ITMO

Hough Transform
Image Features

Computer Vision

Outline

- **Hough Transform**
  - "Voting" points method
  - Hough transform for straight lines
  - Hough transform for circles
  - Hough transform for an arbitrary shape
- **Image Features**
  - Feature points
  - Feature point detectors
  - Blob detectors
  - Scale-invariant feature point detector

Hough Transform

"Voting" points method

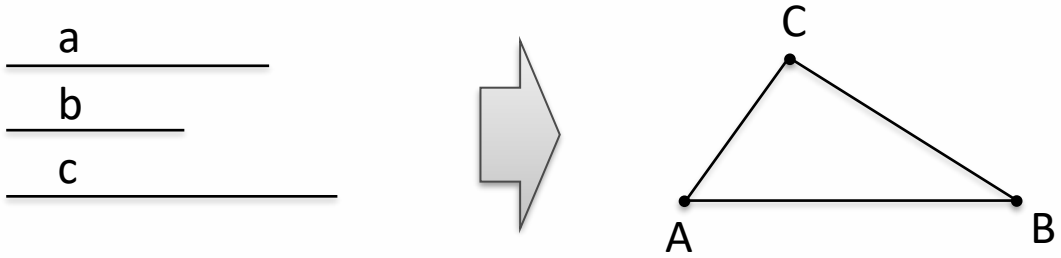
A method of "voting" points

Idea: finding the locus of points satisfying the given criteria, then find the intersection of all loci

The locus (loci, pl.) is a set of points whose location satisfies the specified conditions

Example:

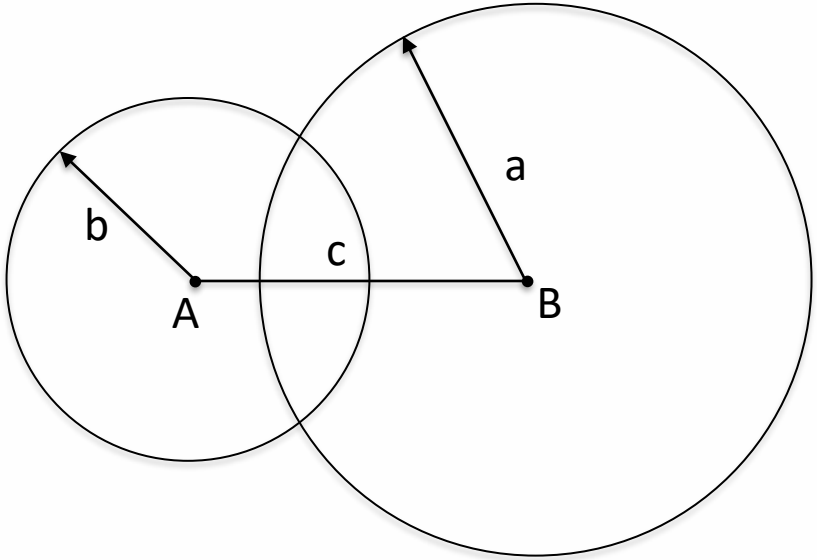
Task: building a triangle by three given side lengths



"Voting" points method

Solution:

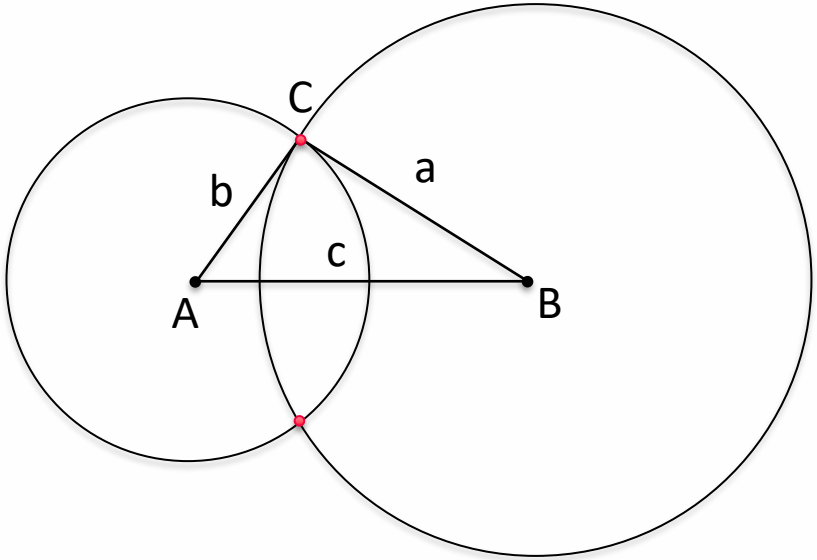
1. Draw one side of the triangle in an arbitrary place
2. Draw two circles with centers at the ends of the first side and radii equal to the lengths of the second and third sides of the triangle
3. Circles are the loci of the ending points of the second and third sides of the triangle correspondingly



"Voting" points method

Solution:

4. For all points of each circle, the following condition is met:
   - The distance from the circle center is equal to the length of the corresponding triangle side
5. The intersection of circles is a common locus (so the required third triangle vertex is located in the circles intersection area; the common locus contains two points)



"Voting" points method

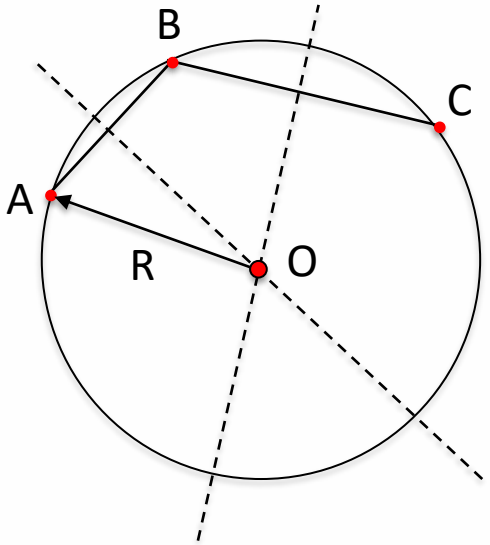
Generalization of the method:

- Points of each circle "voted" in favor of the possible position of the vertex
- There are two “votes” at the intersection of the circles; in this case, this point “won”, since it gained the maximum number of “votes”
- The rest of the points of the plane received zero or one “ vote ”
- The shape of the "voting curve" is determined by a priori knowledge about the voting object (in the example, the sides of the triangle are given)

"Voting" points method

An example of constructing a circle from three given points

- **Locus:** the perpendicular bisector of each line segments connecting pairs of points
- **Solution:** find the point of intersection of the perpendicular bisectors



"Voting" points method

Generalization of the method

Task: detection of a circle of known radius in a binary set of points

Solution:

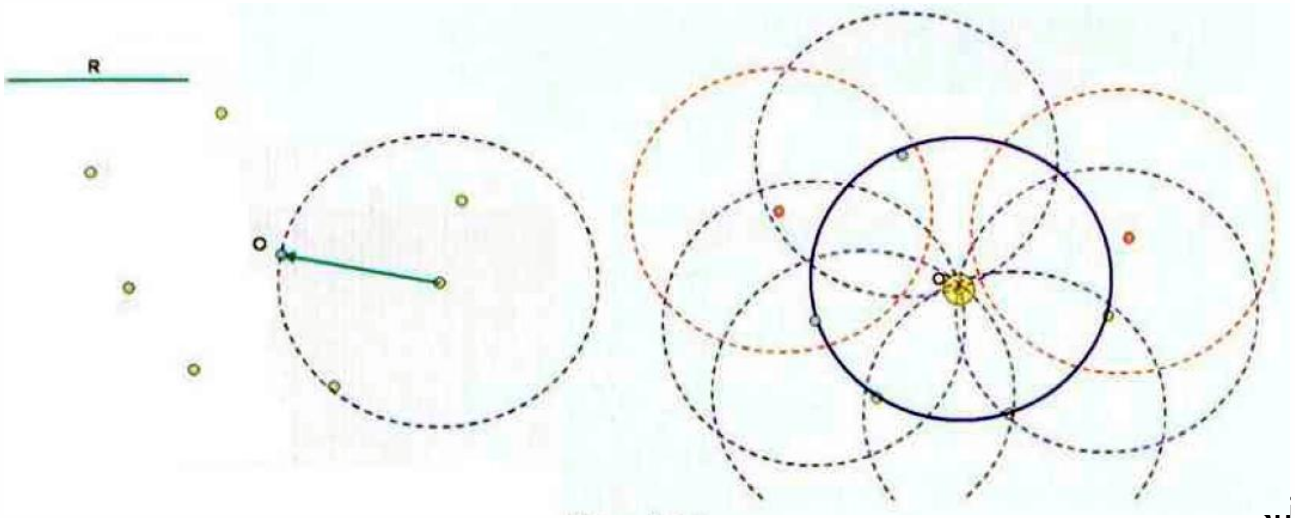
- The set of centers of all circles of radius R passing through each point forms a circle of radius R around that point
- The locus of the points is a circle of the same size with the center at the voting point
- The best solution for finding the position of the center of a circle with given radius is the intersection point of the maximum number of voting circles

"Voting" points method

Generalization of the proposed approach

Task: detection of a circle of known radius in a binary set of points

Solution: the “winner” is a large yellow dot in the center and a solid circle



"Voting" points method

- **Solution reliability:** the greater the ratio of the number of points lying on the found circle to the total number of points, the better. This is the signal-to-noise ratio

- In the considered problem, we searched for one of the most probable circles, so the “less popular” circles were ignored

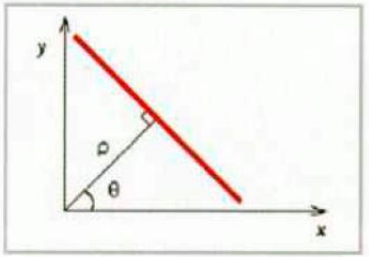
- To search for all possible circles, it is necessary to determine the threshold or "electoral qualification"

- The main problem: how to organize the computation process of generating locus and collecting votes, if the number of points in the image is thousands?

Hough transform for straight lines

Hough transform for finding straight lines

The classic Hough Transform (HT) was proposed in 1962 for finding lines in a binary image.



Idea: convert the set of points to a parameter space

- Let a straight line can be defined as

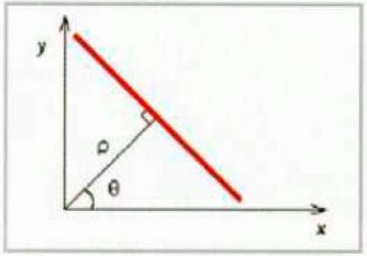
$$Y = kX + b$$

$$X\cos\theta + Y\sin\theta = \rho$$

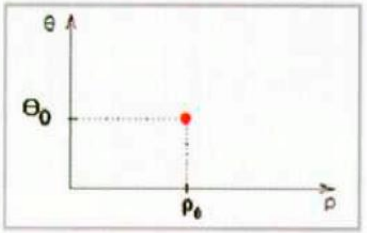
- Since a straight line on a plane is characterized by two parameters, the parameter space has the dimension $n = 2$
- The classical Hough transform uses parameters (ρ, θ)

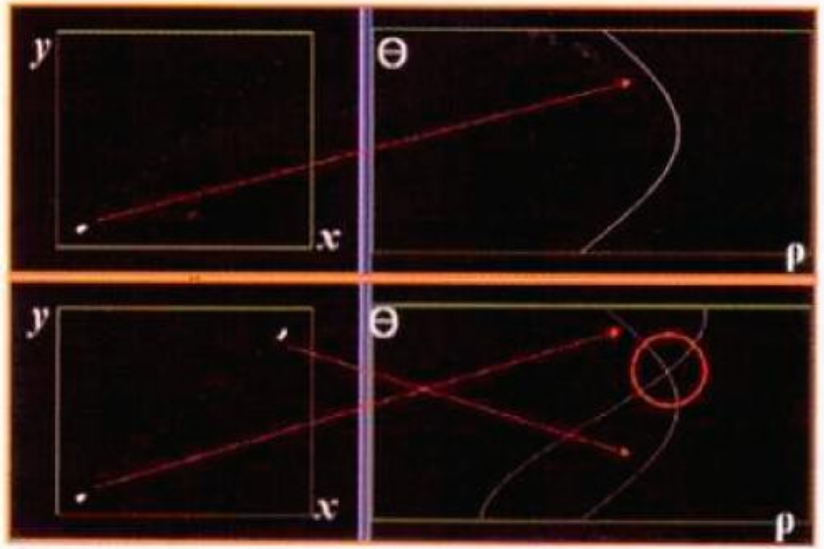
Hough transform for straight lines

- Let the contour image be the set of points (x, y) in the original space $E = (X, Y)$
- The set of lines passing through each point (x, y) can be depicted as a set of points (ρ, θ) in the space $\{\rho, \theta\}$
- The function of mapping a point in Hough space is called a "*response function*"



$$X\cos\theta + Y\sin\theta = \rho$$





Hough transform for straight lines

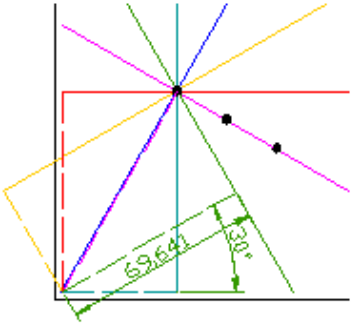
General idea of the method

- For each point of the parameter space, the number of votes given for it is summed up (*the number of points in the source image space that generate response in the parameter space that passes through a given point* (ρ, θ))

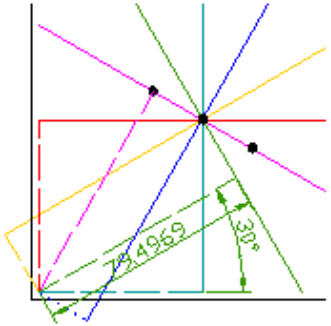
- Each point of the source space generates a sinusoidal response function in the parameter space

- Any two sinusoids in the parameter space intersect at the point *(ρ, θ)* only if the point of the source image space which generated them lie on a straight line

