

Concepțe și aplicații în Vederea Artificială

Bogdan Alexe

bogdan.alexe@fmi.unibuc.ro

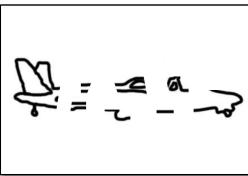
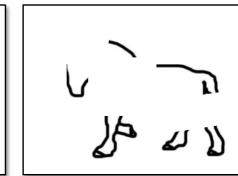
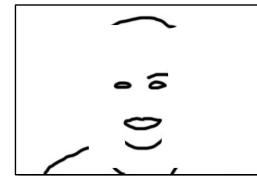
Radu Ionescu

radu.ionescu@fmi.unibuc.ro

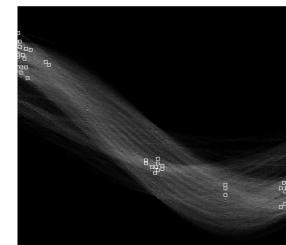
Curs optional
semestrul I, 2023-2024

Cursul trecut

- Aplicații ale filtrelor: extragerea informației (muchii)

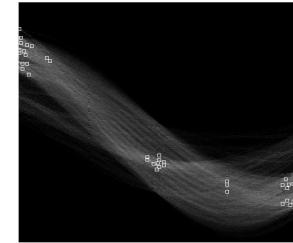


- Aplicație: detectarea liniilor cu
 - transformata Hough

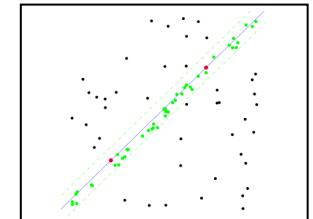


Cursul de azi

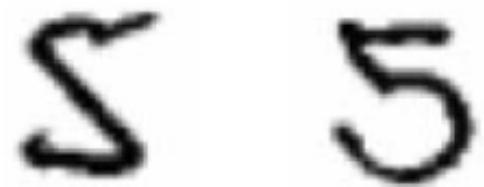
- Aplicație: detectarea liniilor cu
 - transformata Hough



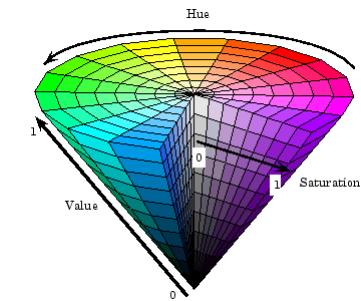
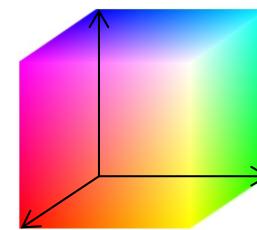
- RANSAC



- Compararea contururilor

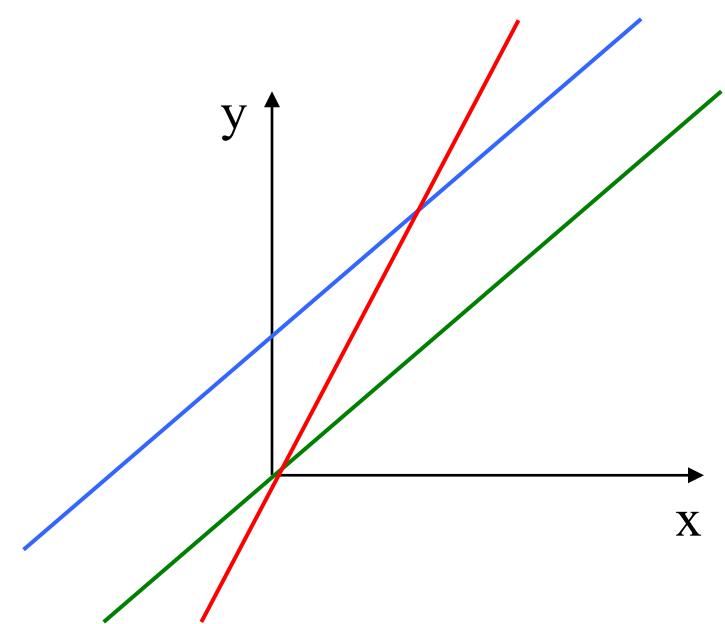


- Spațiile de culori RGB și HSV



Aplicație: detectarea linilor cu transformata Hough

Parametrizarea dreptelor. Exemple



$d_1: y = x$, adică $m=1, b = 0$

$d_2: y = x + 2$, adică $m=1, b = 2$

$d_3: y = 2x$, adică $m=2, b = 0$

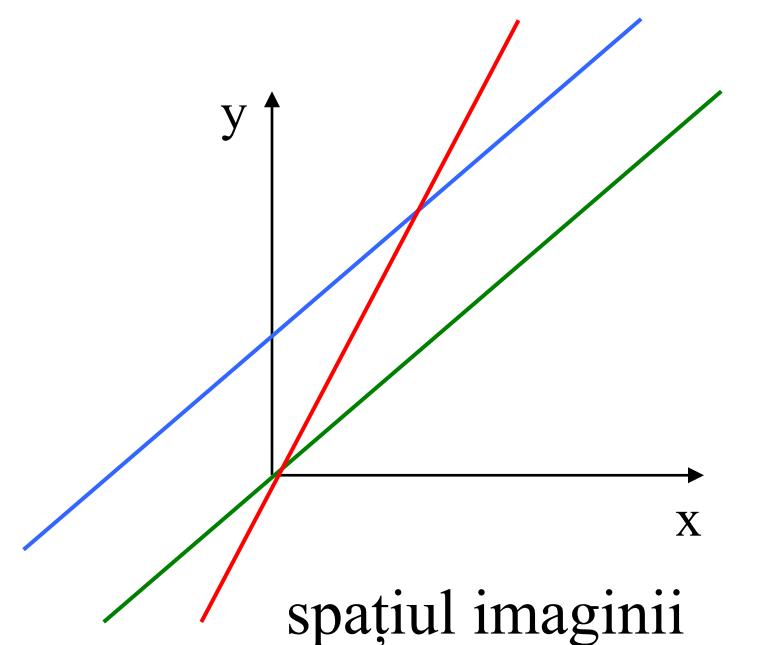
Ecuatia unei drepte in planul imaginii: $y = m * x + b$,

m – panta dreptei (definește înclinarea față de axa Ox),

b – deplasarea (definește deplasarea față de originea O(0,0))

