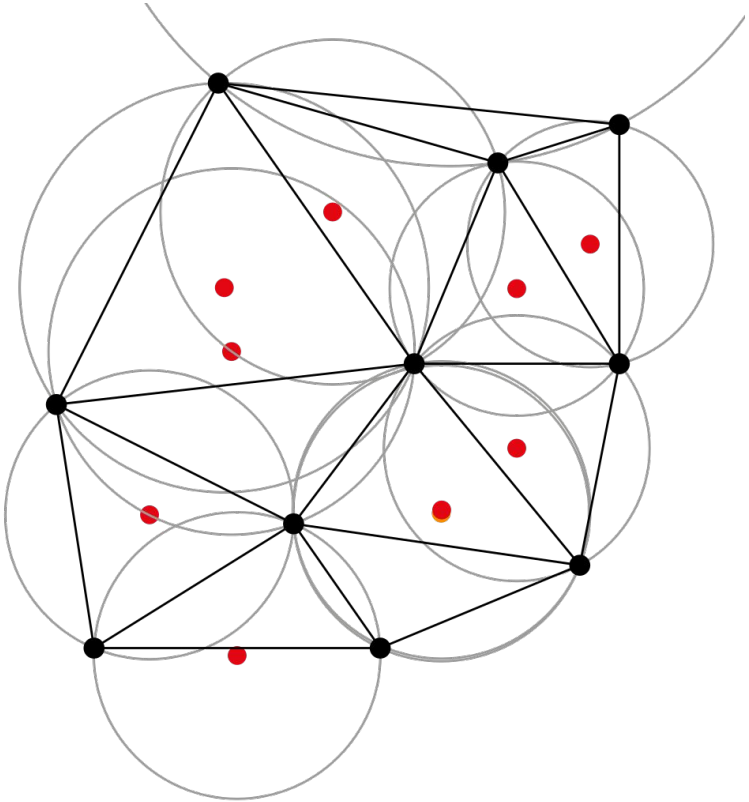
- Each pair of triangles has zero or one common side/facet (edges do not intersect).
- No point of S is contained by any of the circumscribed circles (circumcircles) of the triangles.

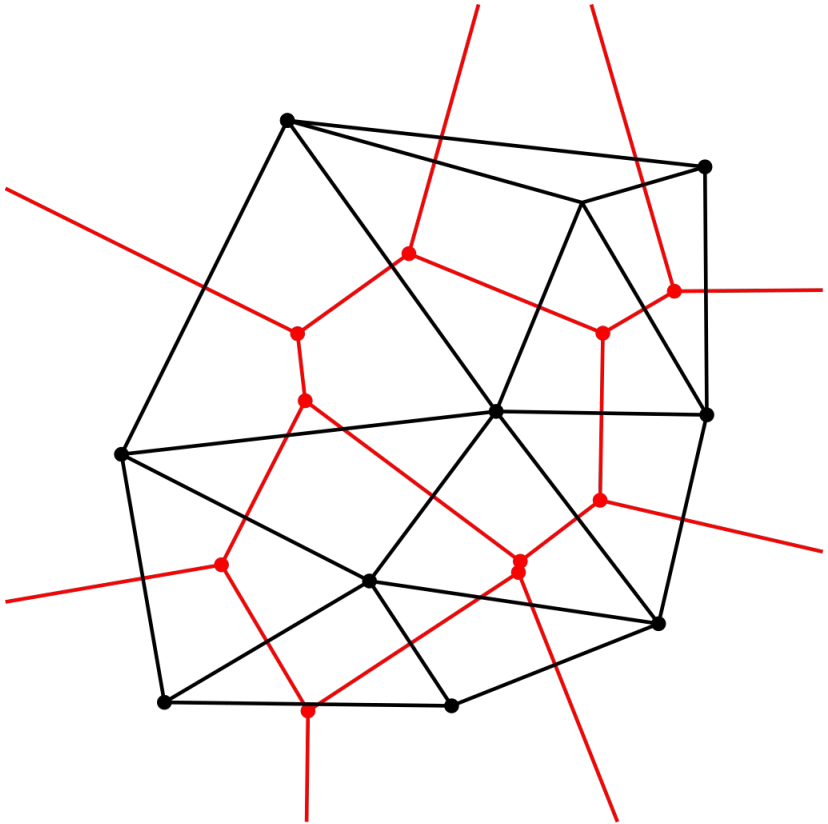
The perpendicular bisectors of the edges of the triangles are the faces/facets of the polygons of the Voronoi diagram.



https://commons.wikimedia.org/wiki/File:Delaunay_circumcircles_centers.svg

Transition from DT to VD





Fractal dimension

Fractal dimension: Definitions

1. A geometric figure or natural object with the property of self-similarity: its parts have ‘the same shape’ as the whole, modulo certain transformations (e.g., affine transforms).

- We consider only non-trivial notions of self-similarity.
  - E.g. we are not interested in the self-similarity of a segment to the line on which it is located.

2. Any shape such that when measuring its properties such as length, area, volume, using discrete and finite units, the measured quantity diverges to infinity when the size of the unit goes to zero.