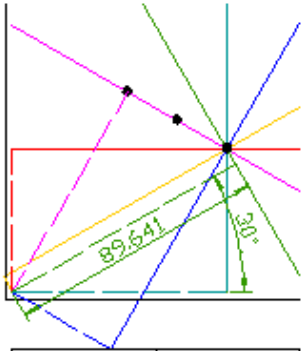
Hough transform for straight lines



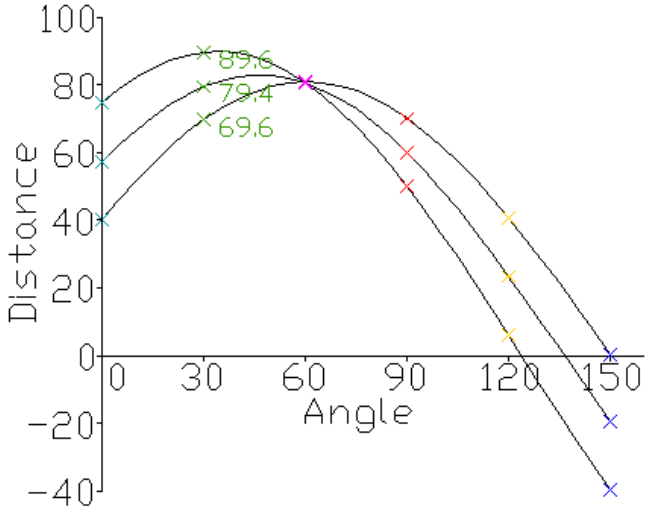
Angle	Dist.
0	40
30	69.6
60	81.2
90	70
120	40.6
150	0.4



Angle	Dist.
0	57.1
30	79.5
60	80.5
90	60
120	23.4
150	−19.5



Angle	Dist.
0	74.6
30	89.6
60	80.6
90	50
120	6.0
150	−39.6



Hough transform for straight lines

- Based on the above assumptions, we can introduce the accumulator function $A(\rho, \theta)$
- The absolute value of the accumulator function at the point (ρ, θ) is equal to the number of contour points lying on the corresponding straight line in the source image space
- If the image contains m lines, then the accumulator function $A(\rho, \theta)$ will have exactly m local maxima
- To find straight lines in the original image, it is necessary to find all significant local maxima of the accumulator function

Hough transform for straight lines

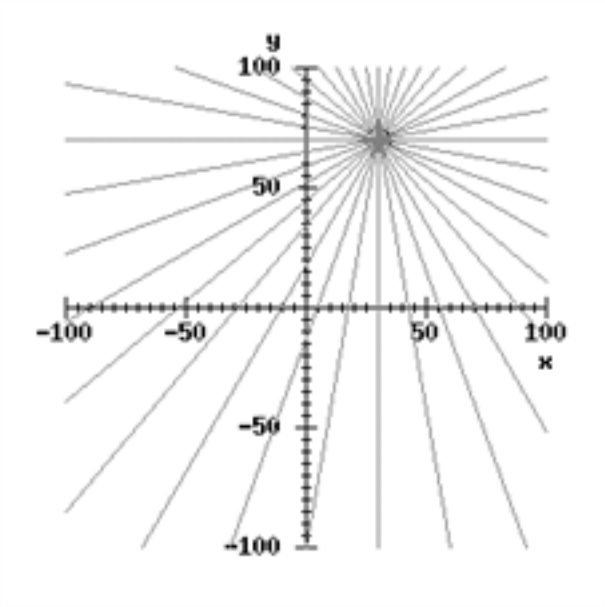
- The algorithm does not rely on the assumption that the analyzed line is connected; therefore, voting methods work well in conditions of obstruction or other interferences

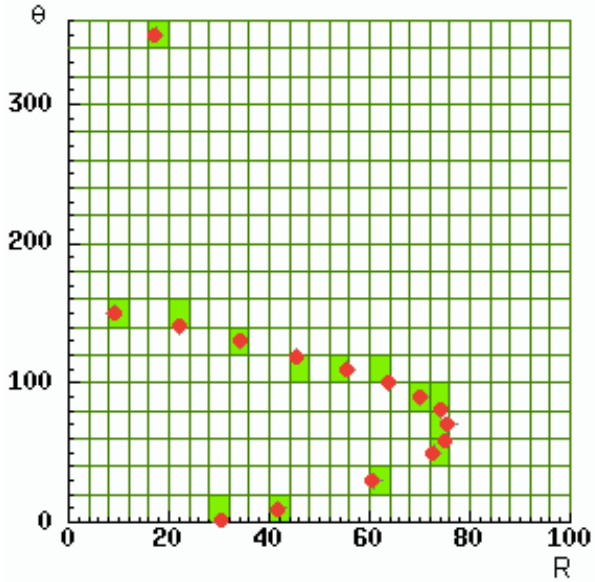
- The accumulator function $A(\rho, \theta)$ is calculated not for every point of the parameter space, but instead for every “accumulator cell”

- **Accumulator cells** are some areas of the rectangular grid into which the parameter space is partitioned

- We discretize the parameters space. After that, each line in the space (x, y) corresponds to a point in the discrete parameters space (ρ, θ)

Hough transform for straight lines





Each cell *(ρ, θ) of the* parameters space accumulates “votes” of dots (x, y) of the source image space which satisfy the condition:

$$X\cos\theta + Y\sin\theta = \rho,$$

where $\Theta_i \le \Theta \le \Theta_{i+1}$

$\rho_i \le \rho \le \rho_{i+1}$

Hough transform for straight lines

Properties of the Hough transform

- The Hough transform is invariant to translation, scaling, and rotation

- Since straight lines under any projective transformations of three-dimensional space are always transformed only into straight lines (in the degenerate case, into points), so the Hough transform makes it possible to detect lines invariantly not only to planar affine transformations, but also to the group of projective transformations in 3D space

Hough transform for straight lines

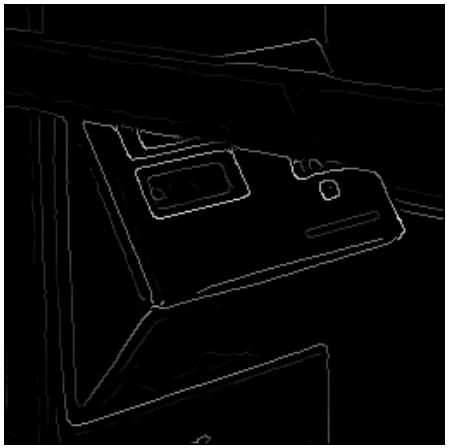
Transformation efficiency

1. Under projective transformations, the straight line always turns into a straight line; therefore, the parameters space of low dimension (n = 2) is formed

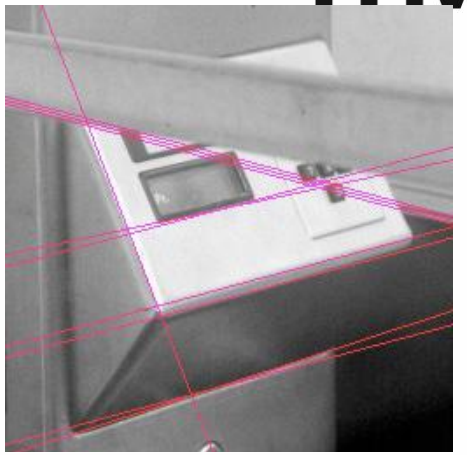
2. Each pixel of the source image is processed only once, and further calculations are performed only for pixels that carry useful information (i.e., outline). The fewer the number of pixels that have the useful information, the higher is the computational efficiency

Hough transform for straight lines

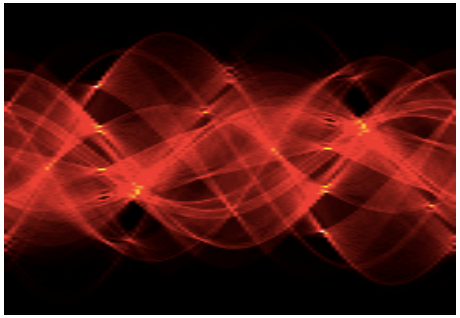
Source image



Selected contours



Found straight lines

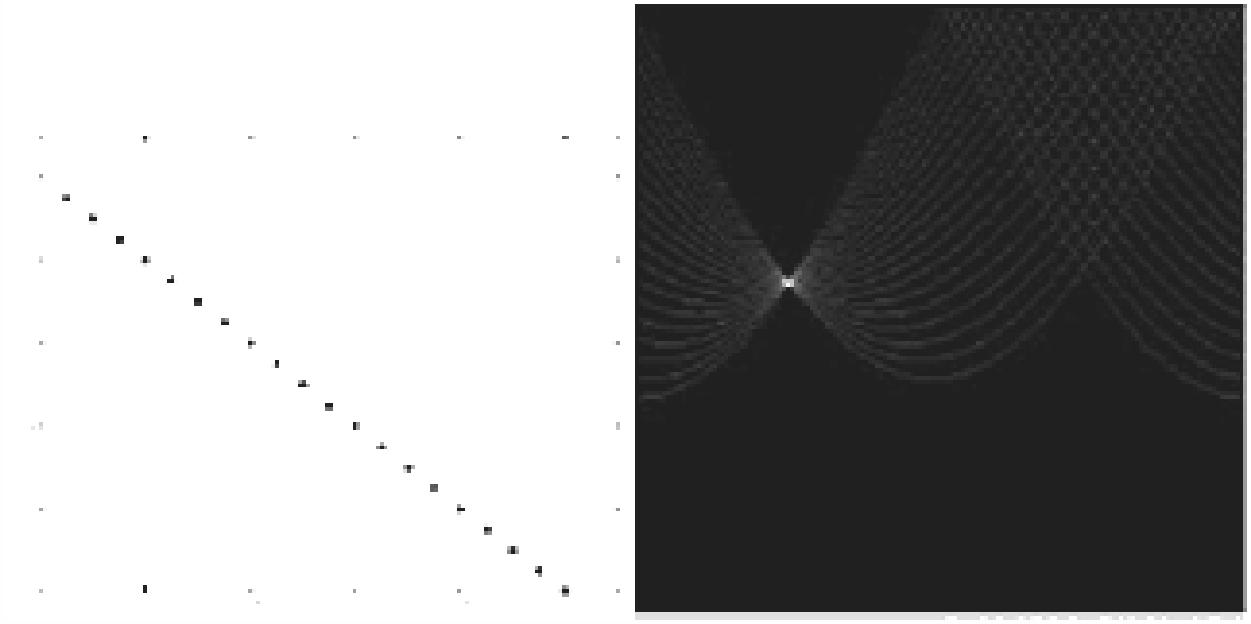


Parameter space

Hough transform for straight lines

Disadvantages of Hough transform:

1. Manual selection of phase space sampling

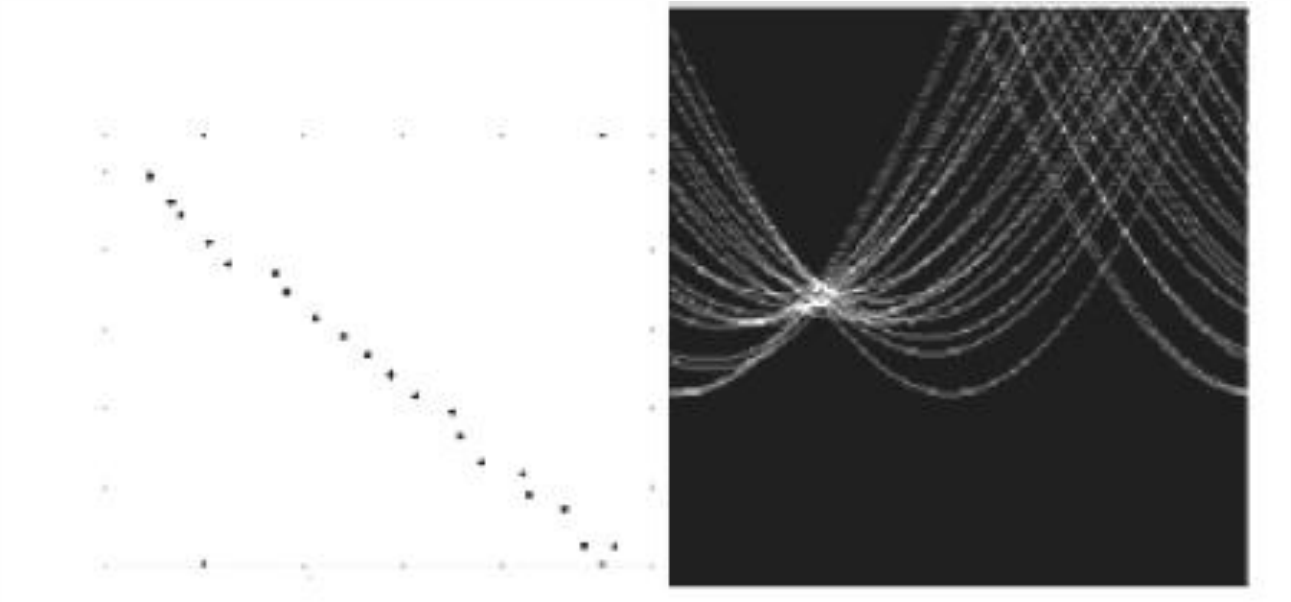


Points and phase space without noise

Hough transform for straight lines

Disadvantages of Hough transform:

2. Noise leads to blurring of maxima

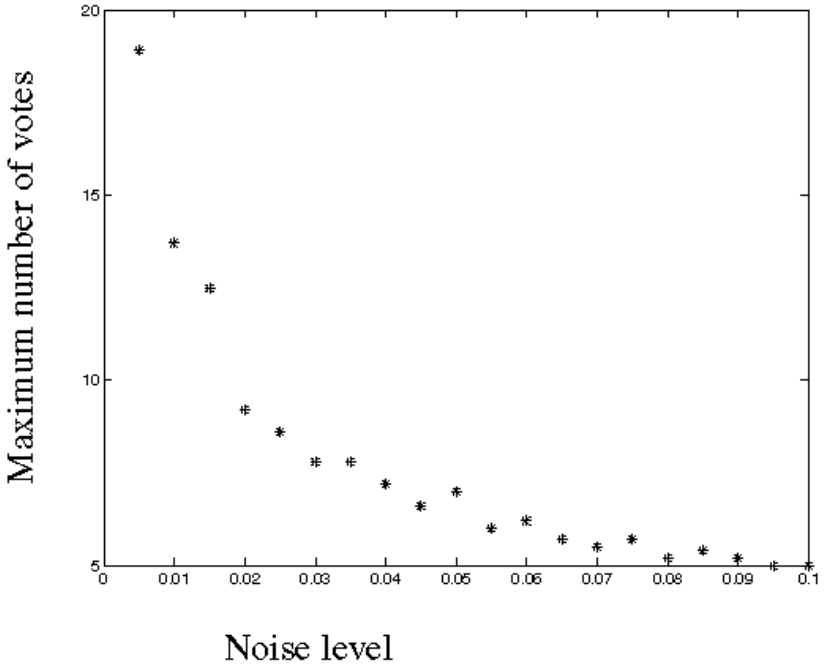


Noisy points and peak blur the phase space

Hough transform for straight lines

Disadvantages of Hough transform:

2. Noise leads to blurring of maxima

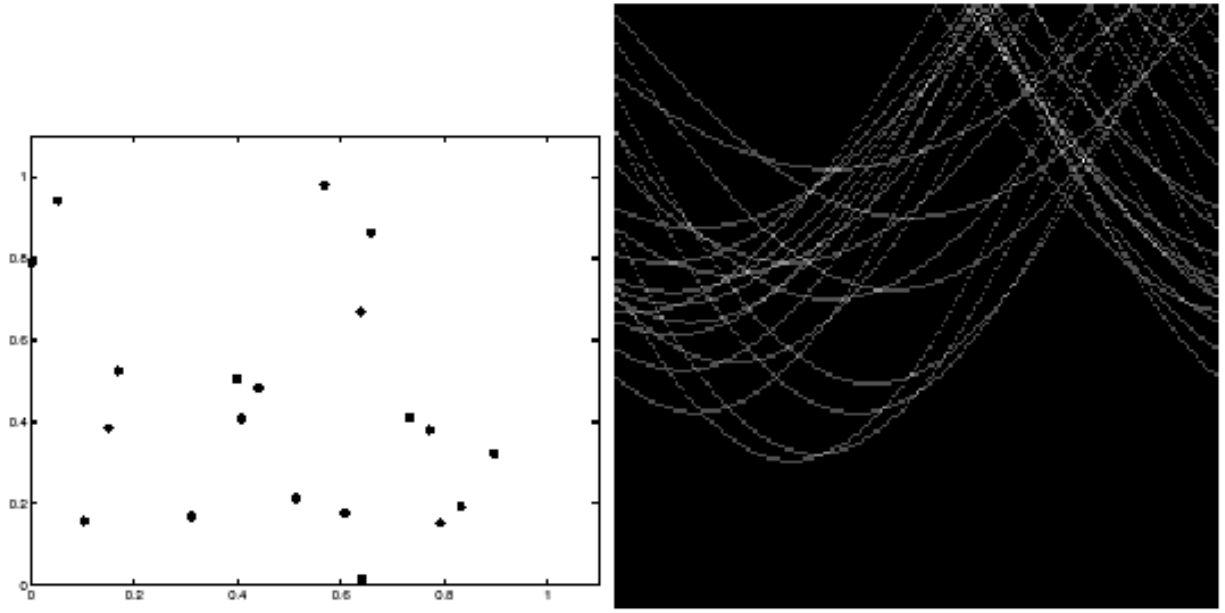


Dependence of the number of votes on the noise level

Hough transform for straight lines

Disadvantages of Hough transform:

3. Evenly spaced points can lead to random peaks in the parameters space:

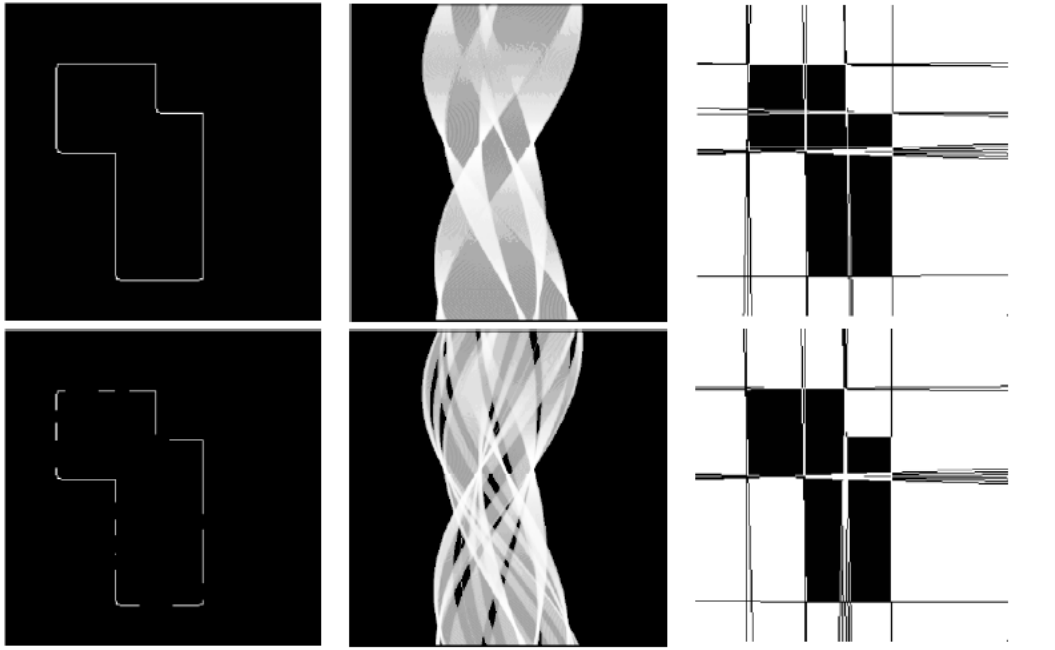


On the left are the random points; on the right - the response functions

Hough transform for straight lines

Disadvantages of Hough transform:

4. Missing parts of source image data results in blurry values in the accumulator



Data blur in the accumulator due to missing data

Hough transform for straight lines

ITMO

Optimization of the Hough transform

1. **Filter the unnecessary features:** for lines it is worth taking points at the edges only with a large gradient

2. **Choose the correct grid resolution** (sampling step):
   Too rough: several close lines will vote for same cell
   Too shallow: can skip lines because noisy points will vote for different cells

3. To better find the maxima, you can smooth the values in the accumulator

4. **It is necessary to mark the votes:** which point corresponds to which line

Hough transform for straight lines

Optimization of the Hough transform

In the classic version, the Hough transform can be represented by the following pseudocode:

```
For each edge point (x,y)
   For θ = 0 to 180
      ρ = x cos θ + y sin θ
      H(θ, ρ) = H(θ, ρ) + 1
   end
end
```

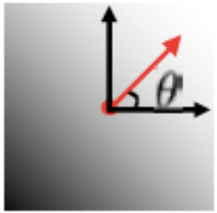
Hough transform for straight lines

Optimization of the Hough transform

In order not to enumerate all possible angles, the effect of the gradient can be taken into an account. Since when detecting the contour, the value of the gradient has already been calculated

When taking the gradient into an account :

```
For each edge point (x, y)
   θ = gradient orientation at (x, y)
   ρ = x cos θ + y sin θ
   H(θ, ρ) = H(θ, ρ) + 1
end
```



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

Hough transform for straight lines

ITMO

Methods for parametrizing straight lines

- As is known, images in digital systems are defined on a discrete rectangular grid, which allows only some appropriate discrete parameterization of the family of straight lines.
- Consider the natural set of straight lines generated by an integer grid $N \times N$ of points containing N^2 elements.
- Any two different points of the lattice define a line, so the size of the set will be $N^2(N^2 - 1)$ lines.
- Many lines will be defined multiple times by their different line segments if they contain more than two points of the original grid.

Hough transform for straight lines

ITMO

Methods for parametrizing straight lines

- The representation of the set of straight lines in the form of a four-parameter array $[(x_1, y_1), (x_2, y_2)]$ of endpoints is redundant

- Note that the set of lines also generates the set of angles (θ_a), which can be described as $\mathrm{tg}\theta_a = \frac{i}{j}$, where $\{i, j\} \in \{0, 1, \dots, N-1\}$

- Given the symmetry of the tangent $0 < \mathrm{tg}\theta_a \leq 1$, and the angles are repeated in all four quadrants:

$$\mathrm{tg}\theta = [\mathrm{tg}(90° - \theta)]^{-1} = -[\mathrm{tg}(90° + \theta)]^{-1} = -[\mathrm{tg}(180° + \theta)]$$

Hough transform for straight lines

Methods for parametrizing straight lines

- All possible angles θ_a can be obtained using $N(N-1)/2$ lines. In the four quadrants, the number of unique angles N_a must be $2N(N-1)$

- This description is also redundant: some i/j ratios are multiples (for example, $3/12 = 2/8 = 1/4$). The probability that two integers are related in a given way is $6/\pi^2$

- Therefore, the size of the irredundant set of angles N_a and lines N_L on the lattice are equal to :

$$N_a = \frac{12}{\pi^2} N(N-1) = 1{,}216N(N-1)$$
$$N_L = 0{,}23N^2(N^2-1)$$

Hough transform for straight lines

Ways to parameterize the Hough transform

1. Perimeter points (m, n)

- Lines are described by a pair of end points lying on the perimeter of the $N \times N$ grid.
- Obviously, the number of points will be equal to $4N$ or (taking into account symmetry) $1/2 \cdot (4N \times 4N) = 8N^2$ lines
- Since a quarter of these points lie on the same straight line (side of the square), the final size of the parameter array will be $6N^2$

Advantage: Application in the case of an image divided into smaller areas makes it easy to connect lines passing through several such areas, since they close to each other along the perimeter

Disadvantage: information about the angular position of straight lines is not explicitly stored

Hough transform for straight lines

Ways to parameterize the Hough transform

2. Perimeter point and angle (a, n)

- One point of intersection of the straight line with the grid perimeter n $(0 \leq n < N)$ is used
- and an angle defined by the perimeter point a $(-N + 1 \leq a \leq N - 1)$ so that straight line passing through the grid center and point a is parallel to a given one
- The accumulator array contains $4N^2$ elements

Hough transform for straight lines

Ways to parameterize the Hough transform

3. Tilt and offset (a, d)

- The angle a is defined by the direction of a line from the grid center to a perimeter point
- The displacement of the line vertically or horizontally from the center is fixed using the distance from the center to the intersection of the straight line with the Oy or Ox axes
- Generates $3N^2$ or $4N^2$ accumulator cells
- The (a, d) -parametrization is closely related to the (ρ, θ) - parametrization and with the parameters of the equation $y = ax + b$, where a is interpreted as the slope of a straight line

Hough transform for straight lines

Ways to parameterize the Hough transform

3. Tilt and offset (a, d)

- For lines with an inclination of less than 45° d is measured from the center to the intersection of the straight line with the vertical axis Oy
- For lines with a slope greater than 45° d is measured along the horizontal Ox axis.
- To preserve the continuity of the display, it is necessary to change the sign of d for angles $45° < \theta_a \leq 90°$.
- Mapping (x, y) to (a, d):

$$d = \begin{cases} y - 2ax(N-1) \text{ for } 0 \leq |a| \leq \dfrac{N-1}{2}, \\ -\left[x - 2y + \dfrac{2ay}{N-1}\right] \text{ for } \dfrac{N-1}{2} < a \leq N-1, \\ \left[x + 2y + \dfrac{2ay}{N-1}\right] \text{ for } -N+1 \leq a < -\dfrac{N-1}{2}. \end{cases}$$

Hough transform for straight lines

Ways to parameterize the Hough transform

4. The foot-of-normal

- The straight line is characterized by the coordinates of the point (x_0, y_0) of the foot-of-normal (perpendicular), which is dropped onto the straight line from the origin (or another base point)
- When we apply the Sobel differential gradient operator to the original image in the plane (x, y) we obtain the components of the local gradient (g_x, g_y) at each characteristic contour point
- Let us define the point (x_0, y_0) as the "foot-of-normal" of the straight line (ρ, θ), passing through the voting point (x, y)

Hough transform for straight lines

Ways to parameterize the Hough transform

4. The foot-of-normal

- We get the ratios:

$$\frac{g_y}{g_x} = \frac{y_0}{x_0},$$
$$(x - x_0)x_0 + (y - y_0)y_0 = 0.$$

- Solving the equations for x_0 and y_0, we get:

$$x_0 = V g_x,$$
$$y_0 = V g_y,$$
$$V = \frac{(x g_x + y g_y)}{(g_x^2 + g_y^2)}.$$

Hough transform for straight lines

Ways to parameterize the Hough transform

4. The foot-of-normal

- Backward transformation:

$$x = x_0 + W g_y,$$
$$y = y_0 - W g_x,$$
$$W = \frac{\left(x g_y - y g_x\right)}{\left(g_x^2 + g_y^2\right)}.$$

Hough transform for circles

ITMO

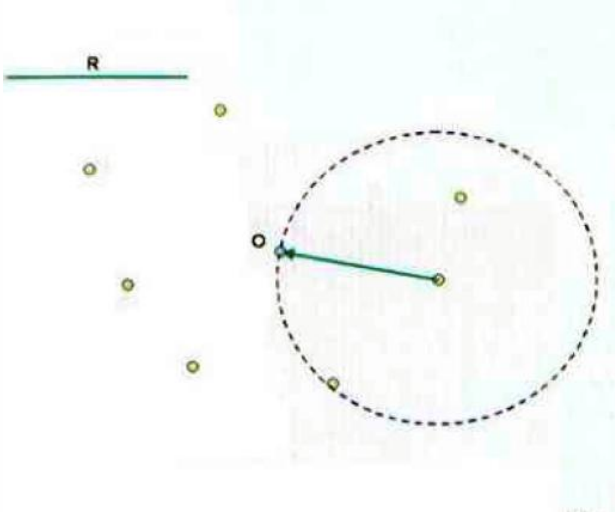
Hough transform for circles

- The described algorithm can work in exactly the same way when detecting any curve that can be described on the plane by a finite number of parameters: $F = (a_1, a_2, \dots, a_n, x, y)$
- In the previously considered problem of finding circles of a given radius R, there was a family of curves defined by a two-parameter equation $(x - x_0)^2 + (y - y_0)^2 = R^2$
- It is necessary to search for the maximum of the accumulator function $A(x, y)$ in the parameter space (x, y)
- In this case, the parameter space practically coincides with the source image one (x, y)