(m, b) sunt parametrii dreptei

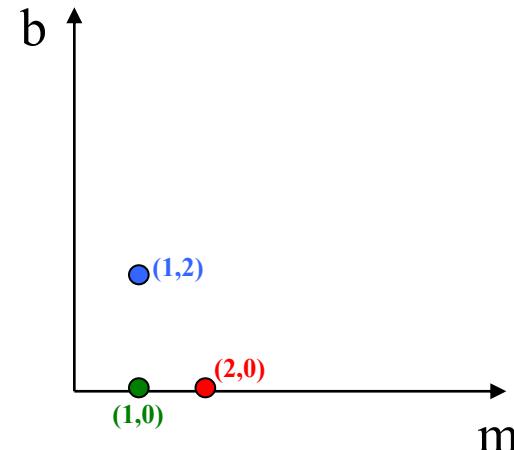
Spațiul Hough al parametrilor



$d_1: y = x$, adică $m=1, b = 0$

$d_2: y = x + 2$, adică $m=1, b = 2$

$d_3: y = 2x$, adică $m=2, b = 0$



spațiul Hough
(al parametrilor)

$d_1: m=1, b = 0 \rightarrow (1,0)$

$d_2: m=1, b = 2 \rightarrow (1,2)$

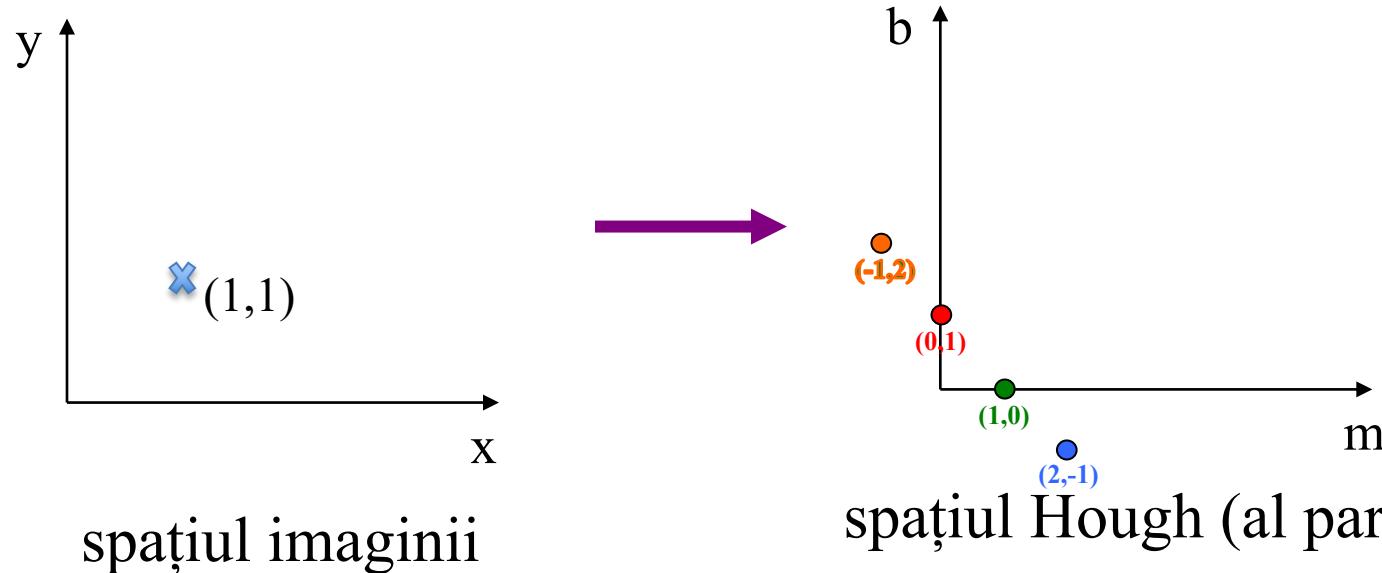
$d_3: m=2, b = 0 \rightarrow (2,0)$

O dreaptă într-o imagine corespunde unui punct în spațiul Hough.

Corespondența dintre cele două spații

O dreaptă într-o imagine corespunde unui punct în spațiul Hough.

Un punct (x_0, y_0) în spațiul imaginii corespunde unei drepte în spațiul Hough.



Prin punctul $(1,1)$ trec o infinitate de drepte de forma $y = mx+b$:

$$y = x \quad (m=1, b=0); \quad y = 2x-1 \quad (m=2, b=-1), \quad y = 1 \quad (m=0, b = -1), \quad y = -x + 2 \quad (m=-1, b = 2),$$

Toate cele 4 puncte $(1,0)$, $(2,-1)$, $(0,-1)$, $(-1,2)$ stau pe dreapta $b = -m+1$

Corespondența dintre cele două spații

O dreaptă într-o imagine corespunde unui punct în spațiul Hough.

Un punct (x_0, y_0) în spațiul imaginii corespunde unei drepte în spațiul Hough.

