Computer Vision - Image Features

Włodzimierz Kasprzak and Artur Wilkowski

26 November 2021

Features

- Feature: “an interesting part of an image”
- Main Goal: Make compute vision problems tractable by reducing full-size image to a limited set of features
- Two subproblems: Feature detection and feature description

Feature detection

We will discuss:

- Corner detection
- Scale space features
- Geometric primitives

Feature detection: Corner features

Moravec corner detector

Moravec operator (1977)

The detector is based on the auto-correlation function of the signal:

- It measures the grey value differences between a window and windows shifted in several directions;
- Four discrete shifts in directions parallel to the rows and columns of the image are used.

Moravec corner detector

PROCEDURE outline:

1. FOR every pixel (x, y) compute average differences of the image function along four directions:

$$V_0 = \frac{1}{20} \sum_{i=-2}^{2} \sum_{j=-2}^{1} [f(x+i, y+j) - f(x+i, y+j+1)]^2$$

$$V_1 = \frac{1}{20} \sum_{i=-2}^{1} \sum_{j=-2}^{2} [f(x+i, y+j) - f(x+i+1, y+j)]^2$$

$$V_2 = \frac{1}{16} \sum_{i=-2}^{1} \sum_{j=-2}^{1} [f(x+i, y+j) - f(x+i+1, y+j+1)]^2$$

$$V_3 = \frac{1}{16} \sum_{i=-2}^{1} \sum_{j=-2}^{1} [f(x+i, y+j) - f(x+i+1, y+j-1)]^2$$

2. Then for every pixel (x,y) compute

$$M = \min\{V_0, V_1, V_2, V_3\}$$

3. if $M > \theta$ set pixel (x,y) as a keypoint

Harris-Stephens corner detector

Harris-Stephens detector improves on Moravec detector by considering every not only discrete 45deg. corner rotations

PROCEDURE outline:

FOR every pixel (x, y):

1. Create and auto-correlation matrix $A(x, y)$ that averages discrete derivatives of the image in a window $W(x, y)$ around a point $P(x, y)$:

$$A(x,y) = \sum_{(x_k,y_k)\in W} w(x-x_k, y-y_k) \begin{pmatrix} I_x^2(x_k,y_k) & I_x(x_k,y_k)I_y(x_k,y_k) \\ I_x(x_k,y_k)I_y(x_k,y_k) & I_y^2(x_k,y_k) \end{pmatrix}$$

with w being the windowing function (guassian for best results)

Harris-Stephens corner detector

PROCEDURE outline:

Perform Gaussian smoothing of the image then FOR every pixel (x, y):

1. Create and auto-correlation matrix $A(x,y)$ that averages discrete derivatives of the image in a window $W(x,y)$ around a point $P(x,y)$:

$$A(x,y) = \sum_{(x_k,y_k)\in W} w(x-x_k, y-y_k) \begin{pmatrix} I_x^2(x_k,y_k) & I_x(x_k,y_k)I_y(x_k,y_k) \\ I_x(x_k,y_k)I_y(x_k,y_k) & I_y^2(x_k,y_k) \end{pmatrix}$$

2. Eigenvalues λ_1, λ_2 of A are to be considered (not necessarily explicitly evaluated)
3. Corner response (using constant k is)

$$R = \lambda_1\lambda_2 - k(\lambda_1+\lambda_2)^2 = det(A) - k(trace(A))^2$$

and k is a small constant

4. Corners are selected after application of non-max suppression procedure

Shi-Thomasi GFT

Shi-Thomasi Good Features to Track algorithm is essentially a Harris corner detector with a modified corner response function yielding better results

$$R = \min(\lambda_1, \lambda_2)$$

Multi-scale features

Scale problem of feature detection

- Problem: features in the image can have arbitrary size - how to find the best scale for a feature
- Solution: perform search both in space and scale (scale-space search)

Scale space representation

Laptev Lindberg classic scale-space representation

- Scalespace is obtained by convolution of the original image with a Gaussian kernel for different values of σ

$$L(\cdot, s) = g(\cdot, s) * f$$

- The Gaussian kernel is given by

$$g(x, y, s) = \frac{1}{2\pi s} e^{-(x^2+y^2)/(2s)}$$

- In order to detect features we use derivatives of the scale space L

$$L_{x^\alpha, y^\alpha}(\cdot, t) = \partial_{x^\alpha, y^\alpha} L(\cdot, t) = g_{x^\alpha, y^\alpha}(\cdot, t) * f$$

- In order for the derivatives to be comparable between scales, they are normalized

$$\partial_\xi = \sqrt{t}\partial_x, \text{ or} \partial_{x,\gamma-norm} = t^{\gamma/2}\partial_x$$

Scale space representation

- Detection of blobs is achieved by looking up scale space maxima for scaled Laplacian-of-Gaussians

$$\Delta^2 L = t(L_{xx} + L_{yy})$$

- Detection of elongated ridges is achived by:
  - Computation of the Hessian matrix

$$H = s^\gamma \begin{pmatrix} L_{xx} & L_{xy} \\ L_{xy} & L_{yy} \end{pmatrix}$$

  - Computation of the eigenvalues of H

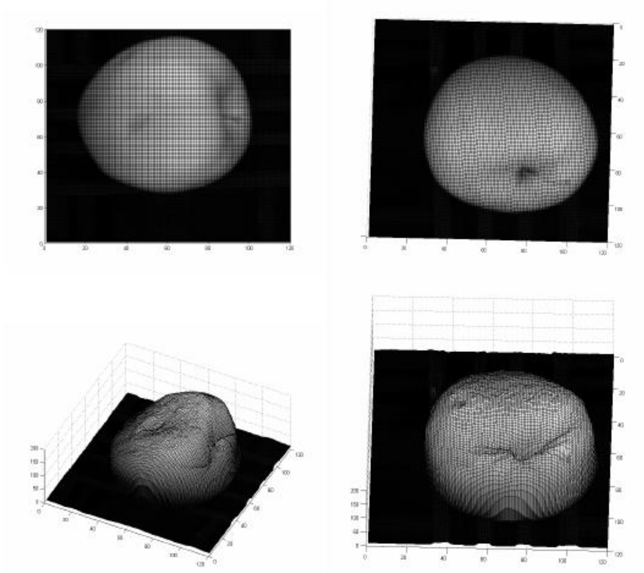
$$L_{pp,\gamma-norm} = \frac{s^\gamma}{2}\left(L_{xx} + L_{yy} - \sqrt{(L_{xx} - L_{yy})^2 + 4L_{xy}^2}\right)$$
$$L_{qq,\gamma-norm} = \frac{s^\gamma}{2}\left(L_{xx} + L_{yy} + \sqrt{(L_{xx} - L_{yy})^2 + 4L_{xy}^2}\right)$$