Hough transform for circles

Hough transform for circles

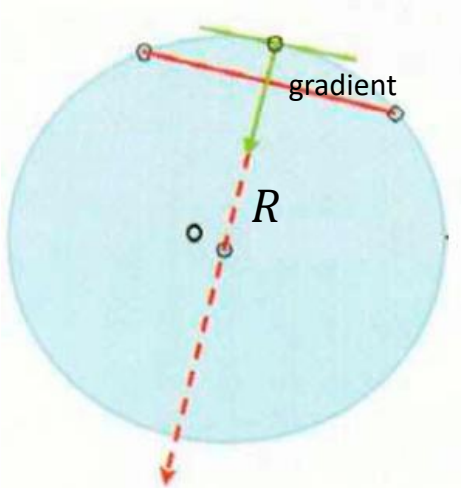
- The set of centers of all possible circles of radius R passing through a given point forms a circle of radius R around this point
- The Hough transform response function for finding circles of a known size is a circle of the same size centered at the voting point
- The maximum of the accumulator corresponds to the position of the center of the circle in the image



Hough transform for circles

Algorithm for finding circles of a given radius on halftone images

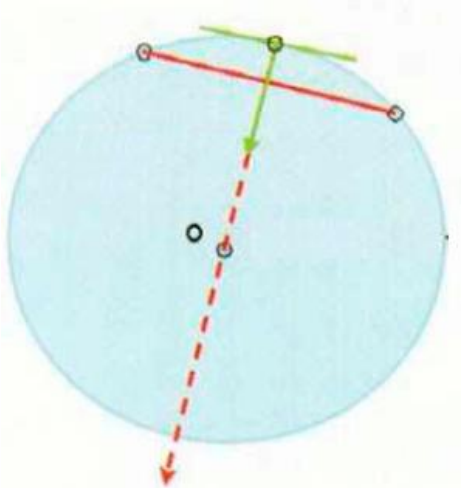
1. Detect edge pixels that surround the perimeter of an object. A gradient operator can be used, such as the Sobel operator
   - Voting contour points are ones with a high gradient
   - For each detected edge pixel, an estimate of the position and orientation of the contour is used to estimate the center of a circle of radius R by moving a distance R from the edge pixel in the direction of the normal to the contour (in the direction of the gradient vector).
   - Repeating this operation for each edge pixel will find a set of positions of the possible center points, which can be averaged to determine the exact location of the center



Hough transform for circles

Algorithm for finding circles of a given radius on halftone images

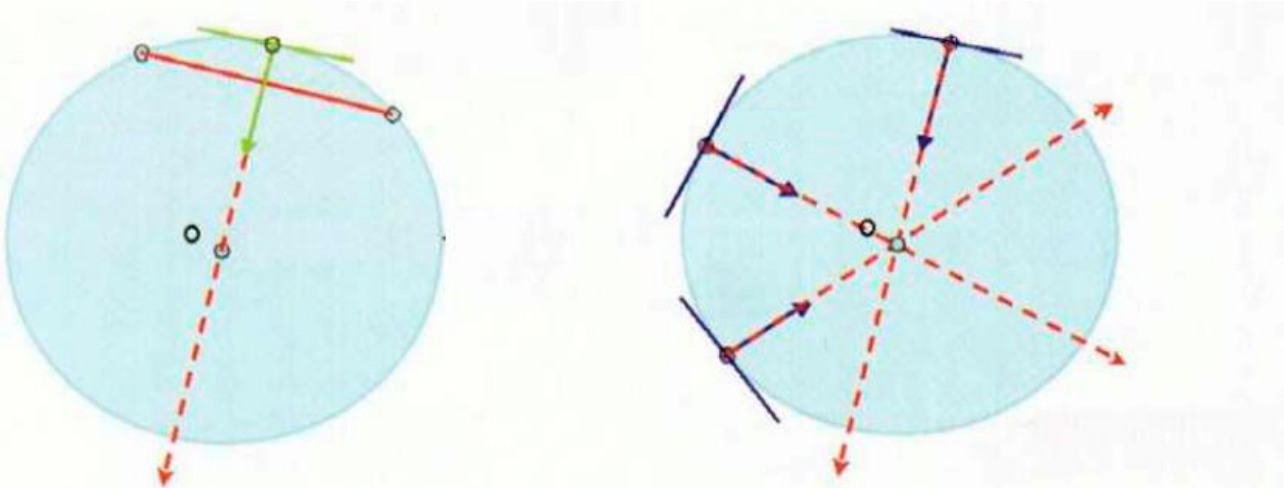
- If the radius of the circle is unknown or is varying, it is necessary to include R as an additional parameter in the parametric accumulator space
- The local maxima search procedure should determine the radius as well as the position of the center by considering changes along the third dimension of the parametric space
- If the size of the detected circle is out of interest and you only want to find its center, then you may not increase the dimension of the parameter space



Hough transform for circles

Algorithm for finding circles of a given radius on halftone images

- For every possible direction to the "center", the contour point votes with a ray in that direction.
- As a result, all possible positions of the "center" will be involved at any scale of the object, which will allow searching for circles regardless of their radius

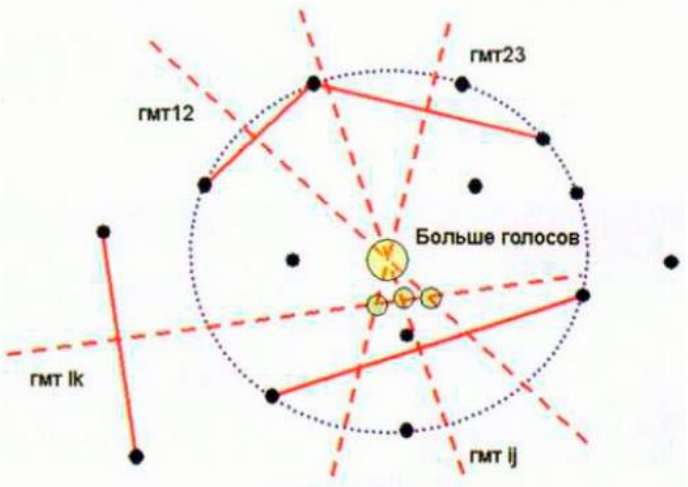


Hough transform for circles

Algorithm for finding circles of a given radius on halftone images

2. After finding the potential centers of the circles, it is necessary to find the radius of the circles with the centers at the found points

In the case of a point set, rather than a continuous contour of a circle, it is possible to implement voting of pairs of points in favor of the corresponding midpoint perpendiculars, and, thus, to solve the problem of identifying circles of unknown size in a binary point set



Hough transform

Analyzing the accumulator function when searching for geometric primitives

1. Direct search for a fixed number of local maxima (one global maximum) in the parameter space. There are various ways to find such maxima

2. Threshold segmentation of the accumulator function and subsequent analysis of connected regions of the parameter space

- If we chose the threshold to be equal to the value of the minimum of local maxima, then the second method will give the same result with the first one

- The problem of the optimal choice of the threshold for a particular image remains

Hough transform

Analyzing the accumulator function when searching for geometric primitives

Short lines (curve segments) will give relatively low peaks in accumulator function if compared to long ones. They will only be detected if it is known to be present in the image prior to setting the threshold

How to avoid thresholding?

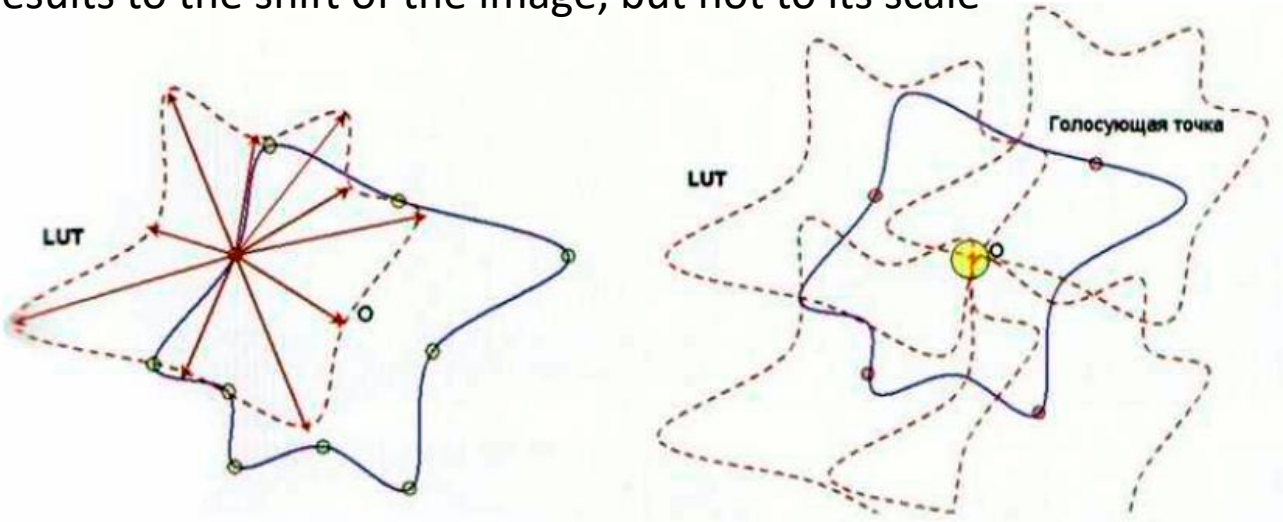
- At each stage of the analysis, we search for single global maximum of the accumulator function
- After finding the global maximum we subtract the income of all source image points that resulted in this line from the parameter space
- Then start searching for the next global maximum

The result: a greater sensitivity to short lines and robustness to noise

Hough transform for an arbitrary shape

Generalized Hough transform (GHT)

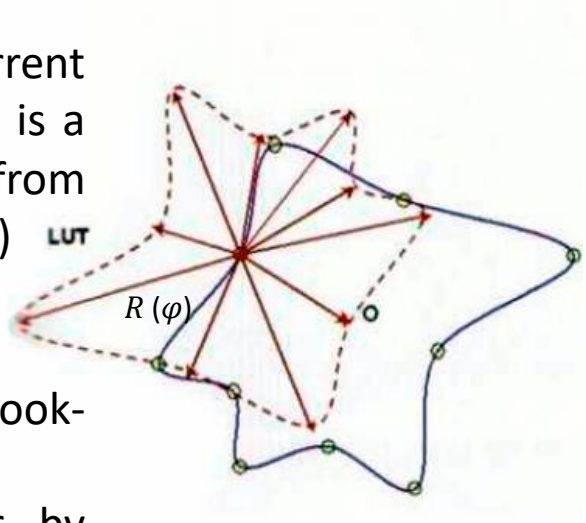
- In 1966, the contour point voting method was generalized to the case of curves that can not be described analytically and was named the Generalized Hough Transform
- First, let us consider the problem of detecting an object of an arbitrary shape, specified by a reference image, in the case when it is required to ensure the invariance of the detection results to the shift of the image, but not to its scale



Hough transform for an arbitrary shape

Generalized Hough transform

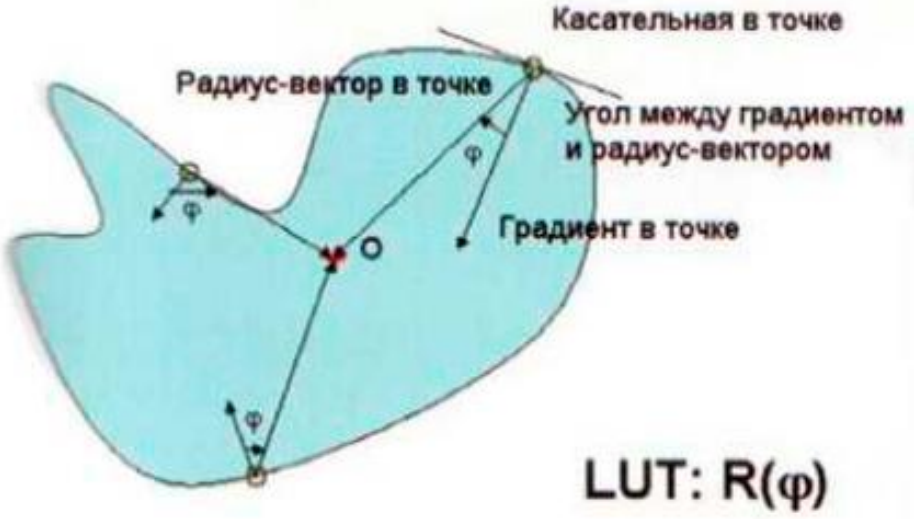
- In this case, it is important that the distance R from the current pixel of the border to its center is no longer a constant but is a function $R(\varphi)$ of the angle φ of the radius vector directed from the contour point to the center (conditional localization point)
- To define simple forms, the function $R(\varphi)$ can be defined analytically
- For complex shapes, the approach remains viable by using a look-up-tables containing discrete $R(\varphi)$ values for different angles
- The algorithm consists of training the Hough detector by composing a LUT from the reference image and the step of detecting an object in the test image by voting of contour points using this LUT



Hough transform for an arbitrary shape

Generalized Hough transform

- The considered approach can be generalized to the case when the object is allowed to rotate, in addition to shifting (invariance to rotation)
- In this case, the radius vector at the edge point is not a function of the absolute angle φ of the direction to the center, but a function of the relative angle between the direction of the gradient and the direction of the radius vector.