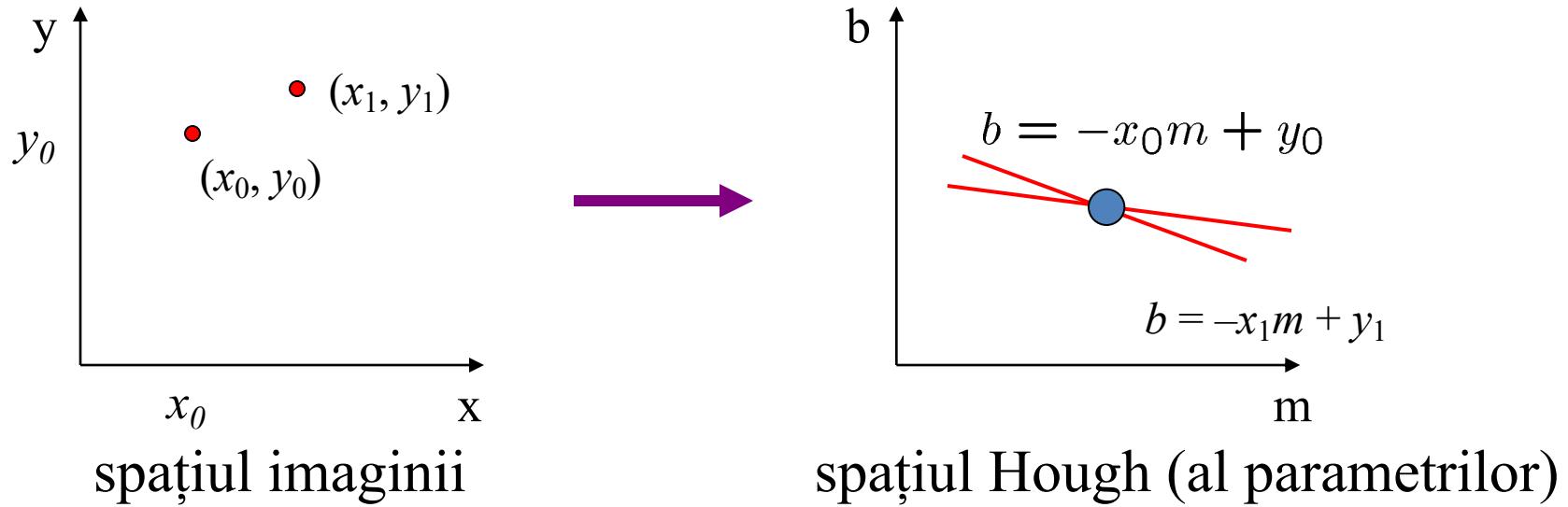
Ecuția unei drepte în planul imaginii: $y = m * x + b$,

Dreptele care trec prin punctul (x_0, y_0) pot avea orice pantă m și orice deplasare b cu condiția ca $y_0 = m * x_0 + b$.

$y_0 = m * x_0 + b \Leftrightarrow b = -x_0 * m + y_0$ (dreapta cu panta $-x_0$ și deplasare y_0)

$(x_0, y_0) = (1, 1) \Rightarrow b = -m + 1$

Găsirea liniilor într-o imagine folosind spațiul Hough



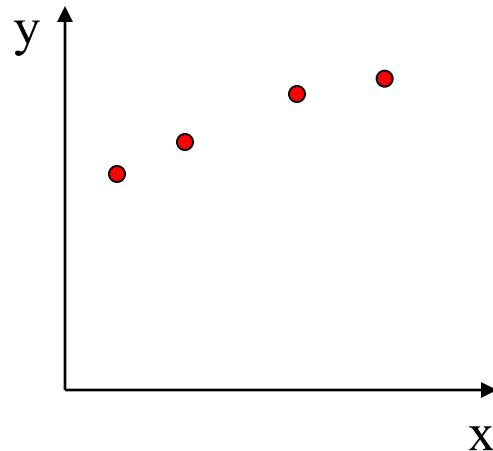
Care sunt parametrii liniei care conține punctele (x_0, y_0) și (x_1, y_1) ?

Punctului (x_0, y_0) îi corespunde dreapta de ecuație $b = -x_0m + y_0$

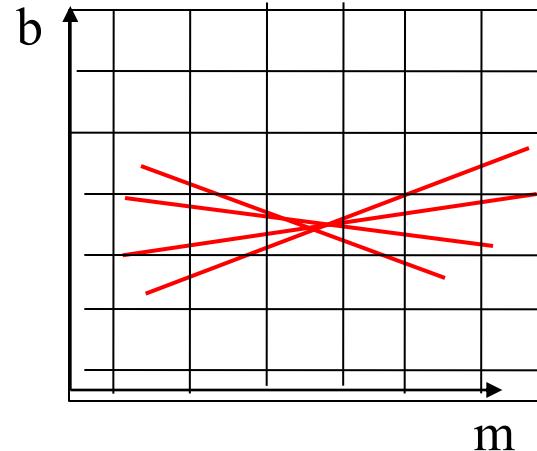
Punctului (x_1, y_1) îi corespunde dreapta de ecuație $b = -x_1m + y_1$

Dreptei care conține punctele (x_0, y_0) , (x_1, y_1) îi corespunde în spațiul Hough un punct. Acest punct se obține ca fiind intersecția dreptelor de ecuație $b = -x_0m + y_0$ și $b = -x_1m + y_1$. **Apar probleme când $x_0 = x_1$.**

Găsirea liniilor într-o imagine: algoritmul Hough



spațiu imaginii



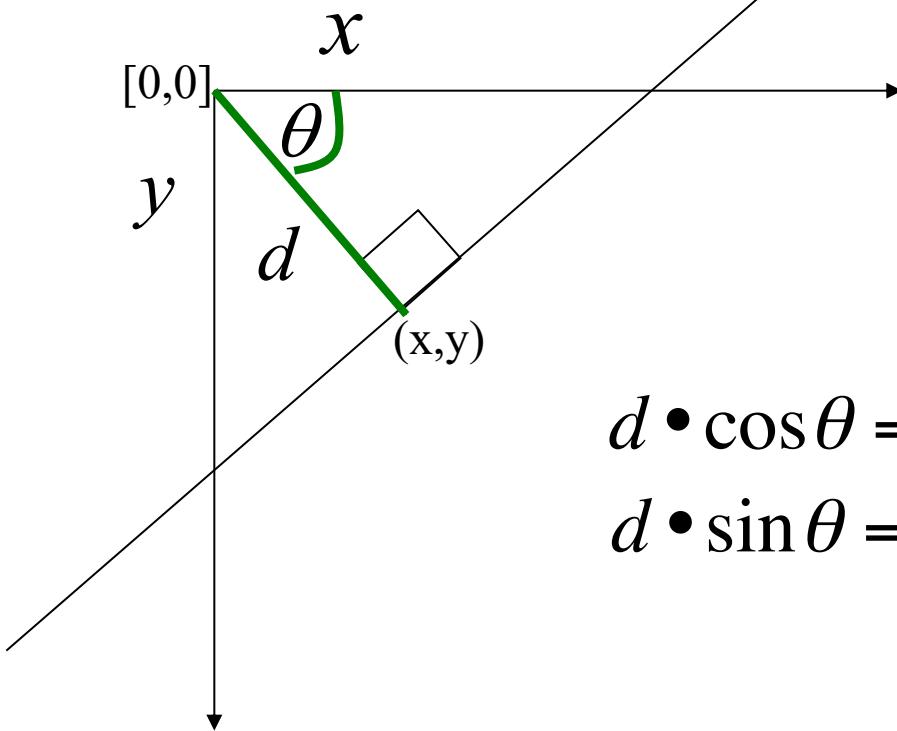
spațiu Hough
(al parametrilor)