  - Comparison of the eigenvalues

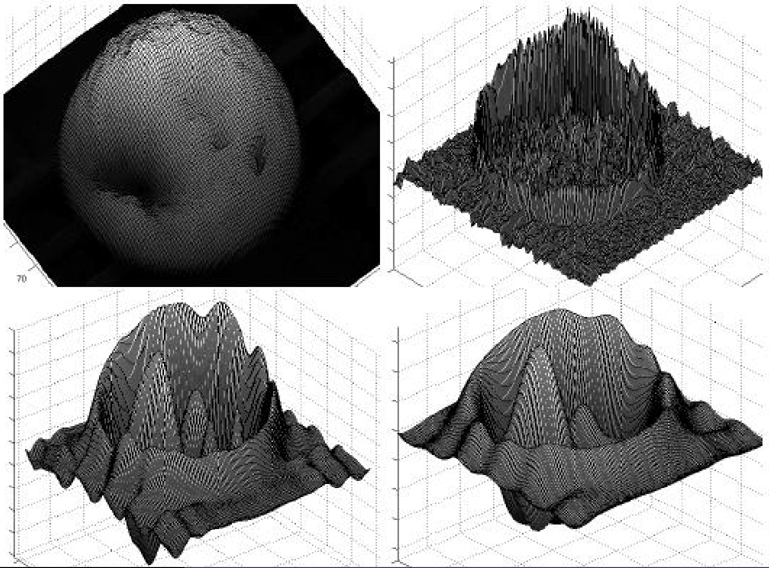
$$\mathcal{A}_{\gamma-norm}L = (L_{pp,\gamma-norm} - L_{qq,\gamma-norm})^2$$

  - Selection of scale-space extrema according to $\mathcal{A}$

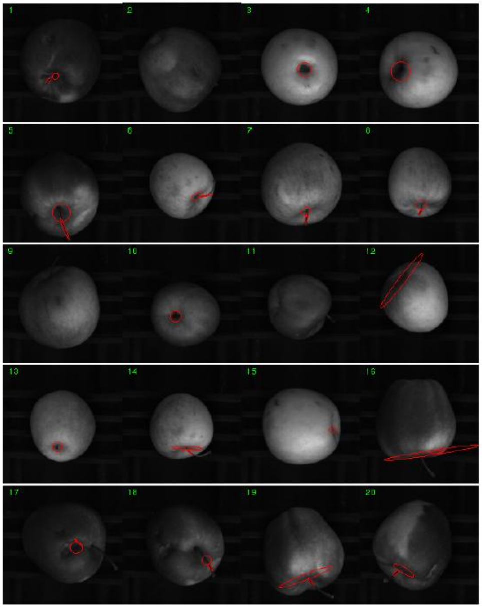
Scale space representation



Scale space representation



Scale space representation



Scale-Invariant Feature Transform

Scale-Invariant Feature Transform

- Detection of optimum keypoints in scale-space
- Computed features have different sizes
- Detected features have are described by a triplet $(\mathbf{x_R}, s, \alpha)$

Scale-Invariant Feature Transform

PROCEDURE outline

1. Compute scale-space be convolving image with Gaussian kernels and subsampling between octaves
   - Image convolution

     $$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

     with a kernel

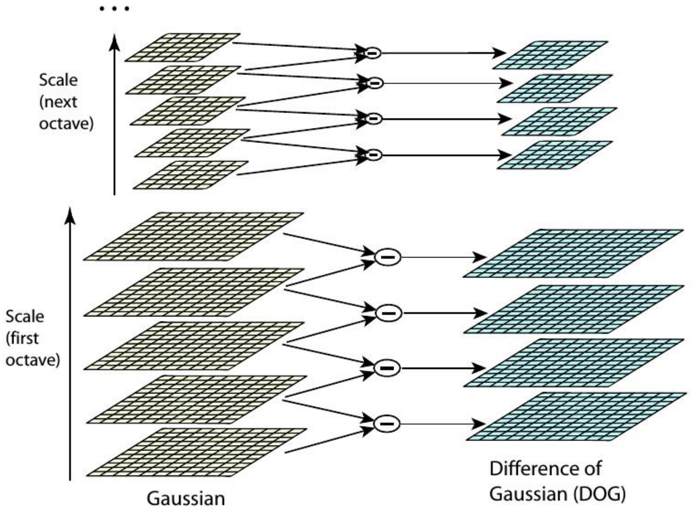
     $$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

   - Computation of the Difference of Gaussians

     $$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned}$$

   - The DoG is close approximation to the LoG

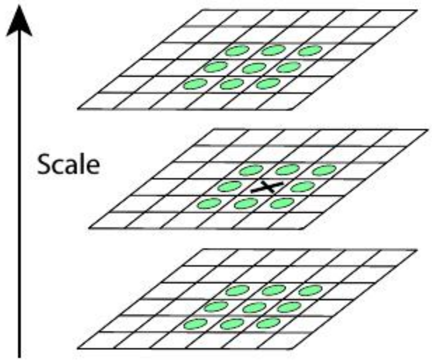
Scale-Invariant Feature Transform



Source: David. G. Lowe, Distinctive image features from scale-invariant keypoints, 2004

Scale-Invariant Feature Transform

2. Perform non-extrema suppression using neighbours from the current image and from one scale up and down
   - Image convolution



Source: David. G. Lowe, Distinctive image features from scale-invariant keypoints, 2004

Scale-Invariant Feature Transform

3 Accurate keypoint localization. Sub-pixel approximation of the local extremum is obtained by applying the Taylor series expansion (up to quadratic terms)

$$D(\mathbf{x}) = D + \frac{dD^T}{d\mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{d^2 D}{d\mathbf{x}^2}\mathbf{x}$$

And the optimum is given by

$$\hat{\mathbf{x}} = \frac{d^2 D^{-1}}{d\mathbf{x}^2} \frac{dD}{d\mathbf{x}}$$

Scale-Invariant Feature Transform

4. Weak extrema are eliminated by thresholding $|D(\hat{\mathbf{x}})| < 0.03$ (D is normalized within $[0, 1]$)
5. Edges are eliminated from the set of detected extrema by analyzing eigenvalues of Hessian Matrix computed as

$$H = \begin{pmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{pmatrix}$$

   Extrema with a small one eigenvalue and a large second eigenvalue are rejected as edges. As in Harris-Sephens operator, the eigenvalues do not need to be explicitly computed.