Hough transform

Recurrent Hough Transform in Sliding Window (RHT)

- Is used to obtain information about the straight lines passing through each point and their parameters
- Define a window of size $W \times W$ sliding across the image from left to right and bottom to top
- For each window position, the corresponding Hough transform accumulator values are calculated
- Next, the results are transferred to a common accumulator space for the entire image
- As a result, each point of the common accumulator space is characterized by the parameters of the most reliable segment of the straight line passing through it

Hough transform

Recurrent Hough Transform in Sliding Window (RHT)

- To correctly transfer the voting result in each individual window to the common accumulator space, we change the parametrization of the Hough space
- We will describe the straight line in the window with the pair of parameters (x,Q), where x is the coordinate of the intersection of the X-axis of the window, and Q is the angle of inclination of this straight line to the Ox-axis
- After all the points in the window voted, we will get an accumulator, in which the point (x,Q) will contain the number of points lying on the line passing in this window through the point (x,Q) at an angle Q

Hough transform

Recurrent Hough Transform in Sliding Window (RHT)

- To transfer the sliding window results to the final accumulator space, it is necessary to determine the straight line for which the largest number of points voted for each point on the Ox axis of the window
- However, when parametrizing using (x,Q) parameters space, not all lines will be captured, since some of them will cross the Ox axis far beyond the window or not cross at all
- To solve this problem, we can introduce an additional parametrization with (y, Q)

Hough transform

Recurrent Hough Transform in Sliding Window (RHT)

- The pass with parametrization (x, Q) will be called the horizontal pass and we will go when window goes from the lower left position and goes along the horizontal line of the source image to the right position, and then go up one line and slides from left to right again
- The (y, Q) pass goes vertically (by columns), from the lowest left position and goes up the column of the source image to the top position, and then move one column to the right, and so on
- After both passes, the results are combined by maximization, and the segments with the most votes are accumulated in the accumulator space

Hough transform

Recurrent Hough Transform in Sliding Window (RHT)

- As a result, the accumulator would contain all possible variants of the arrangement of elements of straight lines for a given image
- After that, it is necessary to select the most reliable line segments. The following processing options are possible:
  1. threshold cut-off (threshold value is a parameter)
  2. selection of local maxima (parameters - the size of the rectangular area in which the maximum is searched)

Hough transform

Recurrent Hough Transform in Sliding Window (RHT)

Source halftone image

Source image contours

Found lines

Hough transform

Recurrent Hough Transform in Sliding Window (RHT)

Small window size

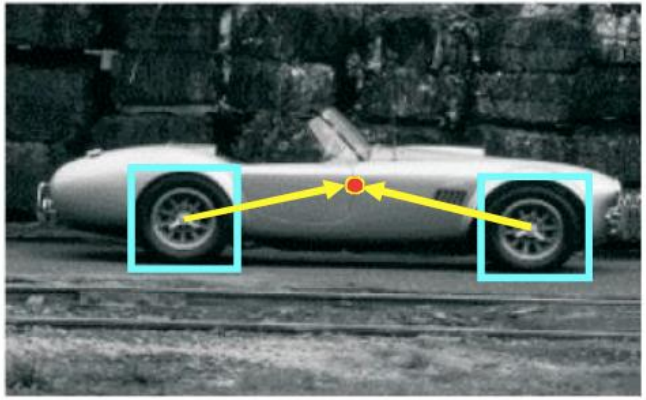
Medium window size

Big window size

Usage example

Use of Hough transform and voting points methods for image recognition

- For example, it is possible to determine the locations of the characteristic fragments of the image relative to the center of the object, as well as the displacement vector of the fragments relative to the center



Usage example

Use of Hough transform and voting points methods for image recognition

- **Task:** Find object center
- **Solution:** Each characteristic element of the image will vote for the centres of objects on the image, and the point with the maximum votes will be the centre of the object

Hough transform

Summary

- Hough transform modifications provide invariant detection of geometric primitives and objects in the image with a high degree of noise immunity and significant accuracy in determining the location and orientation parameters

- The described algorithms do not detect grayscale objects themselves, but their outlines. Objects that do not have a well-defined outline cannot be detected using locus

Image Features

Image Analysis

Marr Paradigm. Image analysis goes from «iconic» data representation to «symbolic» representation

Steps of image processing:

- Image preprocessing
- Primary image segmentation
- Detecting the geometric structure
- Detecting of the relative structure and semantics of the scene

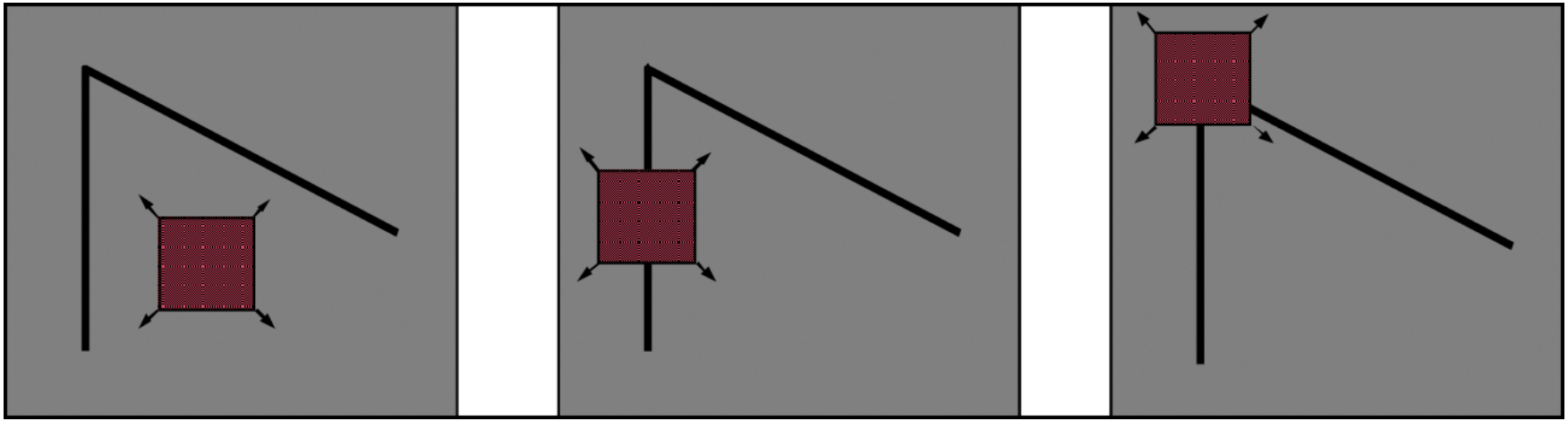
Features

What is the image feature?

Features

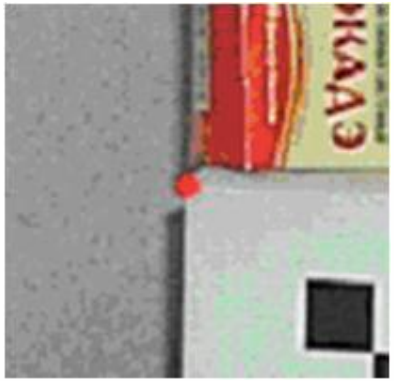
Types of possible features:

«point», «edge», «area», «line», «corner»

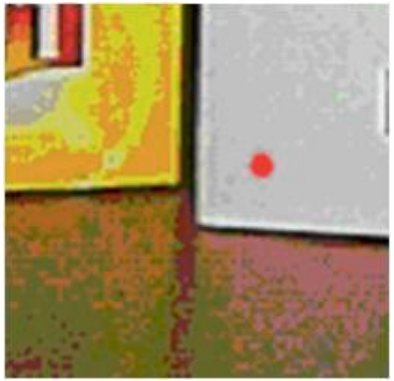


Feature Point

A feature point p of an image I is a point with a characteristic (special) neighborhood.
This point differs from all other points in a certain neighborhood.



Feature point



Nonfeature point

Corner Detectors

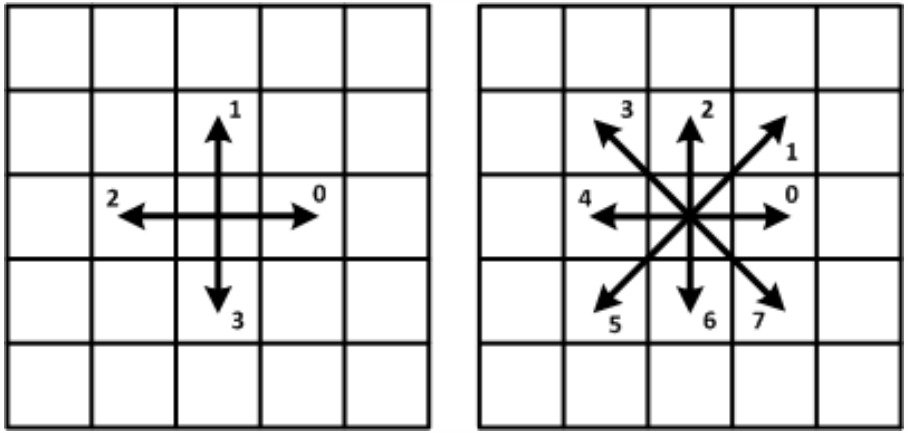
Corners are well repeatable and distinguishable.
The main property of a corner:

- the image gradient has two dominant directions in the area around the corner

Example of feature point detection

Moravec Detector

Idea: to measure the change in pixel intensity $I(x, y)$ by shifting a square window centered in (x, y) to one pixel in each of eight principal directions (2 horizontal, 2 vertical and 4 diagonal).
Window size: 3×3, 5×5 or 9×9 pixels.



Moravec Detector

Algorithm:

1. For each shift direction (u, v), the **change in intensity** is calculated:

$$E_{u,v}(x,y) = \sum_{a,b} \big(I(x+u+a, y+v+b) - I(x+a, y+b)\big)^2$$

2. A **probability map** of finding the corners in each pixel is constructed by calculating the estimated function:

$$C(x,y) = \min\{E_{u,v}(x,y)\}$$

3. Pixels are **cut off** in which the values of the evaluation function are below the threshold value.
4. Duplicate corners are removed by **suppressing non-maximums**.
5. All received **nonzero map elements** correspond to the corners in the image.

Sigma Filter

$$h(x, y) = \mathbf{H} * f(x, y),$$

where $f(x, y)$ – source image,
H – operator of feature detection,
* – operation of applying the operator.

- For each pixel (x, y) the mask *R* of the size $N \times N$ is considered:

$$m = \frac{1}{N^2} \sum_R h(x, y),$$

$$\sigma^2 = \frac{1}{N^2} \sum_R h^2(x, y) - m^2.$$

- The point (x, y) **is feature** if $h(x, y)$ not in a range:

$$m - \alpha\sigma > h(x, y) > m + \alpha\sigma,$$

where α – parameter defines the range, m – mean value,
σ – standard deviation of feature $h(x, y)$.

Sigma Filter

Source image

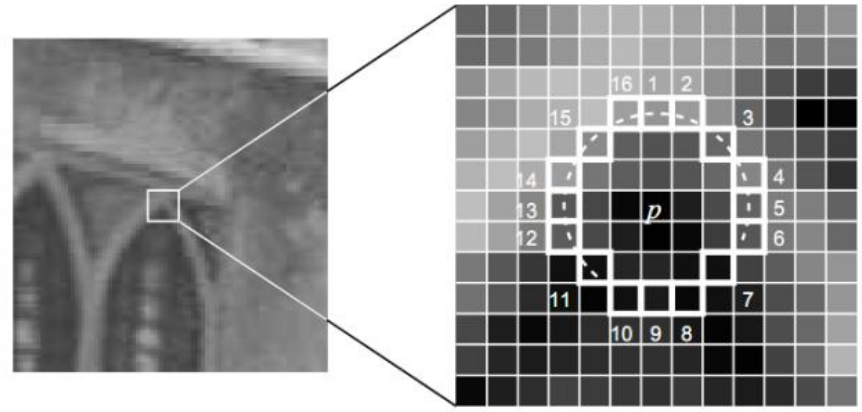
(*N* = 5)

Feature points
(*N* = 5, α = 1)

FAST Detector

FAST Detector – Features from Accelerated Segment Test

For each pixel P of the image a neighbouring Bresenham circle around this pixel is considered. In case if 7x7 kernel is used, then it would contain 16 pixels.



Window of a pixel when using a FAST detector
Bresenham circle of radius 4

FAST Detector

Each of the neighbouring Bresenham circle pixel can be in one of three possible states:

$$S = \begin{cases} d, I_x \leq I_p - t & (darker) \\ s, I_p - t < I_x < I_p + t & (similar) \\ b, I_p + t \leq I_x & (brighter) \end{cases}$$

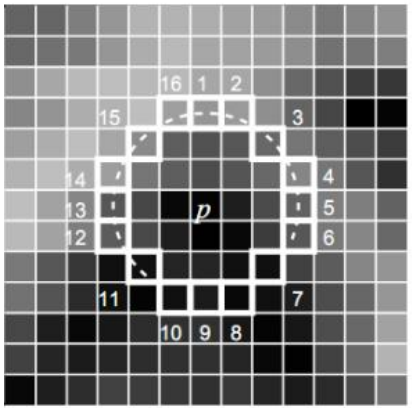
where t – threshold,
$I_p - P$ pixel intensity.

A pixel is considered to be a corner point if it has N adjacent pixels on the circle whose intensities satisfy the state condition d or b.

FAST Detector

In case of detector of radius 4 (7x7 kernel size) we have 16 detector pixels:

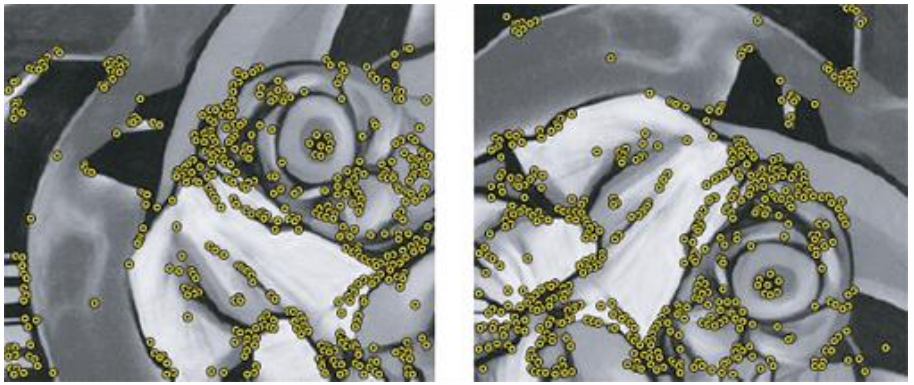
$$x \in \{1,2,\dots,16\}$$



Possible optimization:

- It is necessary to compare the intensities at the vertical and horizontal points on the circle numbered 1, 5, 9 and 13 with the intensity at the point P, I_p
- If the state condition is satisfied for three of these points, then a full test is performed for all 16 points
- The smallest value of N, at which singular points begin to be detected stably, is 9

FAST Detector



FAST detector results

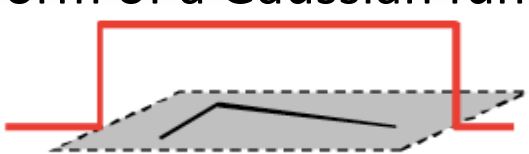
Harris Detector

- Let the variation in image intensities depend on the shift (u, v). Let us consider some some rectangular area. Then the variation inside this area can be evaluated according to the formula:

$$E(u,v) = \sum w(x,y)\big(I(x+u, y+v) - I(x,y)\big)^2,$$

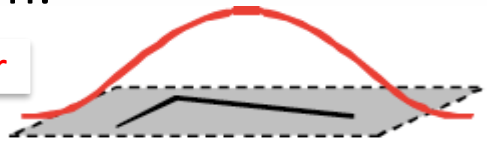
where $I(x,y)$ – intensity in a point (x,y),
$w(x,y)$ – window function,
$I(x+u, y+v)$ – shift of intensity.