Cum putem folosi observația anterioară pentru a găsi parametri (m, b) ce definesc linia cea mai probabilă în spațiul imaginii?

- fiecare punct detectat ca fiind edgel în spațiul imaginii va vota pentru o mulțime de parametri în spațiul Hough
- acumulăm voturi în intervale discrete; parametri cu cel mai mare număr de voturi determină linia din spațul imaginii

Reprezentarea în coordonate polare pentru detectarea liniilor

Probleme cu spațiul Hough al parametrilor (m, b) : m poate lua o valoare infinită, probleme pentru linii verticale. **Din acest motiv vom folosi o altă parametrizare (coordonate polare):**



d : distanța perpendiculară din origine la linie

θ : unghiul dintre perpendiculară și axa Ox

$$d \cdot \cos \theta = x$$

$$d \cdot \sin \theta = y$$

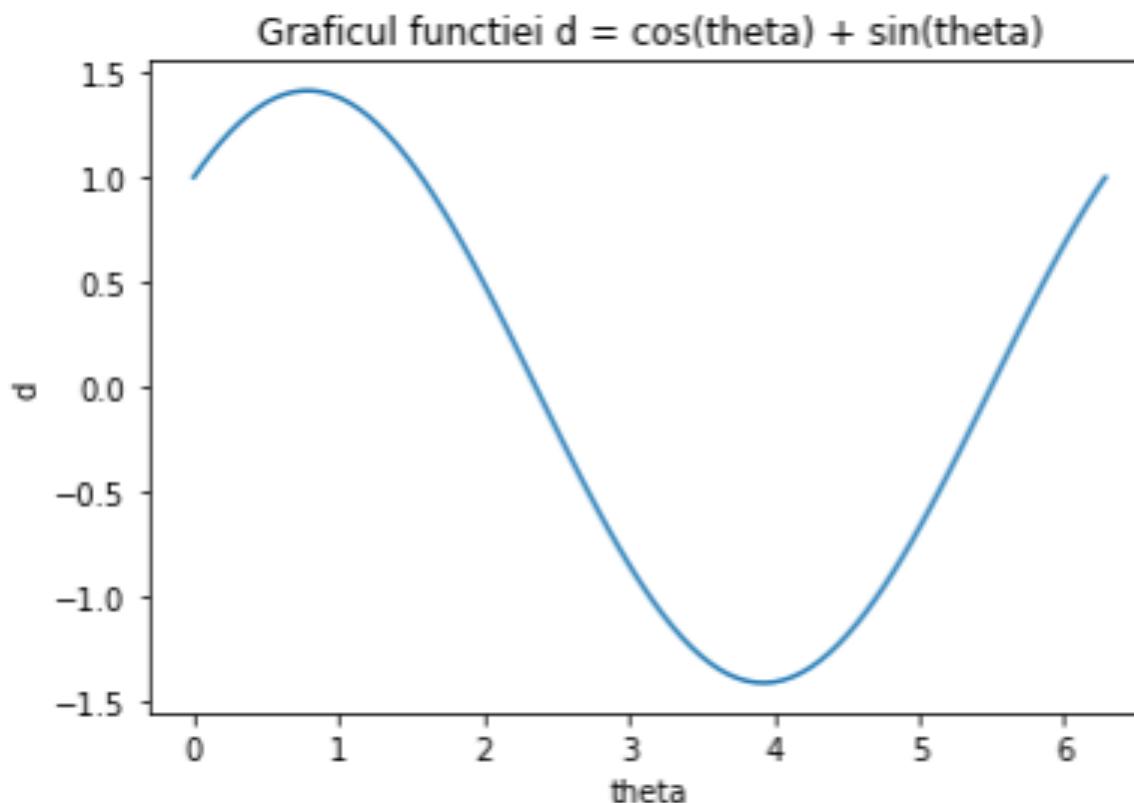
$$d \cdot (\cos \theta)^2 = x \cdot \cos \theta$$