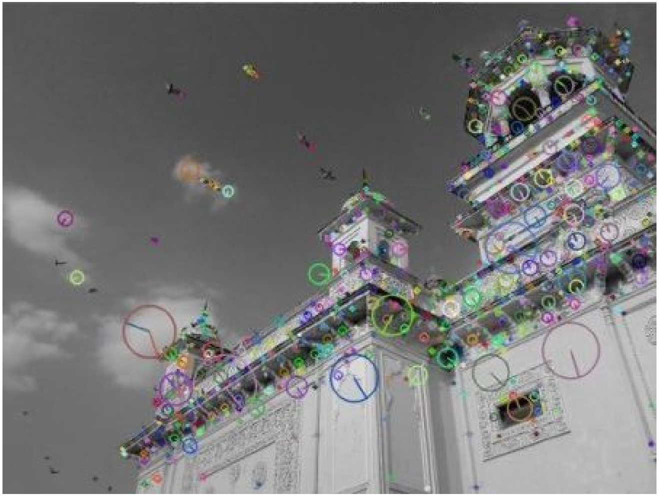
Scale-Invariant Feature Transform

6. Gradient directions and magnitudes are computed

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$
$$\theta(x,y) = atan2(L(x,y+1) - L(x,y-1), L(x+1,y) - L(x-1,y))$$

7. Establishing keypoint orientation by the gradient directions and magnitudes in keypoint neighbourhood.
   - 36 orientation bins is used
   - each gradient votes with its magnitude smoothed by the Gaussian kernel
   - the highest histogram peak together with all peaks above 80% of the highest one are selected
   - in effect: a keypoint can be multiplicated with different orientations

Scale-Invariant Feature Transform



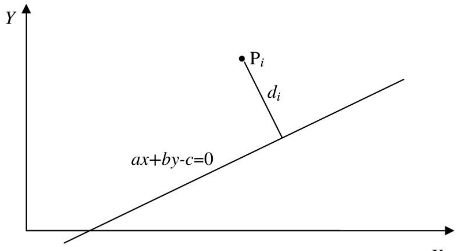
Source: `https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html`

Feature detection: Geometric primitives

Approximating contour with a polyline

- The Douglas-Peucker algorithm tries to approximate a contour of arbitrary length using a small number of linear segments, be splitting initial segment
- Splitting stops when all points corresponding to the line segment are closer to the segment than some threhold value θ
- Distance from the line segment $a \cdot x + b \cdot y + c = 0$ to samo pont (x_i, y_i) is given by

$$d_i = \frac{|ax_i + by_i + c|}{a^2 + b^2}$$



Approximating contour with a polyline

Douglas–Peucker algorithm

PROCEDURE APPROXIMATE(C,L,n,m,θ) outline

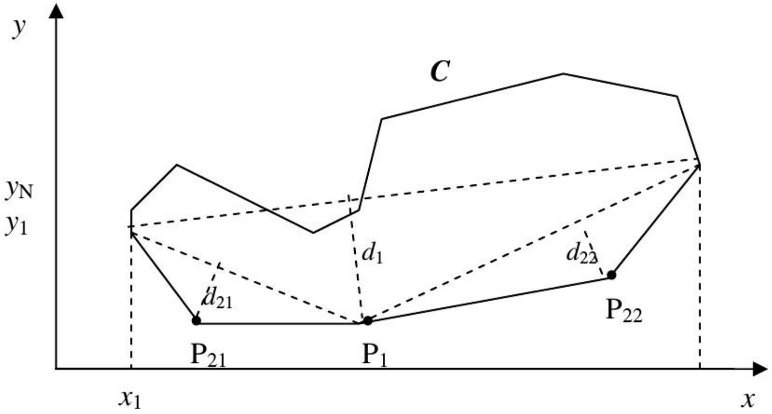
C - contour points, L-line segments,n and m - point indices, θ - distance threshold

1. Add a straight line l segment beginning in C(n) and ending in C(m) to the set of linear segments L
2. For each point in $C(i)$, where $n \leq i \leq m$ compute the distance $d(C(i), l)$
3. Select k with the largest distance to the line segment $d(C(k), l) = \max_{n \leq i \leq m} d(C(i), l)$
4. If for all computed $d(C(k), l) \leq \theta$ then return from procedure
5. Remove l form the set L
6. Call APPROXIMATE(C,L,n,k,θ),APPROXIMATE(C,L,k,m,θ)

For a contour with point indices $1 \ldots N$ the procedure can be called as follows

APPROXIMATE(C,L,1,N,θ)

Approximating contour with a polyline



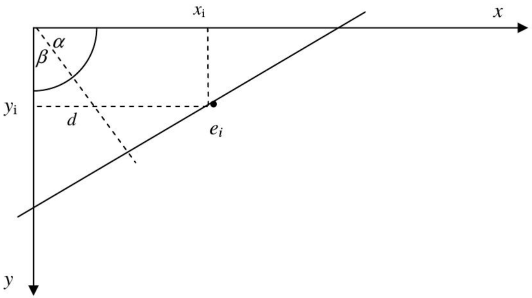
Hough transform for line detection

Let as have straight line equation given in the form:

$$d = x cos(\alpha) + y sin(\alpha)$$

where

- α - the angle orientation of the vector normal to the line
- d - the distance from the line to the coordinate sytem origin

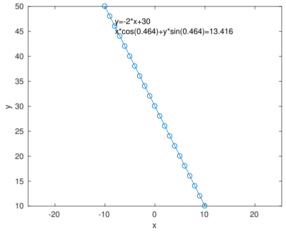


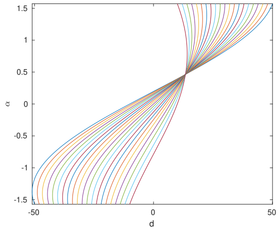
Hough transform for line detection

Then for every (x_i, y_i) pixel that is a part of the line the above equation must hold, so

$$d = x_i cos(\alpha) + y_i sin(\alpha)$$

Thinking the opposite way we will see that all lines going through the point (x_i, y_i) correspond to some sine-like function in a space $[d, \alpha]$. Let us now draw sine function for each point on the line in one graph.