- The window function $w(x,y)$ can be defined either in a binary form or in the form of a Gaussian function:



1 in a window, 0 – out

or



Gaussian

Harris Detector

- By expanding the difference in intensities squared into a second-order Taylor series at the point (x, y) (with bilinear interpolation) for small shifts, we obtain the following approximation for variation:

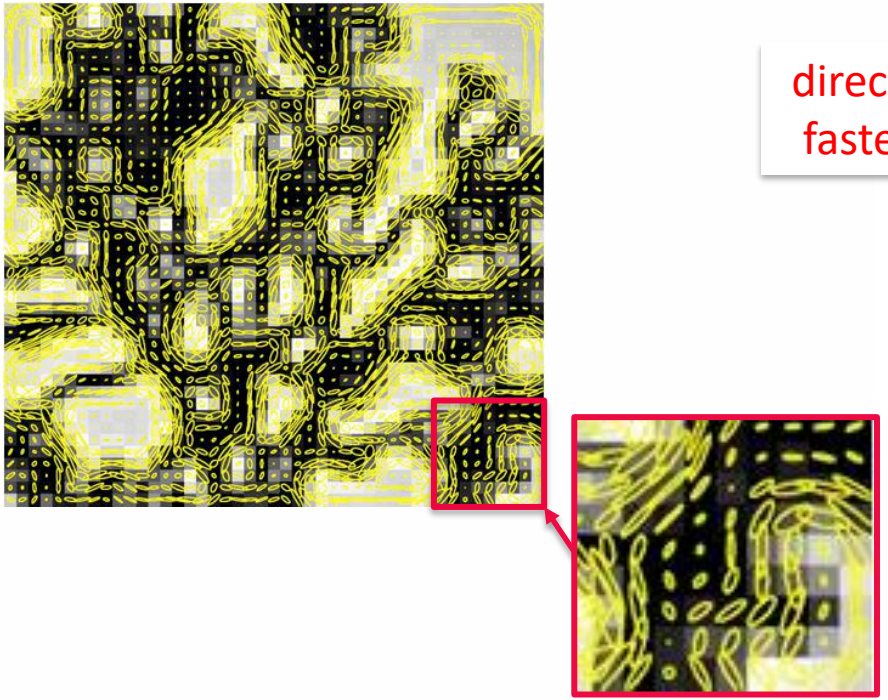
$$E(u, v) \approx [u \quad v]\mathbf{M}\begin{bmatrix} u \\ v \end{bmatrix},$$

$$\mathbf{M} = \sum_{x,y} w(x, y)\begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}.$$

where $\mathbf{M}$ is a matrix consisting of weighted values of the derivative of the intensity function, while I_x and I_y are are image derivatives in x and y directions.

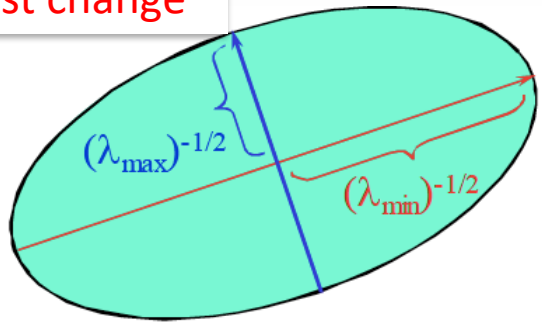
Harris Detector

- In a geometric interpretation, the matrix $\mathbf{M}$ represents an *ellipsoid* in which the axis lengths are determined by **eigenvalues**, and the orientation is determined by the orthogonal matrix of **eigenvectors**.



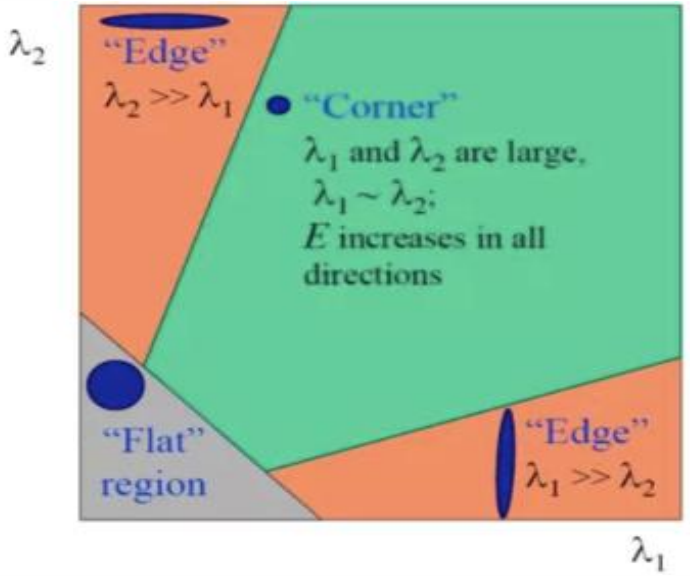
direction of the fastest change

direction of the slowest change



Harris Detector

- If one of the eigenvalues is significantly larger than the other, then an **edge** is found.
- If both eigenvalues are small, then a **“flat” uniformly bright region** is found.
- If both eigenvalues are large and the same order with each other, then a **corner point** in the center of the win is found.

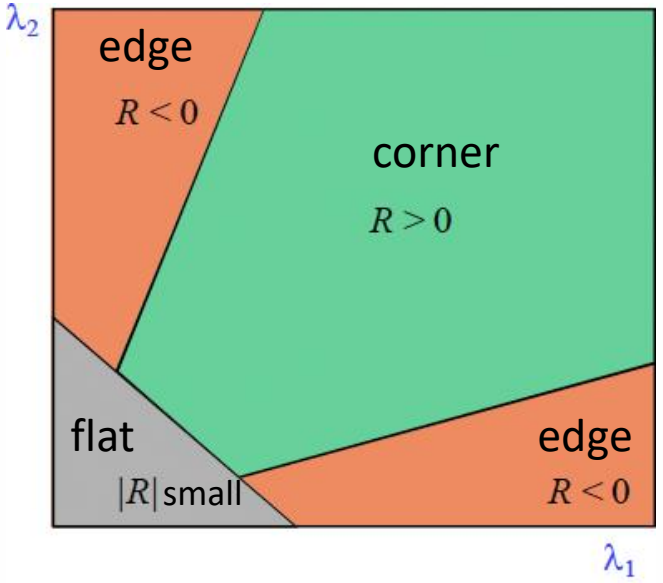


Harris Detector

- Harris introduced a measure of the corner intensity:

$$\mathbf{R} = \det\mathbf{M} - k(\mathrm{tr}\mathbf{M})^2,$$

where $\det\mathbf{M} = \lambda_1\lambda_2$;
$\mathrm{tr}\mathbf{M} = \lambda_1 + \lambda_2$;
k – an empirically selected parameter with values of 0.04–0.06



- if $\mathbf{R} > 0 \Rightarrow$ «corner»
- if $\mathbf{R} < 0 \Rightarrow$ «edge»
- if $\mathbf{R} = 0 \Rightarrow$ «flat» region

Harris Detector

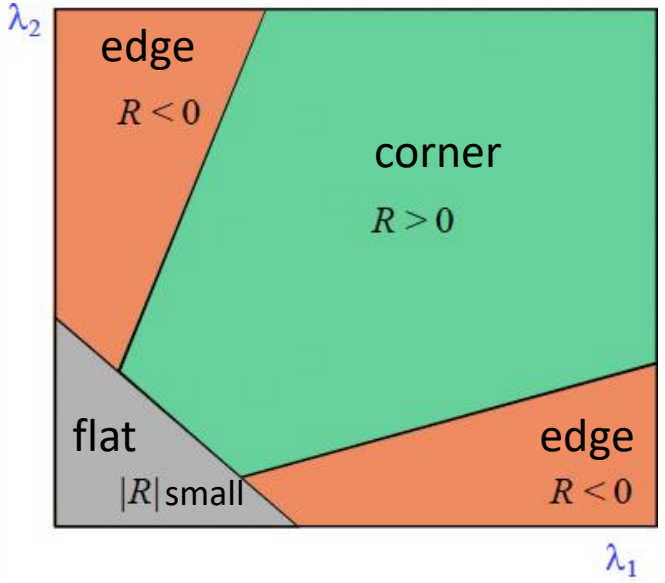
ITMO

Algorithm:

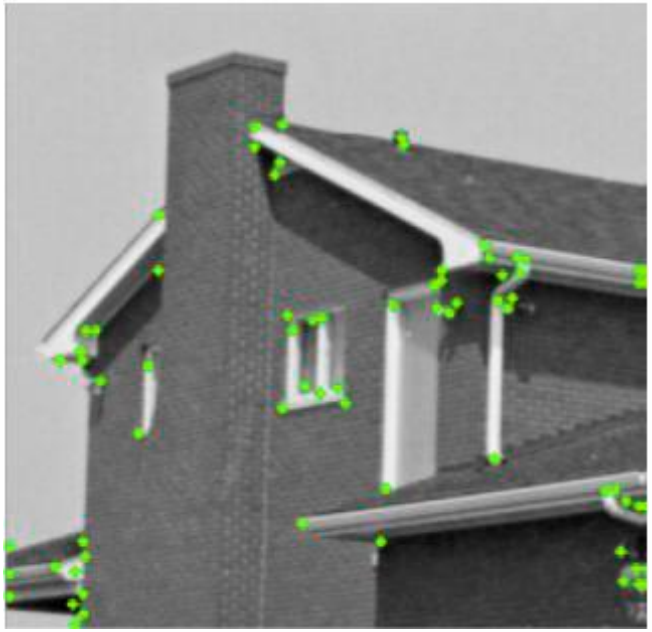
1. Calculate the gradient of the image in each pixel.
2. Calculate the matrix $\mathbf{M}$ from the window around each pixel.
3. Calculate the response of the corner $\mathbf{R}$.
4. Cut off weak corners at threshold $\mathbf{R}$.
5. Find the local maxima of the response function in a neighborhood of a given radius.
6. Choose the $\boldsymbol{N}$ strongest local maxima.

Förstner Detector

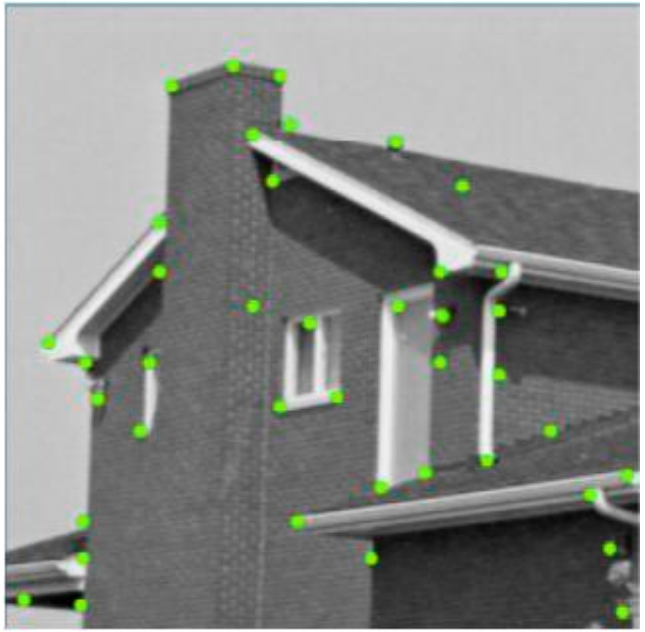
$$\mathbf{R} = \frac{\det \mathbf{M}}{\mathrm{tr}\mathbf{M}}$$



Comparison of Förstner and Harris



Harris



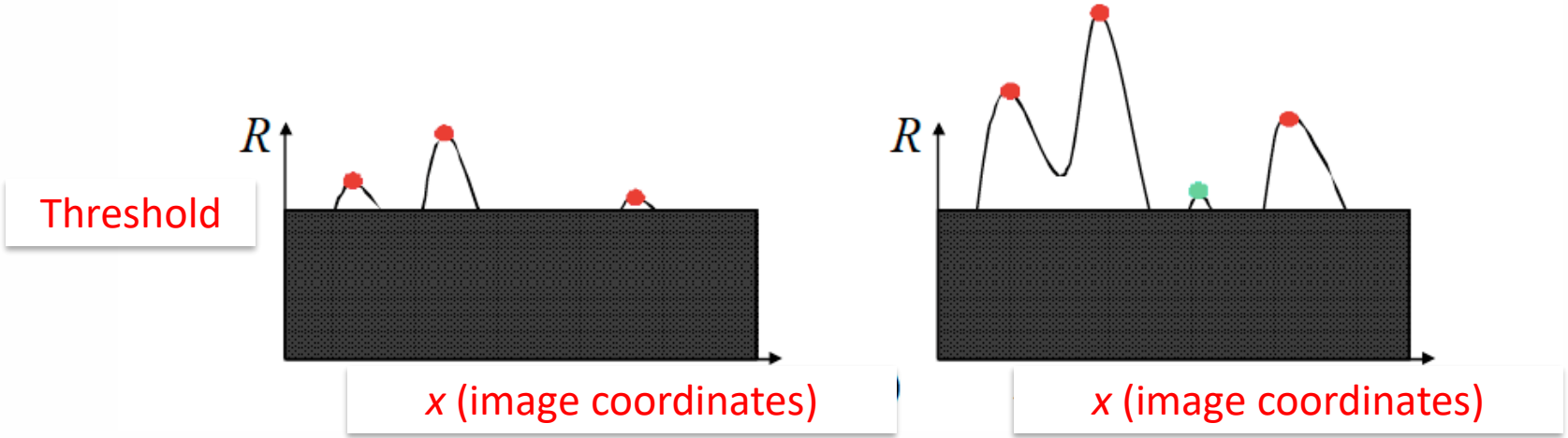
Förstner

Detectors Invariance

- **Invariance Property** – if we apply a geometric (affine) or photometric (affine intensity change $I \rightarrow aI + b$) transformation to the image, then the detector must find the same set of points.