$$d \cdot (\sin \theta)^2 = y \cdot \sin \theta$$

$$d = x \cdot \cos \theta + y \cdot \sin \theta$$

Punct în spațiul imaginii \rightarrow curbă sinusoidală în spațiul Hough

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 x, y = 1,1
4 theta = np.linspace(0,2*np.pi,100)
5 d = x * np.cos(theta) + y * np.sin(theta)
6 plt.figure,
7 plt.plot(theta, d)
8 plt.xlabel('theta')
9 plt.ylabel('d')
10 plt.title('Graficul functiei d = cos(theta) + sin(theta)')
11 plt.show()
```

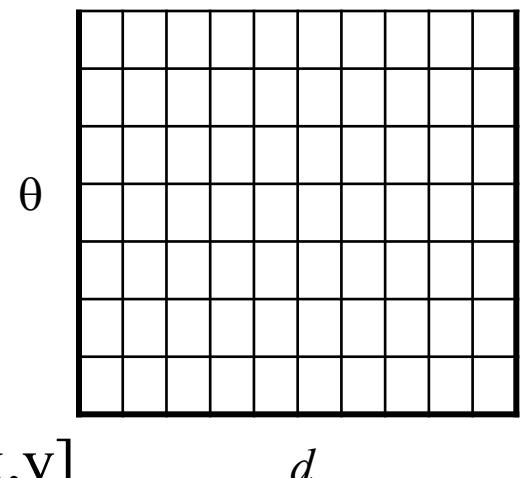


Algoritmul transformatei Hough

- folosește coordonate polare

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

H: accumulează voturi



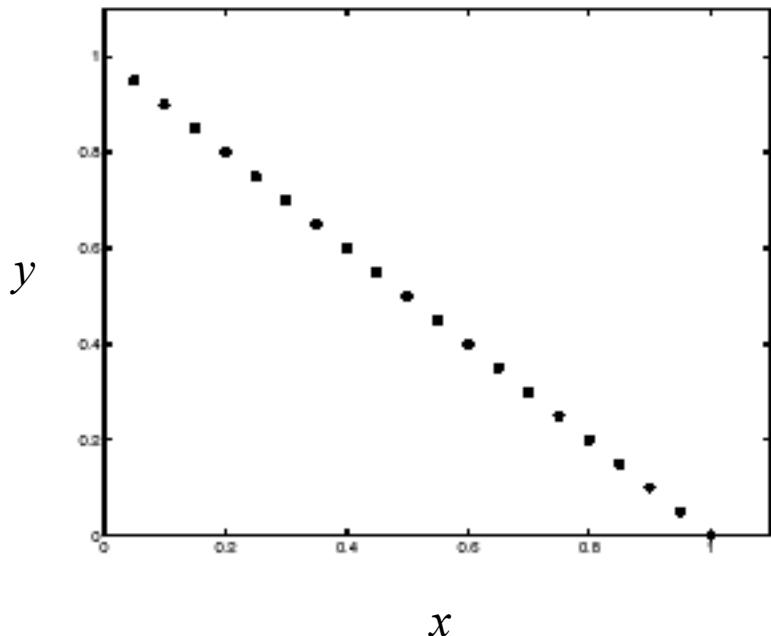
Algoritmul transformatei Hough

1. initializează $H[d, \theta] = 0$