Sines cross at the point $[\alpha, d]$!

Hough transform for line detection - basic algorithm

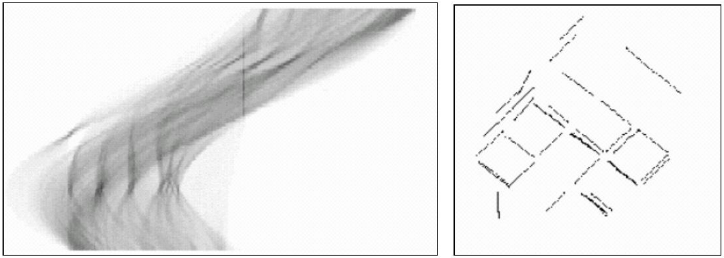
INPUT: image with detected contour elements
ALGORITHM outline:

1. Create a 2-dimensional array H (Hough accummulator) representing $[d, \alpha]$ space. The space must be discretized within range

$$-\sqrt{M^2+N^2} \le d \le \sqrt{M^2+N^2}, -\frac{\pi}{2} \le \alpha \le \frac{\pi}{2}$$

   where (M, N) is the image size
2. Set all elements of H to zero
3. For each contour element (x_i, y_i) and for all discrete values of α^+ do
    - compute $d_i = x_i cos(\alpha^+) + y_i sin(\alpha^+)$
    - approximate (d_i, α^+) with a nearest discrete value (d_i^+, α^+)
    - increment $H[d_i^+, \alpha^+]$ by 1
4. $H[d, \alpha]$ contain votes for particular (d, α) pairs. Maxima in the (smoothed) accummulator correspond to straight line segments.

Hough transform for line detection - basic algorithm


(a) (b)

(a) - Hough accumulator, (b) - input image

Hough transform for line detection - faster algorithm

INPUT: image with detected contour elements and their orientations (normals)
ALGORITHM outline:

1. Create a 2-dimensional array H (Hough accummulator) representing $[d, \alpha]$ space. The space must be discretized within range
$$-\sqrt{M^2 + N^2} \leq d \leq \sqrt{M^2 + N^2}, -\frac{\pi}{2} \leq \alpha \leq \frac{\pi}{2}$$
where (M, N) is the image size
2. Set all elements of H to zero
3. For each contour element (x_i, y_i) and for **established** contour element orientation α_i do
    - compute $d_i = x_i cos(\alpha_i) + y_i sin(\alpha_i)$
    - approximate (d_i, α_i) with a nearest discrete value (d_i^+, α_i^+)
    - increment $H[d_i^+, \alpha_i^+]$ by 1
    - $H[d, \alpha]$ contain votes for particular (d, α) pairs. Maxima in the (smoothed) accummulator correspond to straight line segments.

Hough transform for line detection - excercise

Exercise: Why not use simpler parametrization (line-to-line conversion)

$$y = a \cdot x_i + b \Leftrightarrow b = -a \cdot x_i + y$$

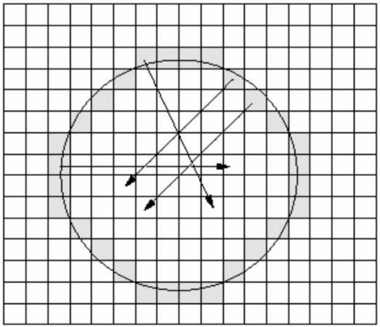
Hough transform for circle detection approach I

The circle $[c_x, c_y, r]$ is given by the equation

$$r^2 = (x - x_c)^2 + (y - y_c)^2$$

Now, select a point on the circle boundary (e_i) and construct a line going through this point and perpendicular to the circle boundary.

$$g_i : y = a_i \cdot x + b_i \text{for some boundary element } e_i$$



Hough transform for circle detection approach I

The line obviosly goes through the point (x_c, y_c).

$$g_i : y_c = a_i \cdot x_c + b_i$$

Now, considering all such lines, the following condition must hold for their parameters $[a, b]$

$$y_c = a \cdot x_c + b$$

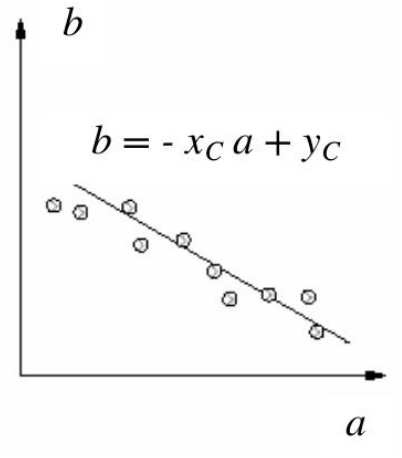
which can be rewritten as

$$b = -x_c \cdot a + y_c$$

and b is a linear functin of a So for each boundary element (e_i) introduces a pair of line parameters (a_i, b_i). The points (a_i, b_i) should lie on a single line (with unknown parameters $(-x_c, y_c)$). **Thus we just reduced a problem of detecting circles into a problem of detecting lines**

Hough transform for circle detection approach I

Note: If we transform edge elements of real images of circles we will have multiple lines with imperfectly aligned points in $[a, b]$ space.



Hough transform for circle detection approach I

PROCEDURE outline:

1. Transform boundary elements (e_i) from the image into $[a, b]$ space
2. Perform line detection in $[a, b]$ space (another Hough transform can be used here)
3. Restore (a_i, b_i) points assigned to a single detected straight line segment
4. Estimate (x_c, y_c) from the set of (a_i, b_i) points using LSE approach:

$$(c_x, c_y) = argmin \sum_i \varepsilon_i^2(c_x, c_y)$$

$$\varepsilon_i(c_x, c_y) = b_i + x_c \cdot a_i - y_c$$

Hough transform for circle detection approach II

A more direct Hough-transform-based approach to circles detection can be also given. The algorithm uses a 3-dimensional Hough accumulator $H(x_c, y_c, r)$. INPUT: image with detected boundary elements (e_i) and their orientations (normals)

PROCEDURE outline:

1. Set all the elements of the Hough accummulator to 0 $(H(x_c, y_c, r){=}0)$
2. For each boundary element e_i with corrdinates (x_i, y_i) and contour normal direction α do
   - For each allowed discrete value of circle radius r^+ do
     - Compute circle center candidate $(\tilde{x}_c, \tilde{y}_c)$

       $$\tilde{x}_c = x_i + cos(\alpha) * r^+$$

       $$\tilde{y}_c = y_i + sin(\alpha) * r^+$$

     - Round $(\tilde{x}_c, \tilde{y}_c)$ to the nearest Hough accumulator grid coordinates $(\tilde{x}_c^+, \tilde{y}_c^+)$ and increment by one $H(x_c^+, y_c^+, r^+)$
3. $H[x_c, y_c, r_c]$ contain votes for particular $(x_c, y_c, r_c$ triplets.. Maxima in the (smoothed) accummulator correspond to detected circles.