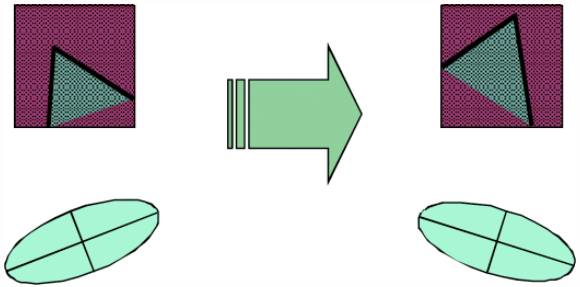
Detectors Invariance

1. Light invariance:
   - To light shifting ($I \rightarrow I + b$);
   - To light scaling ($I \rightarrow aI$).

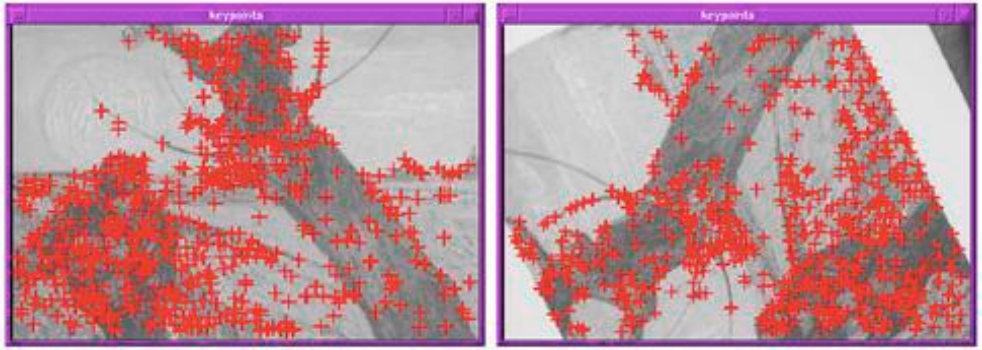


Detectors Invariance

2. Invariance to rotation:



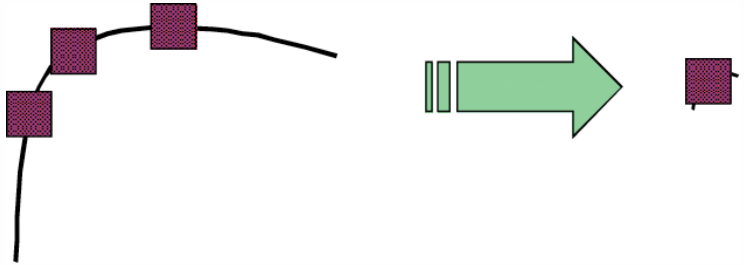
- The ellipse rotates, but its shape (eigenvalues) remains unchanged.



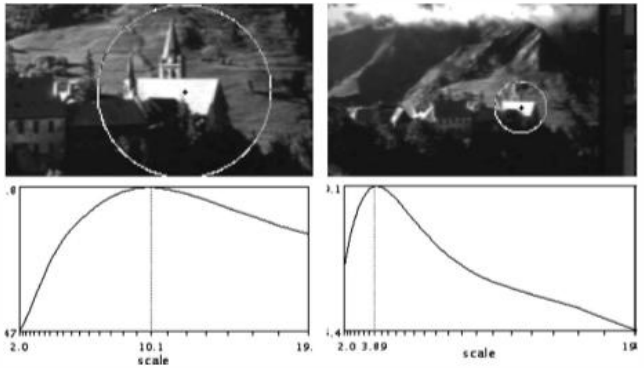
Harris detector on changed and rotated images

Detectors Invariance

3. Invariance to scaling:



- The marked points on the left will be the edges, and on the right the corner.
- It is necessary to determine the size of the neighborhood of a feature point in scaled versions of the same image.

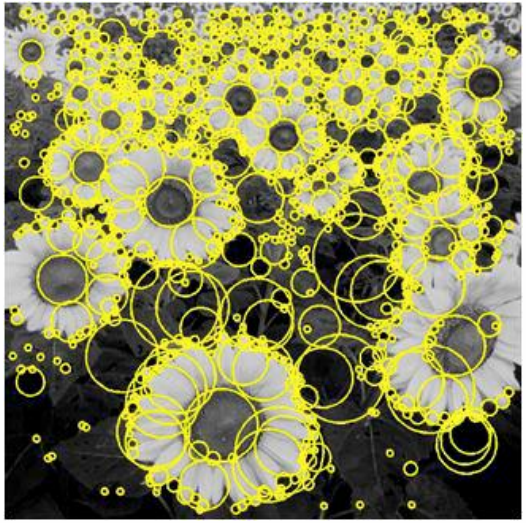


Different scale
of the neighborhood

Blob Detectors

LoG (Laplacian of Gaussian) Detector

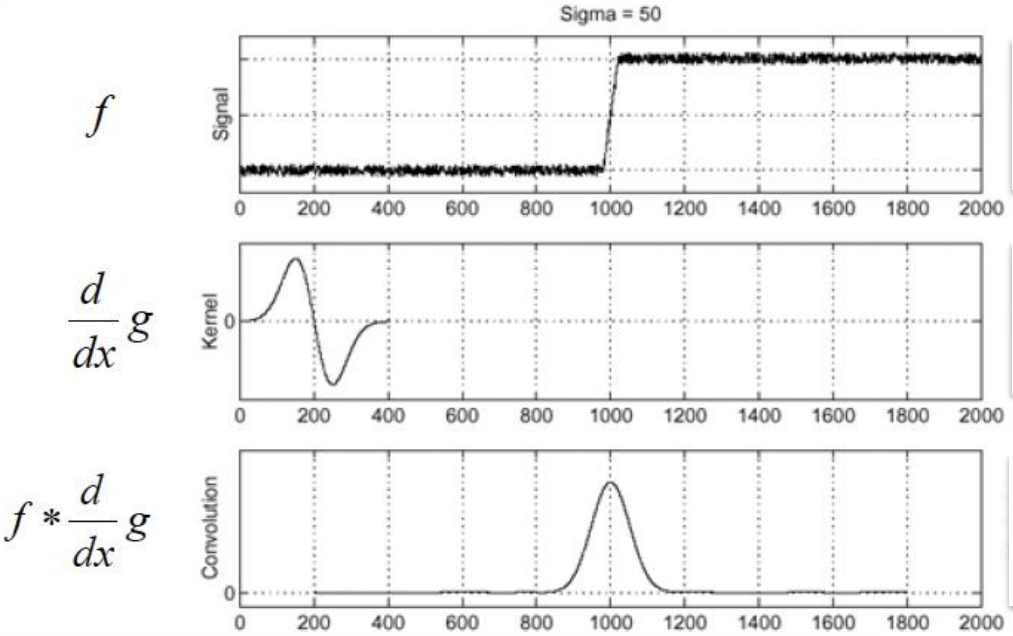
- Blobs are blob-shaped groups of connected pixels that share a common property and are different from surrounding regions
- Blob centers are feature points
- Blobs are described by 4 parameters
  - center coordinates (x, y)
  - scale
  - direction



Example of blob detection

LoG (Laplacian of Gaussian)

Gaussian is a Gauss function which are used to blur images.

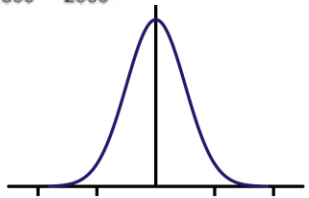


Edge

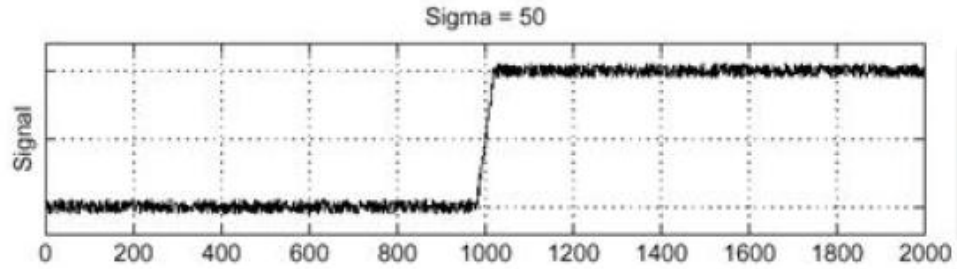
Derivative of Gaussian

Edge = maximum of derivative

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{x^2}{2\sigma^2}}$$

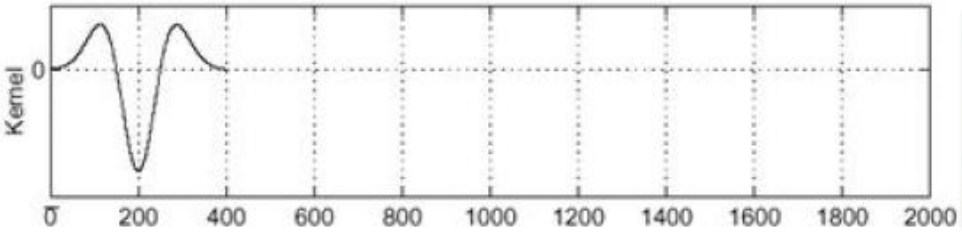


LoG (Laplacian of Gaussian)



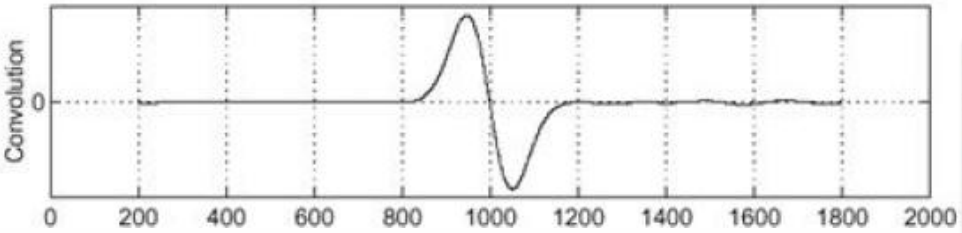
f

Edge



$$\frac{d^2}{dx^2} g$$

Second derivative of Gaussian (Laplacian)

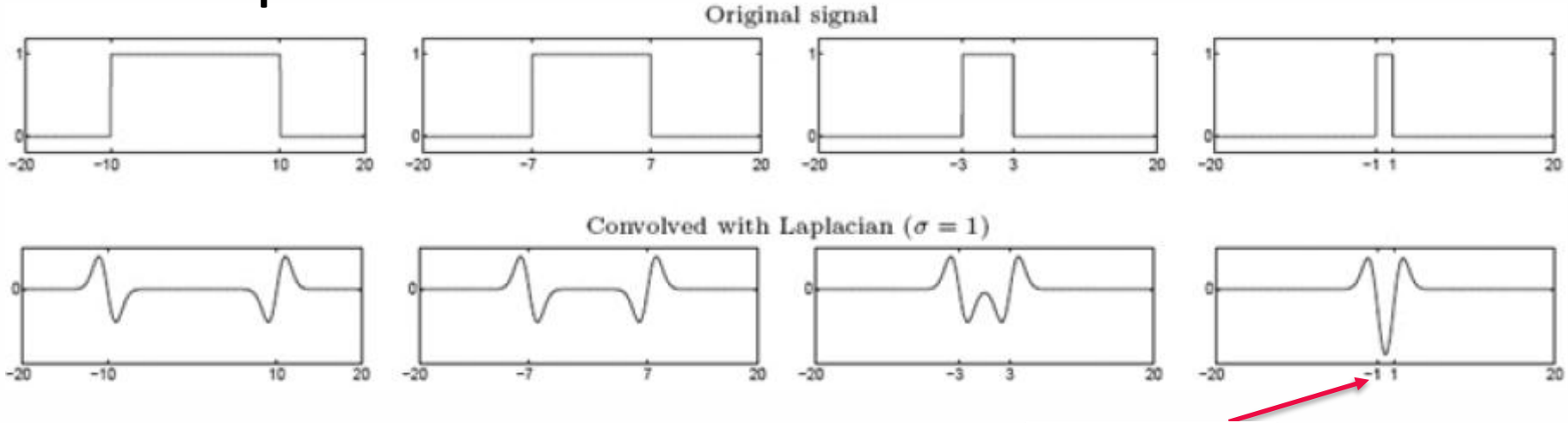


$$f * \frac{d^2}{dx^2} g$$

Edge = zero transition of the second derivative

LoG (Laplacian of Gaussian)

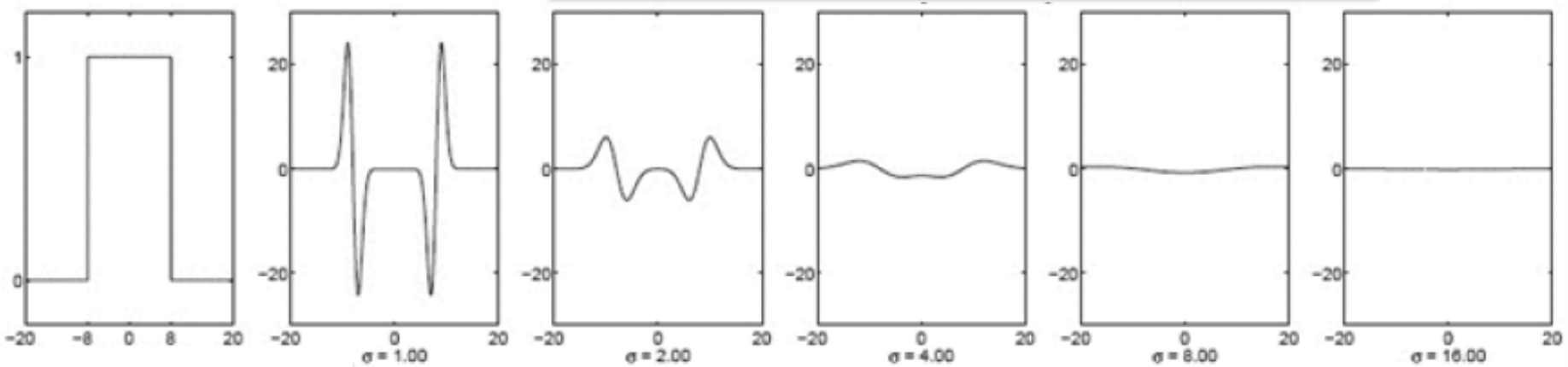
- «Edge» is a burst of function
- «Blob» is a combination of two bursts
- The magnitude of the Laplacian of Gaussian response reaches a maximum in the center of the blob **if the size** of the Laplacian "**corresponds"** to the size of the blob



LoG (Laplacian of Gaussian)

- Search for blob characteristic size: convolution with Laplacian at several scales and search for maximum response
- The response of the Laplacian fades if zooming in (changing scale)

Non normalized Laplacian response

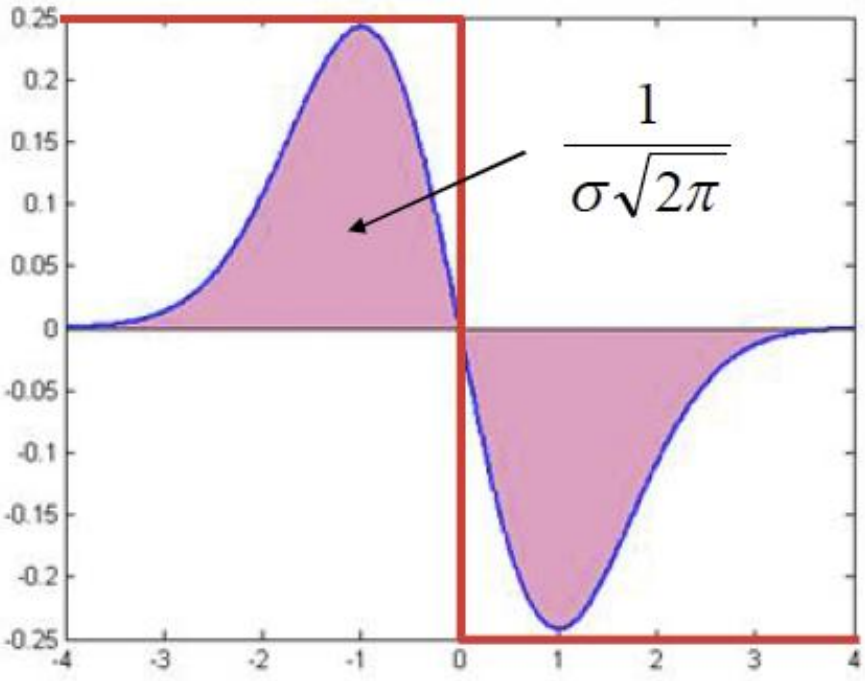


Source signal (radius = 8)

Increasing σ

LoG (Laplacian of Gaussian)

- To achieve scale invariance of the response function, it is necessary to multiply the first derivative by σ, and the Laplacian by σ^2.