2. pentru fiecare punct edgel din imagine $I[x, y]$
pentru $\theta = 0$ to 360 // $[0, 2\pi]$ e împărțit în intervale
$$d = x \cos \theta + y \sin \theta$$
$$H[d, \theta] += 1$$
3. găsește valorile (d, θ) pentru care $H[d, \theta]$ este maxim
4. linia detectată în imagine este dată de:
$$d = x \cos \theta + y \sin \theta$$

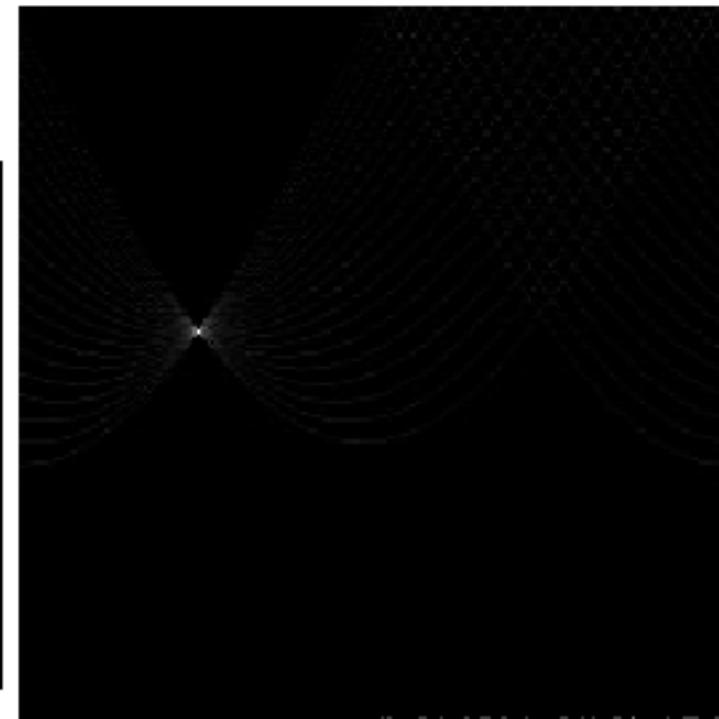
DEMO:

<https://www.aber.ac.uk/~dcswww/Dept/Teaching/CourseNotes/current/CS34110/hough.html>

Exemplu: transformata Hough pentru linii



**spațiu imaginii
coordonatele punctelor edgels**

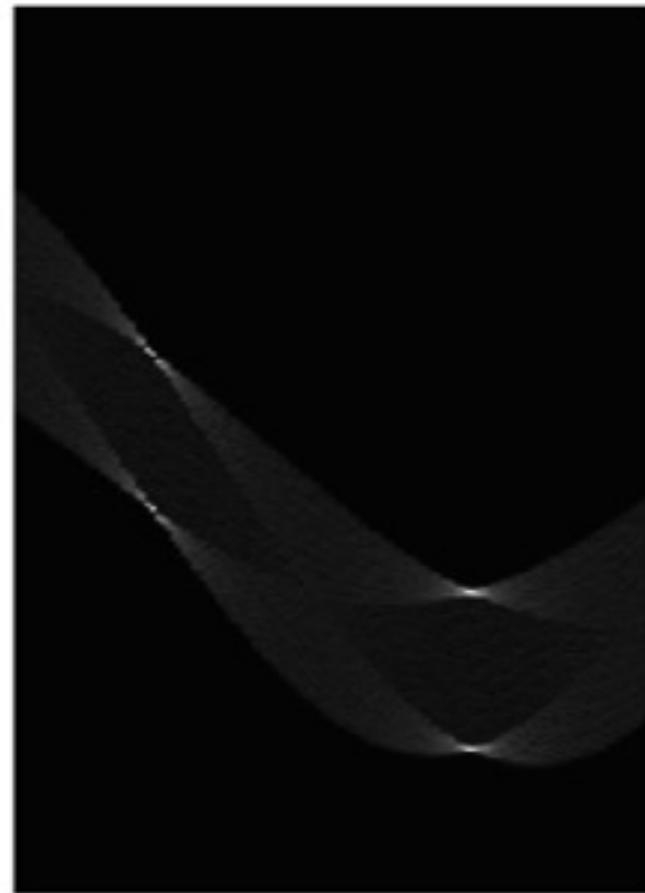


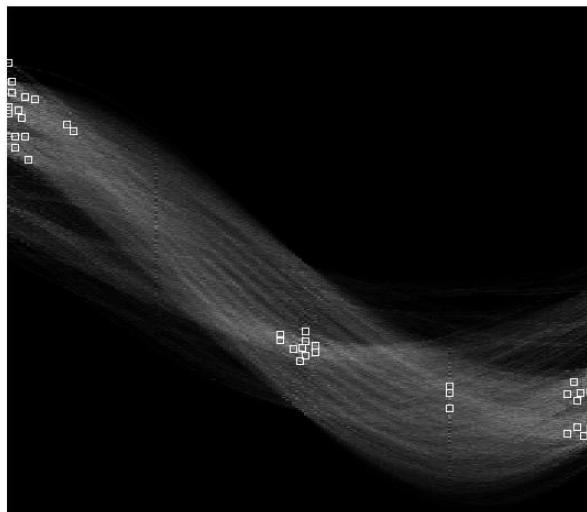
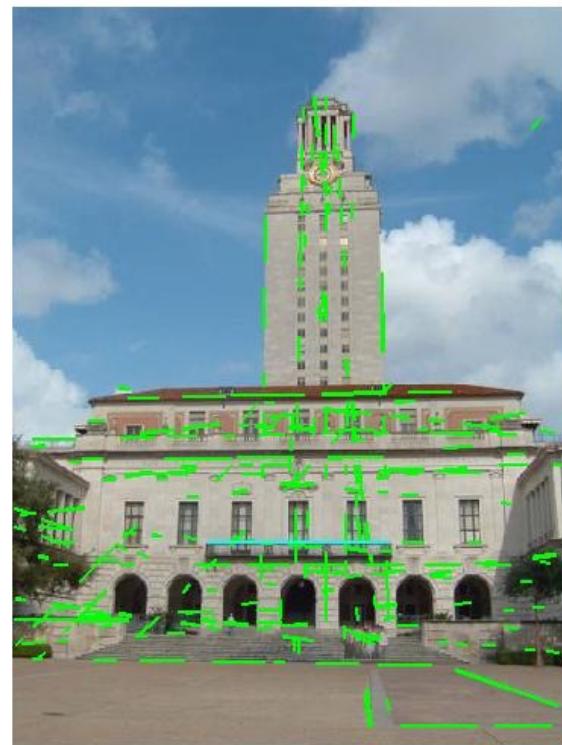
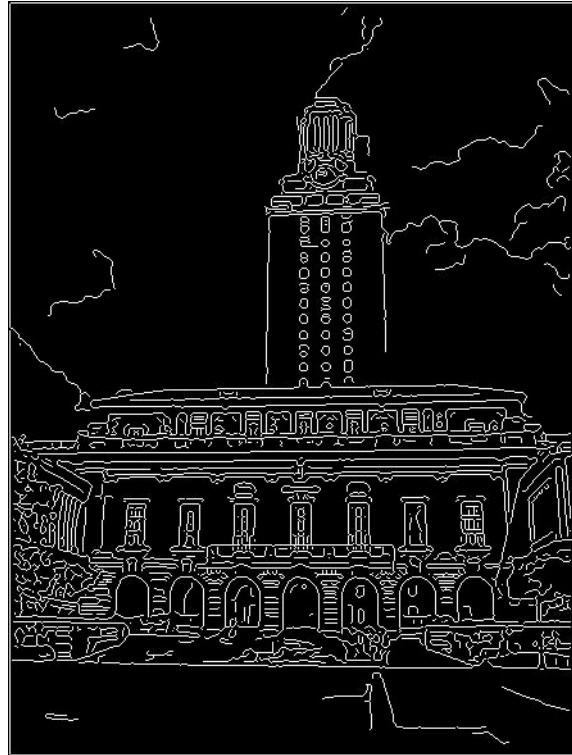
Voturi

culoare deschisă = # mare de voturi