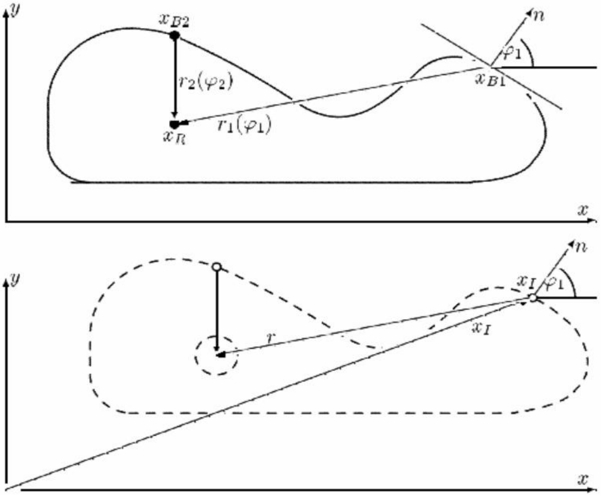
GHT for arbitrary contour detection

Generalized Hough Transform enables to detect artibtraty contour shapes in the image. The method is divide into two steps: Learning and Detection. For each detected contour, the following parameters are established:

- $\mathbf{x_r} = (x_r, y_r)$ - reference point of the contour
- s - contour scale
- α - contour orientation

For now we will assume that scale s and α are fixed.

GHT for arbitrary contour detection



GHT for arbitrary contour detection

Training - creating a lookup table R

1. Select a reference point $\mathbf{x_r}$ - e.g. a center of mass
2. For each contour point $\mathbf{x_B}$
   - compute $\phi(\mathbf{x_B})$ - gradient direction in $\mathbf{x_B}$
   - compute $\mathbf{r} = \mathbf{x_R} - \mathbf{x_B}$
   - store $\mathbf{r}$ as a pair $\phi_B \rightarrow \mathbf{r_B}$
3. Discretize ϕ and create R-table mapping gradient direction ϕ into a sequence of possible $\mathbf{r}$ vectors.

GHT for arbitrary contour detection

Detection

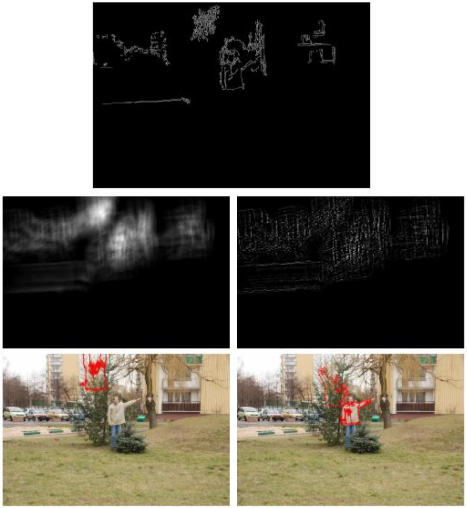
1. Determine all contour points $\mathbf{x_I}$ together with their gradient orientations
2. Initialize accummulator $H = 0$
3. For all contour points $\mathbf{x_I}$ in the image do
   - Restore from R-table all values $\mathbf{r}(\phi(\mathbf{x_I}))$ matching the current gradient $\phi(\mathbf{x_I})$
   - For each $\mathbf{r}(\phi(\mathbf{x_I}))$ increment by 1 a value of Hough accumulator cell $\mathbf{x_I} + \mathbf{r}(\phi(x_I))$ in a plane $H(s, \alpha)$ of the Hough accummulator
4. Smooth accumulator and detect local maxima
5. Detected object reference points (with respective scales and orientions) are found in the positions of local maximas.

GHT for arbitrary contour detection

GHT can be generalized for detecting object at different scales and orientations. This can be done by creating additional (or extending existing) R-tables

1. In order to prepare R-table for different scale, select scale s and and multiply $\mathbf{r_B}$ by s for each pair $(\phi_b \rightarrow \mathbf{r_b})$
2. In order to prepare R-table for different angle, select rotation angle α and add α to $\phi_B (mod \quad 2\pi)$ for each pair $(\phi_b \rightarrow \mathbf{R}_\alpha \mathbf{r_b})$ before discretizing
3. In such way the original R-tables are converted to R-tables for different scales s and angles α
4. During detection for each contour elements all R-tables calculated for different s and α must be applied. Therefore we must use and augmented Hough Accumulator $H(x_b, y_b, s, \alpha)$

GHT for arbitrary contour detection



Random Sample Consensus (RanSaC)

RANSAC algorithm outline
INPUT: D - data points, θ - maximum error of model fitting
Initialize inliers $I = \emptyset$ and model $M = null$
REPEAT until the maximum number of iterations is reached

1. Randomly select minimum number of data points from D needed to establish a model
2. Compute model parameters $M_{current}$ basing on the selected points
3. Run through the data D and select all points matching the model $M_{current}$ with error lower than θ as $I_{current}$
4. If $|I_{current}| > |I|$, then $I = I_{current}$, $M = M_{current}$

return M and I

Random Sample Consensus (RanSaC)

Remarks:

- The algorithm may take into account not only the number inliers but also overall quality of model fit
- The model can be filtered after generation (e.g. to accept only models with certain parameter values)
- The required number of iterations can be established basing on a pre-defined *confidence level p*. Confidence level is p if the probability of finding true object by RanSaC is greater than p.