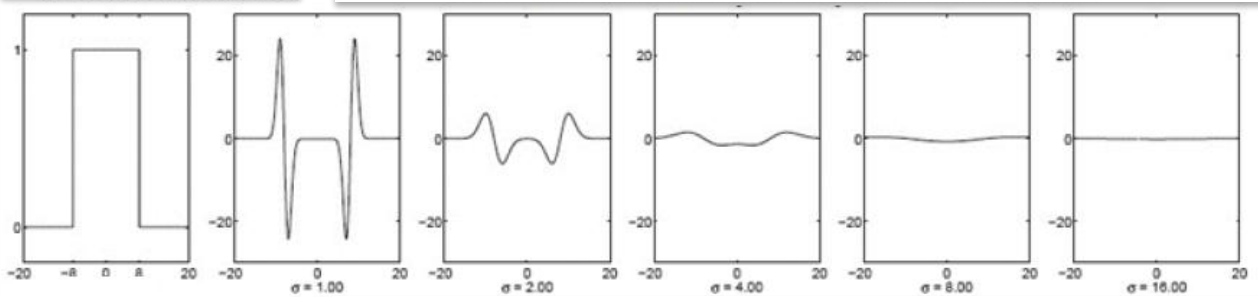


$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{x^2}{2\sigma^2}}$$

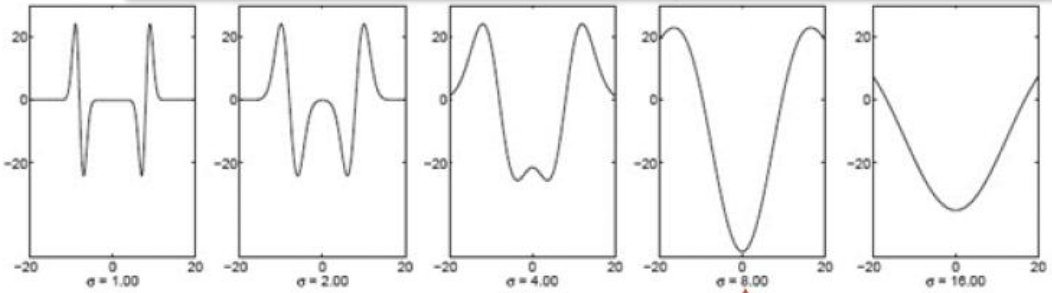
LoG (Laplacian of Gaussian)

Source signal (radius = 8)

Non normalized Laplacian response



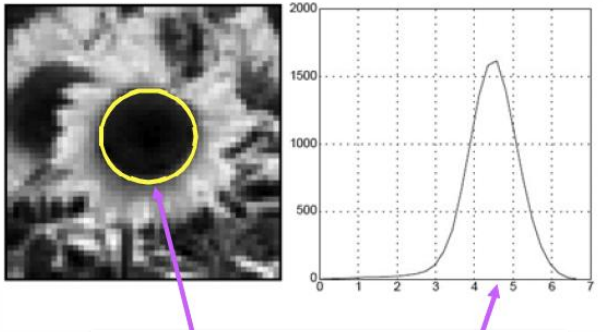
Normalized by scale Laplacian response

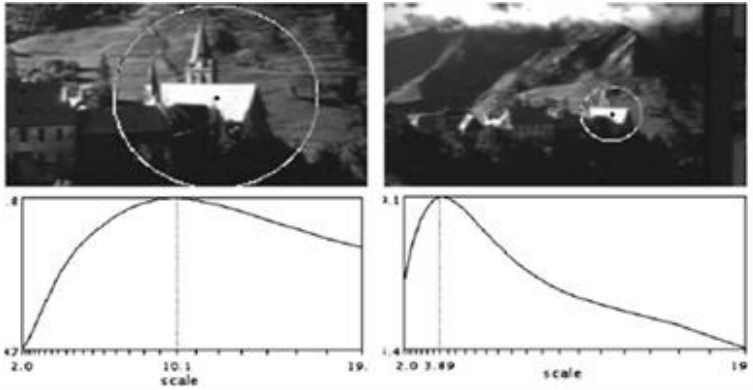


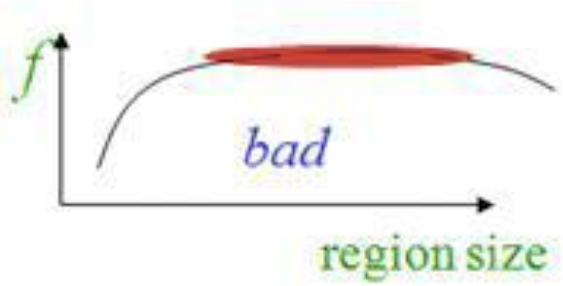
Maximum

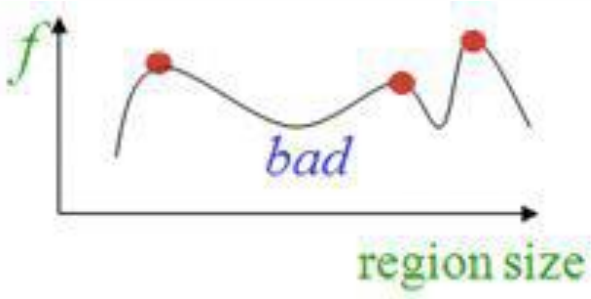
LoG (Laplacian of Gaussian)

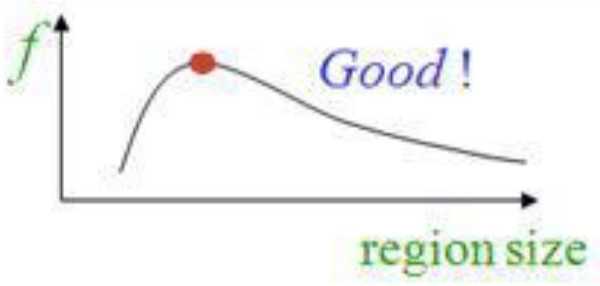
- The characteristic size is defined as the scale at which the maximum Laplacian response is reached:





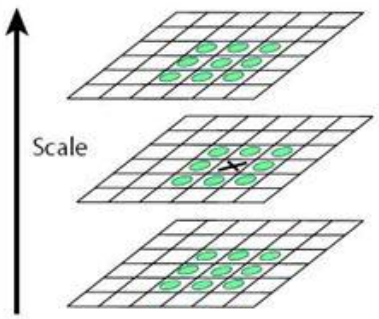


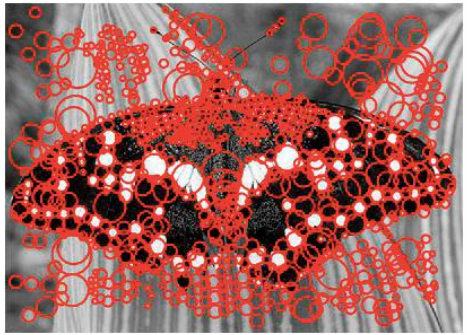




LoG (Laplacian of Gaussian)

- Multiscale blob detector: convolution of a blurry image using the normalized Laplace filter at several scales and the choice of scale with the maximum Laplacian response.

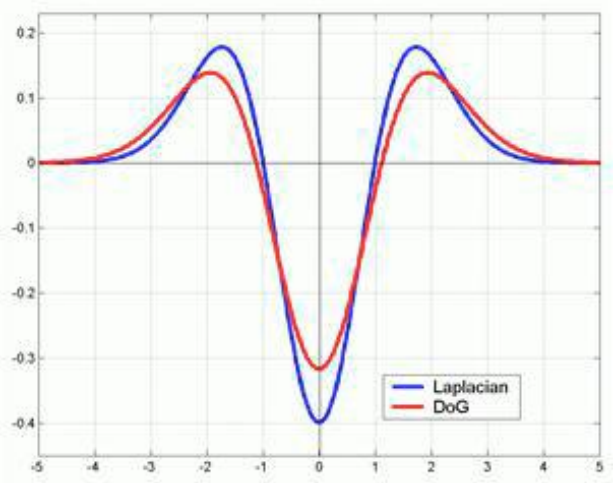


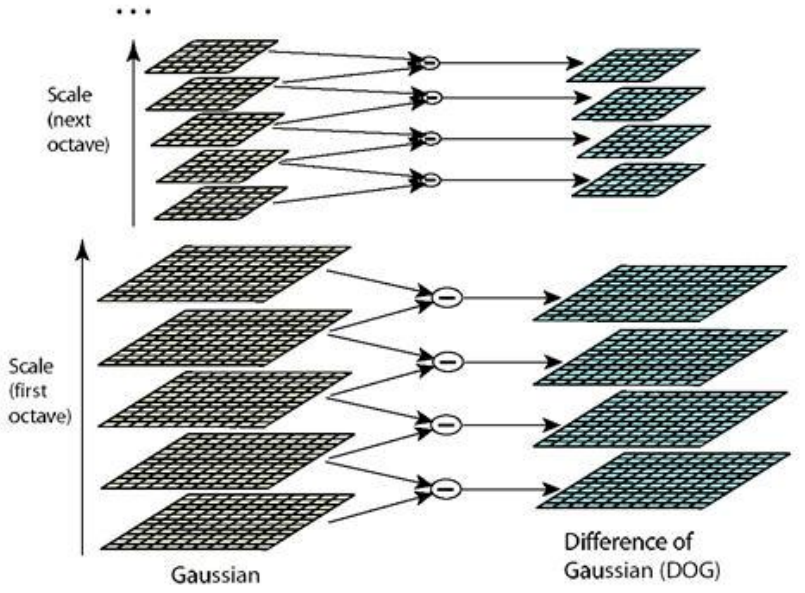


DoG (Difference of Gaussians)

- Method for the approximate calculation of the Laplacian of a Gaussian: the search for the difference of two Gaussians with different scales:

$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

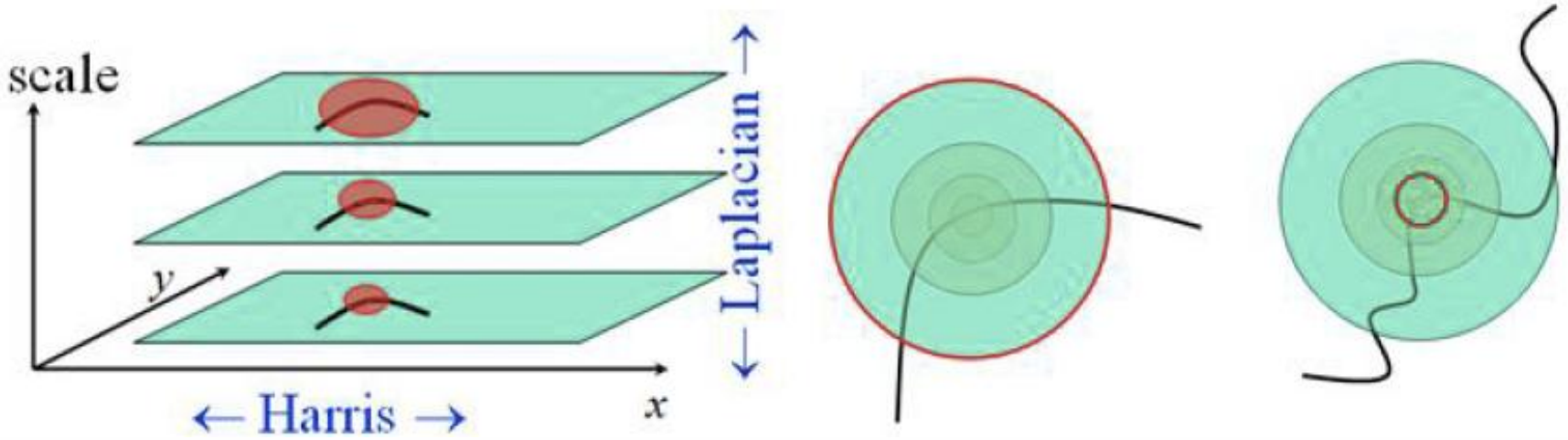




DoG (Difference of Gaussians)

Harris-Laplace Detector

- Selecting the corners in the image, considering the characteristic size: looking for points that maximize the response of the Harris corner in the image and the Laplacian response in scale



Harris-Laplace Detector

In addition to presented detectors, there are many other: Shi-Tomasi, SUSAN, Trajkovic, CSS, CPDA, Harris-Affine, EBR, Hessian, Hessian-Laplace/Affine, Salient Regions, Superpixels, SURF, FAST-9, FAST-12, FAST-ER, ORB, etc.

Test

Lecture 3-4 Test

Please scan the code to start the test

THANK YOU
FOR YOUR TIME!

ITMO more than a UNIVERSITY

Andrei Zhdanov
adzhdanov@itmo.ru