3. Hausdorff definition (formal): Any geometric figure with an irrational Hausdorff dimension.

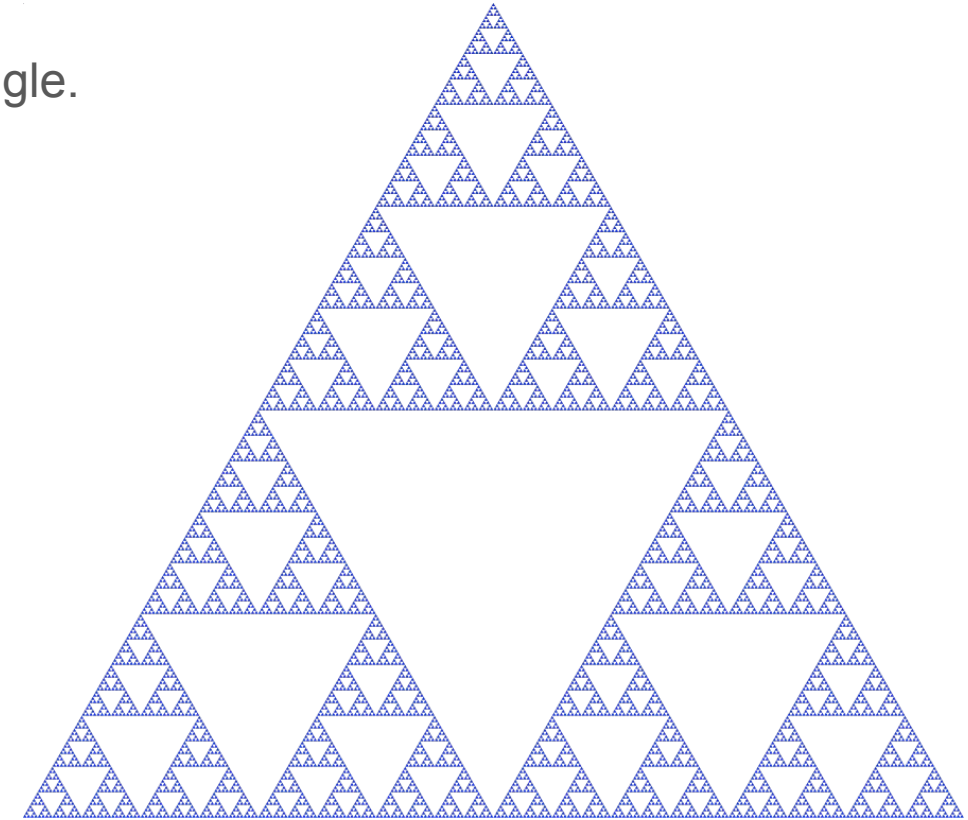
Example: Sierpiński triangle

Results from recursive removal of "sub-triangles" from an equilateral triangle.

Wacław Sierpiński

1928

Państwo działania	Polska
Data i miejsce urodzenia	14 marca 1882 Warszawa, Królestwo Polskie
Data i miejsce śmierci	21 października 1969 Warszawa, Polska



Fractal dimension

In image recognition we can try:

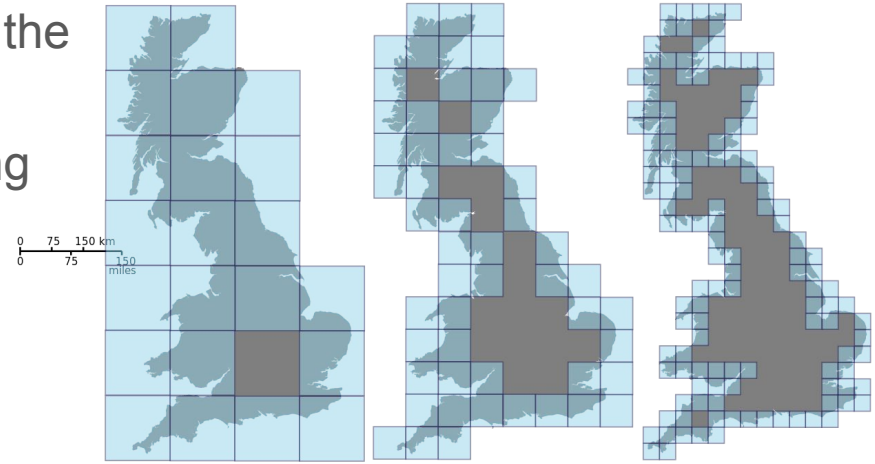
- Treat the objects being analyzed as if they were (approximate) fractals.
- Measure (estimate) their fractal dimensions.

Popular methods:

- ruler method,
- box counting method (so-called *box dimension*).

Box dimension

1. Estimate of the area of the figure using the "area unit" – a b×b square.
2. Repeat the measurement in step 1 using units of size b/2, b/4, b/8, ...
3. Determine the dependency N(b), the number of placements of the unit.



Box dimension (Minkowski-Bouligand):

$$\lim_{b \to 0} \frac{\log N(b)}{\log(1/b)}$$

where N(b) is the number of offsets (‘placements’) of unit b.

Summary

Module outline

1. Introduction
2. Geometric features
    a. Simple geometric features
    b. Signature
    c. Skeleton
    d. Chain codes and edge description in the frequency domain
    e. Polygonal approximations
    f. Moments
3. Non-geometric features
    a. Texture analysis
4. Related topics
    a. Hough transform
    b. Voronoi diagram
    c. Fractal dimension

Credits

- Wikipedia: https://en.wikipedia.org/
- OpenCV documentation: https://docs.opencv.org/
- Scikit-image documentation: https://scikit-image.org/