Random Sample Consensus (RanSaC)

Using confidence p to obtain number of iterations

- Let N be the number of points in the set and N_o the number of points in our object. Let $w = \frac{N_o}{N}$ be the probability of selecting inlier in a one draw.
- The probability of selecting enough inliers to estimate the model is w^n, where n is the number of points required to establish the model
- The probability of successful establishment of at least one valid model in T iterations is

$$1 - (1 - w^n)^T = p$$

- And the minimum number of iterations to obtain confidence p

$$T \geq \frac{\log(1-p)}{\log(1-w^n)}$$

- Number of points in the object N_o is typically unknown, but can be approximated by $|I|$ during the algorithm execution

Random Sample Consensus (RanSaC)

After finishing the RanSaC procedure the resulting set of inliers can be iteratively adjusted according to the procedure
REPEAT until the set I stabilizes

1. Reestimate model parameters $\tilde{M}$ using ALL inliers from I
2. Run through the data D and select all points matching the model $\tilde{M}$ with an error lower than θ as I

Random Sample Consensus (RanSaC)

RanSaC is a generic parametric model-fitting tool that is robust to outliers. Some applications in Computer Vision include

- Detection of objects in 2D and 3D images
- Estimation of image transformations (e.g. homography, fundamental or essential matrices) basing on noisy point correspondences
- Fitting curves and surfaces to noisy data

Feature description

Feature description methods

Abundance of feature descriptors depending on the type of feature used, e.g.:

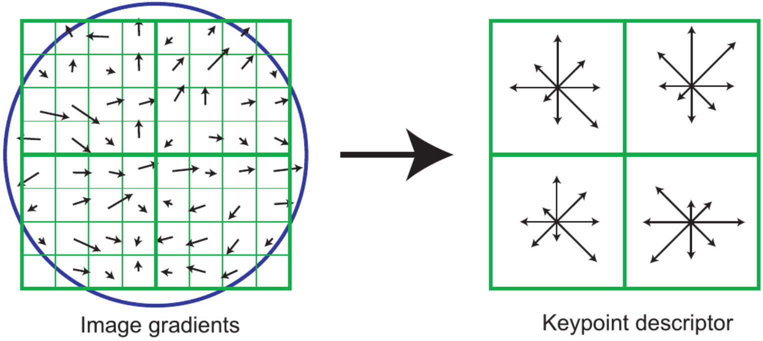
- Geometric properties (geometric moments) or Zernike's momements for geometric description
- Various pixel or color histograms
- Contour descriptors for contour features
- **SIFT**,SURF,ORB,HoG,wavelet(Haar,Gabor) descriptors for image patches
- Bag of Words features
- . . .

Scale Invariant Features Descriptor

Assume that we already have SIFT (or other type of) keypoints. For each keypoint using the image scale closest to the scale of detected keypoint

1. a 16x16 grid is imposed on the image in keypoint position (coordinates of the descriptor and gradient location are rotated according to keypoint orientation)
2. for each 4x4 cell a 8-bin histogram of gradient orientations is computed (Gaussian weighing with σ corresponding to descriptor window size is used to emphasize center pixels)
3. the descriptor is formed from all histogram bins (128 elements)
4. descriptor is normalized to unit length and its particular values are thresholded

Scale Invariant Features Descriptor



SIFT descriptor 8x8 version

Source: David. G. Lowe, Distinctive image features from scale-invariant keypoints, 2004

Automatic feature detection and description

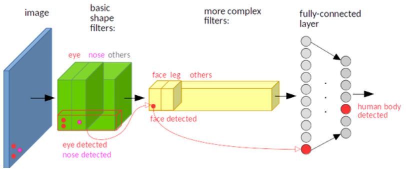
Automatic feature detection and description

- Machine Learning algorithms may be used to craft feature detection algorithms
- Some of them base on unsupervised learning - generic feature detectors (e.g. clustering methods)
- Others base on supervised learning - specialized feature detectors

These are the Feature Learning methods.

Convolutional Neural Networks for feature detection

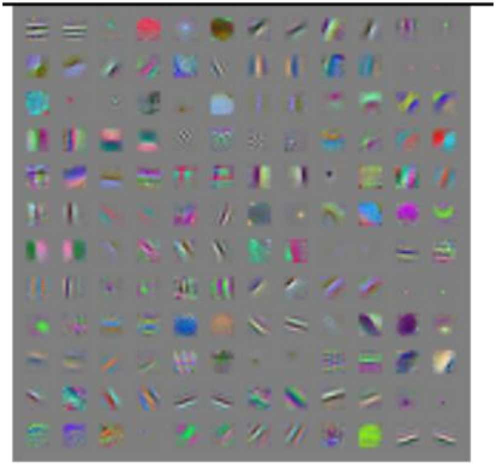
- CNNs inherently use feature detection in their convolutional layers
- Each convolutional layer takes as an input image or feature map and outputs another set of feature maps (interpreted as feature densities), higher layers incorporate information from larger areas
- Higher layers aggragate larger portions of image
- Computed feature maps are not always human-interpretable (features are generated to serve some purpose - e.g. recognition of some object class), but frequently can be re-used.



Source: `https://www.quora.com/How-is-a-convolutional-neural-network-able-to-learn-invariant-features`

Convolutional Neural Networks for feature detection

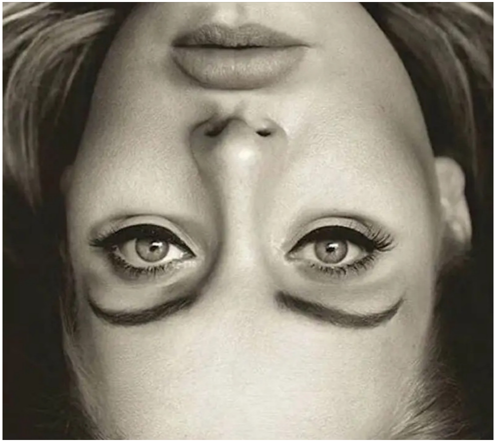
Sample features



Source: `https://www.deeplearningbook.org`

Specialized feature detectors

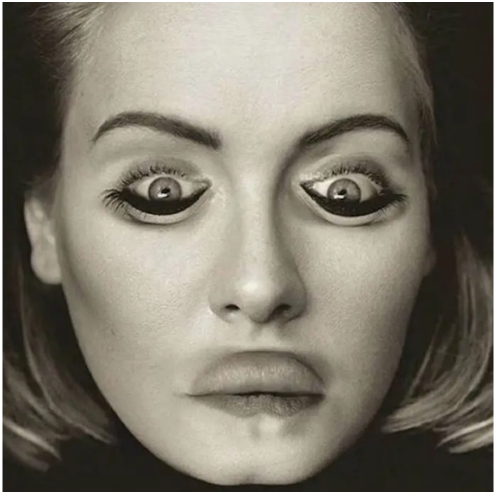
Does human brain use specialized feature detectors?



Source: https://www.businessinsider.com/

Specialized feature detectors

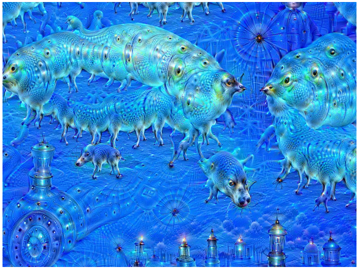
Does human brain use specialized feature detectors?



Source: `https://www.businessinsider.com/`

Specialized feature detectors

Deep Dream



Source: `https://en.wikipedia.org/wiki/DeepDream`

Convolutional Neural Networks for feature detection

- Dedicated solutions to keypoint detection/feature extraction are also present

Exercises

Exercise 1. Hough transform for line detection

The following table represents pixel coordinates and orientations for contours detected in the image:

x_i	y_i	$r(\alpha_i)$
1	1	3
2	2	3
3	3	3
4	3	3
5	5	3
3	2	6
4	2	6
5	2	6
7	3	6
1	4	0
2	4	0
3	4	0

- Assume that gradient directions are discretized into 8 bins, with $r(0) = 0$ and $r(45) = 1$ . . . .
- Apply the Hough transform for line segment detection
- Assume that at least 2 contour elements must vote for each line segment
- Provide the resulting Hough accumulator, intermediate results and an illustration

Exercise 2. Hough transform for circle detection

The following table represents pixel coordinates and orientations for contours detected in the image:

x_i	y_i	$e(\alpha_i)$
1	0	1
5	0	3
1	4	7
5	4	5

- Assume that gradient directions are discretized into 8 bins, with $e(0°) = 0$ and $e(45°) = 1$ . . . .
- Apply the Hough transform 'approach I' for circle detection
- Provide provide intermediate results and relevant illustrations

Exercise 3. Generalized Hough transform

You are given a template of a white 6×4 rectangle with rounded corners on a black background. Some points on the perimeter of the rectangle were sampled:

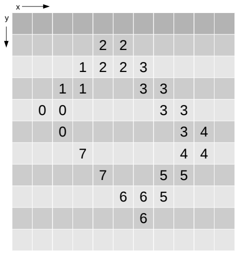
- (3,0), (3,2), (0,2), (-3,2), (-3,0), (-3,-2), (0,-2), (3,-2)

Asssume that the reference point is in the mass centre of the rectangle. Gradient direction in the corner is the average of the gradient directions of the adjacent sides.

1. Synthesize GHT R-table for this sample shape
2. How many points are there in each R-table entry? What if we add additional contour element (1,2)?
3. Synthesize GHT R-table for this shape rotated by 45 degrees

Exercise 3. Generalized Hough transform

There is given an image with detected contour points and orientations (orientations are discretized into 8 bins, shading does not reflect the actual colors).



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

1. Apply the GHT to find a single reference point in the detection of the sample rectangle. Use the rotated version of R-table.
2. Did we obtain one or multiple solutions?
3. Provide illustrations and intermediate results.

Exercise 4. RanSaC

You perform ellipse detection in the image using RanSaC algorithm. The ellipse equation is given as

$$\frac{(x - x_0)^2}{a^2} + \frac{(y - y_0)^2}{b^2} = 1$$

You detected 50000 contour points and you expect the detected ellipse to contain at least 10000 points. How many RanSaC iterations must be performed in order to detect an ellipse with the confidence of 0.99?

Exercise 5. Douglas-Peucker algorithm

Apply the Douglas-Puecker algorithm with distance threshold $\theta = 2$ to the polyline with nodes

- $(2,2) - (4,5) - (6,1) - (12,3) - (14,2)$

Provide intermediate states of the polyline.

Exercise 1. Hough transform for line detection - hints

The following table represents pixel coordinates and orientations for contours detected in the image:

x_i	y_i	$r(\alpha_i)$
1	1	3
2	2	3
3	3	3
4	3	3
5	5	3
3	2	6
4	2	6
5	2	6
7	3	6
1	4	0
2	4	0
3	4	0

- Assume that gradient directions are discretized into 8 bins, with $r(0) = 0$ and $r(45) = 1 \ldots.$
- Apply the Hough transform for line segment detection
- Assume that at least 2 contour elements must vote for each line segment
- Provide the resulting Hough accumulator, intermediate results and an illustration

Exercise 1. Hough transform for line detection - hints

Extended table: $d_i = x_i \cos(\alpha) + y_i \sin(\alpha)$

x_i	y_i	$r(\alpha_i)$	$\sin(\alpha_i)$	$\cos(\alpha_i)$	d_i	d_i^+
1	1	3	$\frac{\sqrt{2}}{2}$	$-\frac{\sqrt{2}}{2}$	0	0
2	2	3	$\frac{\sqrt{2}}{2}$	$-\frac{\sqrt{2}}{2}$	0	0
3	3	3	$\frac{\sqrt{2}}{2}$	$-\frac{\sqrt{2}}{2}$	0	0
4	3	3	$\frac{\sqrt{2}}{2}$	$-\frac{\sqrt{2}}{2}$	$-4\frac{\sqrt{2}}{2} + 3\frac{\sqrt{2}}{2} = -\frac{\sqrt{2}}{2}$	-1
5	5	3	$\frac{\sqrt{2}}{2}$	$-\frac{\sqrt{2}}{2}$	0	0
3	2	6	-1	0	-2	-2
4	2	6	-1	0	-2	-2
5	2	6	-1	0	-2	-2
7	3	6	-1	0	-3	-3
1	4	0	0	1	1	1
2	4	0	0	1	2	2
3	4	0	0	1	3	3

Exercise 1. Hough transform for line detection - hints

Hough accumulator (red - local maxima of votes)

$d_i^+ / r(\alpha_i)$	0	1	2	3	4	5	6	7
-3							1	
-2							3	
-1				1				
0				4				
1	1							
2	1							
3	1							

Line 1: $x \cos(135^\circ) + y \sin(135^\circ) = 0 \rightarrow y = x$
Line 2: $x \cos(270^\circ) + y \sin(270^\circ) = -2 \rightarrow y = 2$

Exercise 2. Hough transform for circle detection - hints

The following table represents pixel coordinates and orientations for contours detected in the image:

x_i	y_i	$e(\alpha_i)$
1	0	1
5	0	3
1	4	7
5	4	5

- Assume that gradient directions are discretized into 8 bins, with $e(0°) = 0$ and $e(45°) = 1$ $\ldots$.
- Apply the Hough transform 'approach I' for circle detection
- Provide provide intermediate results and relevant illustrations

Exercise 2. Hough transform for circle detection - hints

Basing on contour points and their orientation one can find lines normal to contour elements. Each line is described by the equation $y = ax + b$

x_i	y_i	$e(\alpha_i)$	α_i	$a_i = \tan(\alpha_i)$	b_i
1	0	1	$45°$	1	-1
5	0	3	$135°$	-1	5
1	4	7	$315°$	-1	5
5	4	5	$225°$	1	-1

It can be observed that pairs of coordinates in the Hough space (a_i, b_i) are compressed to just 2 points $(1, -1), (-1, 5)$. Through these points we can (in the Hough space) draw a line $b = -x_0 a + y_0$. In our case the line has equation $b = -3a + 2$. Therefore the detected circle center has coordinates $(x_0, y_0) = (3, 2)$.

Remark: For larger number of points (a_i, b_i) in the Hough space it is neccesary to solve the regression problem.

Exercise 3. Generalized Hough transform - hints

You are given a template of a white 6×4 rectangle with rounded corners on a black background. Some points on the perimeter of the rectangle were sampled:

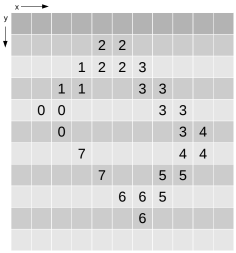
- (3,0), (3,2), (0,2), (-3,2), (-3,0), (-3,-2), (0,-2), (3,-2)

Asssume that the reference point is in the mass centre of the rectangle. Gradient direction in the corner is the average of the gradient directions of the adjacent sides.

1. Synthesize GHT R-table for this sample shape
2. How many points are there in each R-table entry? What if we add additional contour element (1,2)?
3. Synthesize GHT R-table for this shape rotated by 45 degrees

Exercise 3. Generalized Hough transform - hints

There is given an image with detected contour points and orientations (orientations are discretized into 8 bins, shading does not reflect the actual colors).



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

1. Apply the GHT to find a single reference point in the detection of the sample rectangle. Use the rotated version of R-table.
2. Did we obtain one or multiple solutions?
3. Provide illustrations and intermediate results.

Exercise 3. Generalized Hough Transform - hints

ad. 1

- Rounded corners - we assume that in rectangle corners we have edge directions : 1,3,5,7
- For example, in $(2, -3)$, we have direction 3 and a vector $\mathbf{r}(3) = [-3, 2]^T$

ϕ_i	$\mathbf{r}(\phi_i)$
0	$[3, 0]^T$
1	$[3, 2]^T$
2	$[0, 2]^T$
3	$[-3, 2]^T$
4	$[-3, 0]^T$
5	$[-3, -2]^T$
6	$[0, -2]^T$
7	$[3, -2]^T$

Exercise 3. Generalized Hough Transform - hints

ad. 2

- After adding a new contour element $(1, 2)$, R-table for direction 6 is extended by a new vector.

ϕ_i	$\mathbf{r}(\phi_i)$
0	$[3, 0]^T$
1	$[3, 2]^T$
2	$[0, 2]^T$
3	$[-3, 2]^T$
4	$[-3, 0]^T$
5	$[-3, -2]^T$
6	$[0, -2]^T$, $[-1, -2]^T$
7	$[3, -2]^T$

Exercise 3. Generalized Hough Transform - hints

ad. 2

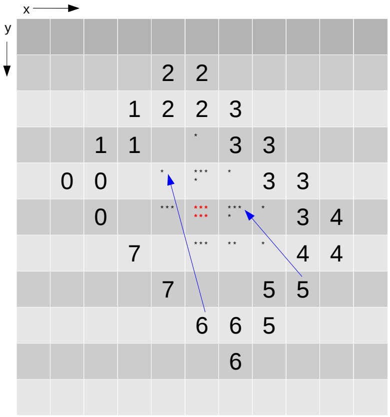
- Rotation $\alpha = \pi/4$
- All elements in R-table are first circularly shifted by 1
- Rotation matrix:
$$\mathbf{R}_\alpha = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$$
- Rotation formula:
$$\mathbf{R}_\alpha \mathbf{x} = \frac{\sqrt{2}}{2} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \frac{\sqrt{2}}{2} [x - y, x + y]^T$$

ϕ_i	$\mathbf{r}(\phi_i)$
0	$\frac{\sqrt{2}}{2}[3-(-2), 3+(-2)]^T = \frac{\sqrt{2}}{2}[5,1]^T \approx [4,1]^T$
1	$\frac{\sqrt{2}}{2}[3-0, 3+0]^T = \frac{\sqrt{2}}{2}[3,3]^T \approx [2,2]^T$
2	$\frac{\sqrt{2}}{2}[3-2, 3+2]^T = \frac{\sqrt{2}}{2}[1,5]^T \approx [1,4]^T$
3	$[-1,1]^T$
4	$[-4,-1]^T$
5	$[-2,-2]^T$
6	$[-1,-4]^T$
7	$[1,-1]^T$

Exercise 3. Generalized Hough Transform - hints

ad. 2

- dots - vote counts
- blue - sample votes in the Hough accumulator
- red - the only maximum (single detection of the rectangle)



Exercise 4. RanSaC - hints

You perform ellipse detection in the image using RanSaC algorithm. The ellipse equation is given as

$$\frac{(x-x_0)^2}{a^2}+\frac{(y-y_0)^2}{b^2}=1$$

You detected 50000 contour points and you expect the detected ellipse to contain at least 10000 points. How many RanSaC iterations must be performed in order to detect an ellipse with the confidence of 0.99?

Ćwiczenie 4. RanSaC - hints

- Number of points required to estimate the model of ellipse is 4
- Required number of iterations is 2876

Exercise 5. Douglas-Peucker algorithm - hints

Douglas-Peucker

Apply the Douglas-Puecker algorithm with distance threshold $\theta = 2$ to the polyline with nodes

- $(2,2) - (4,5) - (6,1) - (12,3) - (14,2)$

Provide intermediate states of the polyline.

Exercise 5. Douglas-Peucker algorithm - hints

The state of the estimated polyline in subsequent iterations of the algorithm.

1. $(2,2)$-$(14,2)$
2. $(2,2)$-$(4,5)$-$(14,2)$
3. $(2,2)$-$(4,5)$-$(6,1)$-$(14,2)$