negru = zero voturi

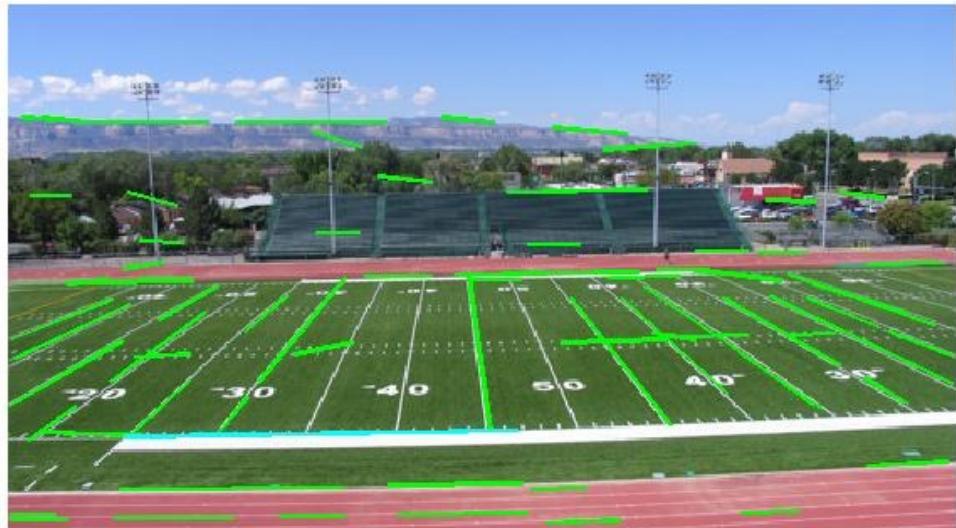
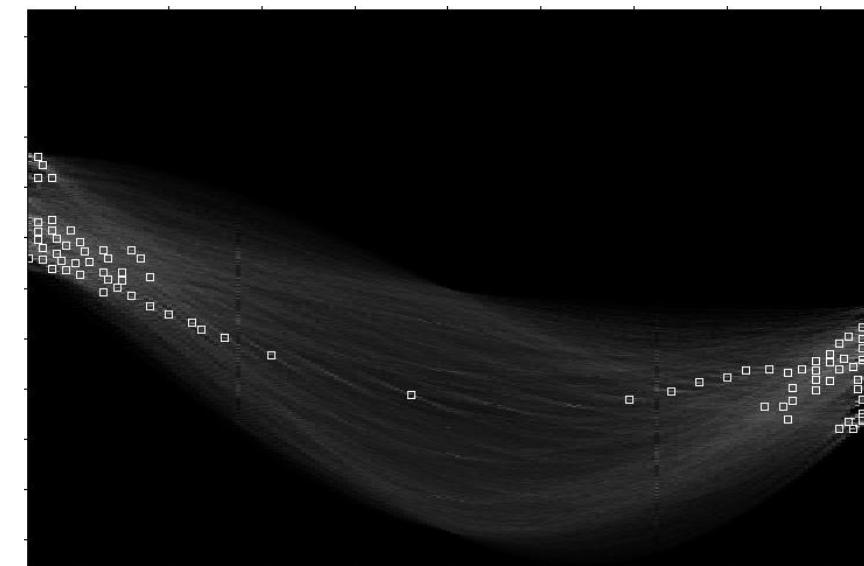
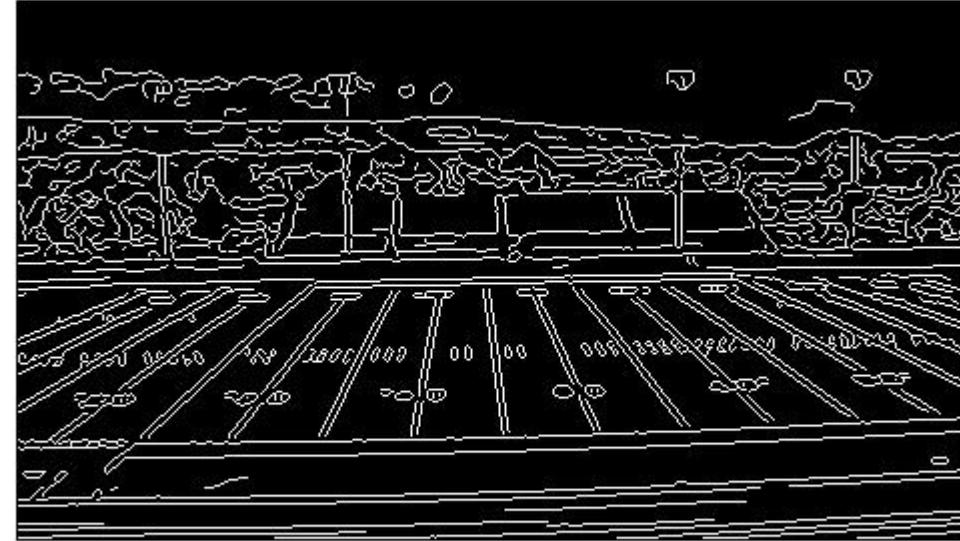
Exemplu: transformata Hough pentru linii

Pătrat:



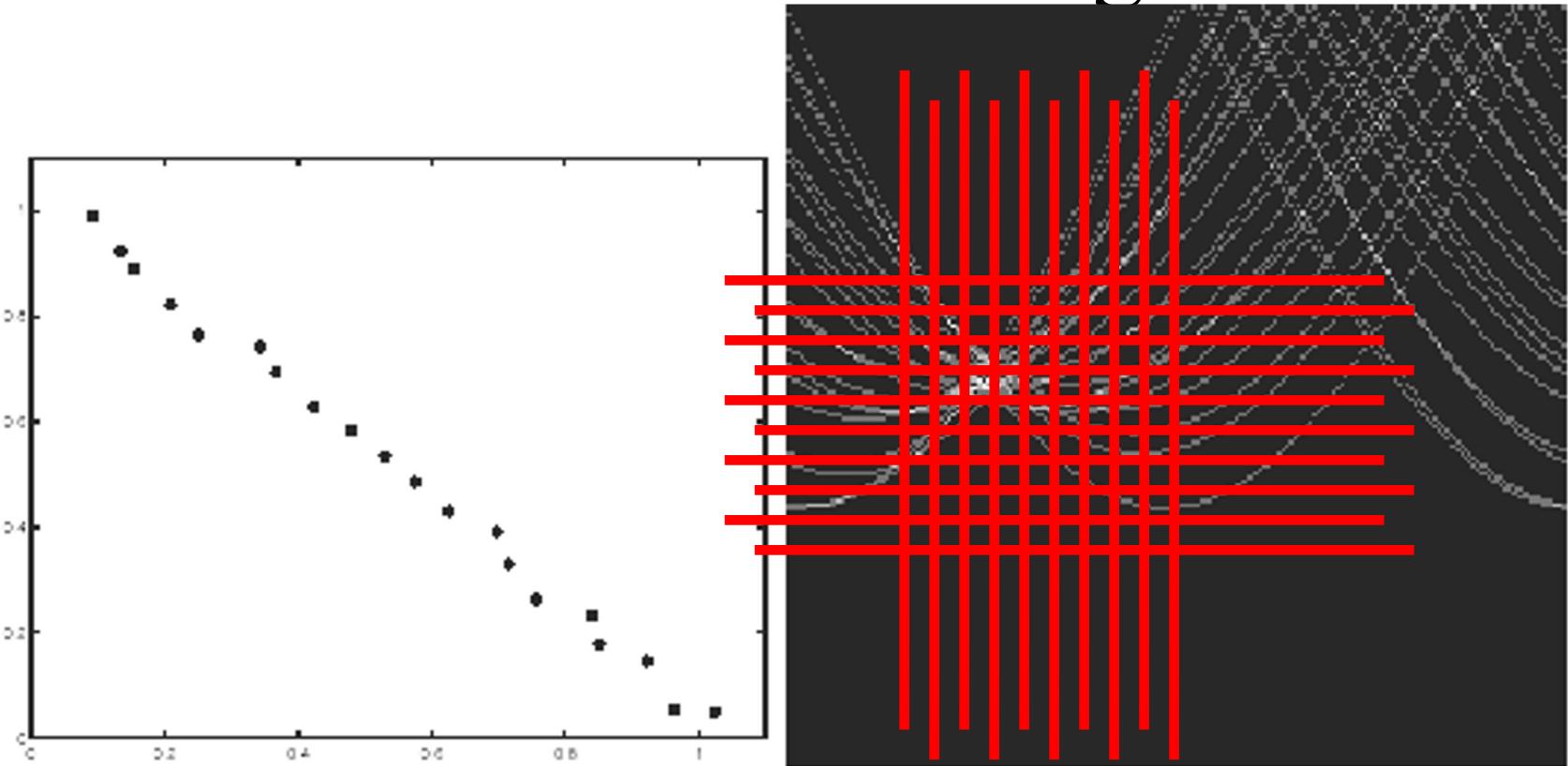


Slide adaptat după S. Lazebnik



Sunt afișate cele mai lungi segmente

Impactul zgomotului asupra transformatei Hough

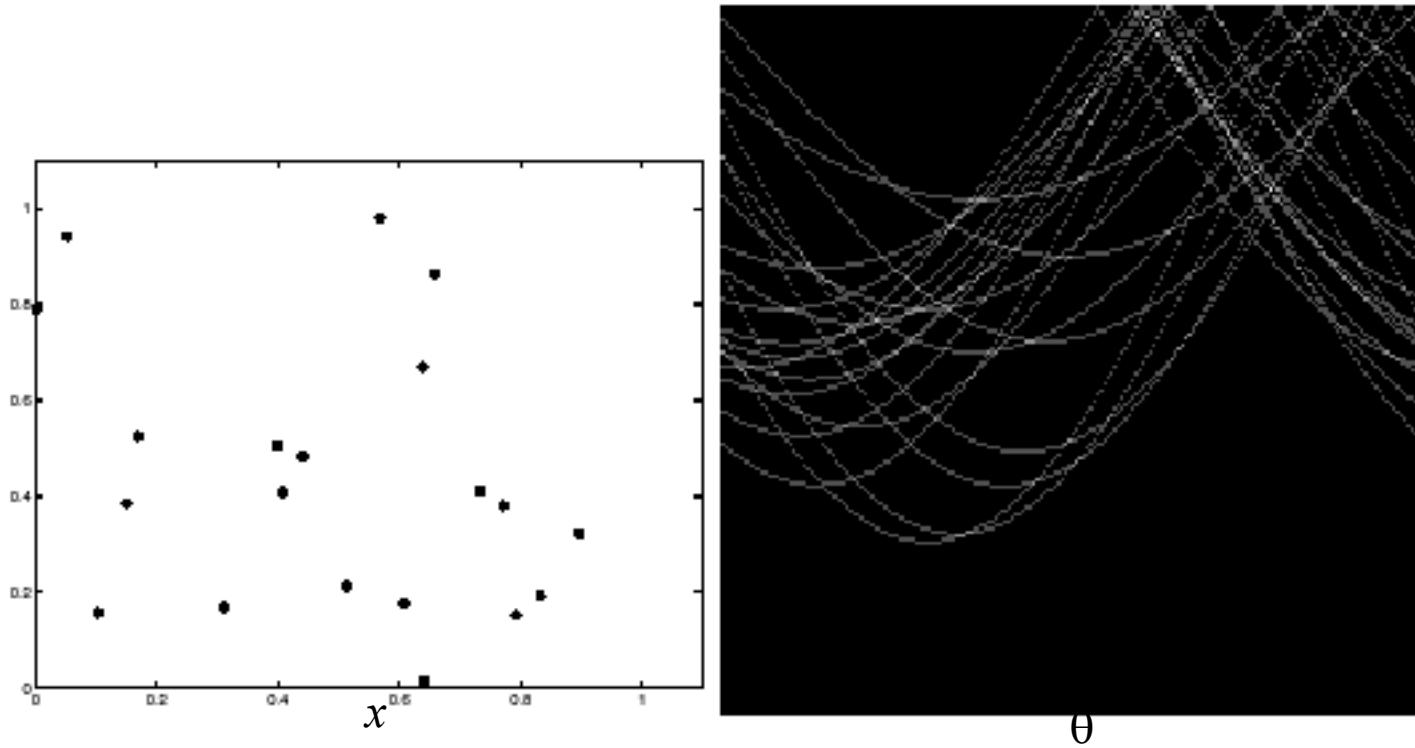


spațiu imaginii
coordonatele punctelor edgels

Voturi

culoare deschisă = # mare de voturi
negru = zero voturi

Impactul zgomotului asupra transformatei Hough

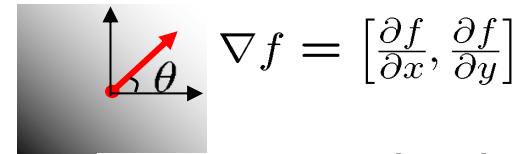


spațiu imaginii
coordonatele punctelor edgels

Voturi

Totul pare a fi zgomot sau puncte edgels aleatoare. Se observă niște peak-uri în spațiul parametrilor.

Extensi



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

Extensia 1: folosește gradientul imaginii

1. initializează $H[d, \theta] = 0$ (la fel)
2. pentru fiecare punct edgel din imagine $I[x, y]$

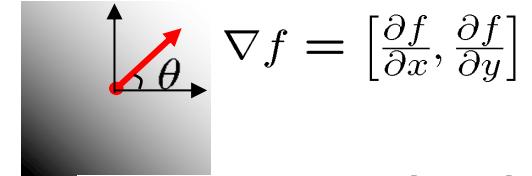
θ = gradientul imaginii la (x, y)

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

$$H[d, \theta] += 1$$

3. găsește valorile (d, θ) pentru care $H[d, \theta]$ este maxim (la fel)
4. linia detectată în imagine este dată de: $d = x \cos \theta + y \sin \theta$

Extensiî



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

1. initializează $H[d, \theta] = 0$ (la fel)
2. pentru fiecare punct edgel din imagine $I[x, y]$

θ = gradientul imaginii la (x, y)

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

$$H[d, \theta] += 1$$

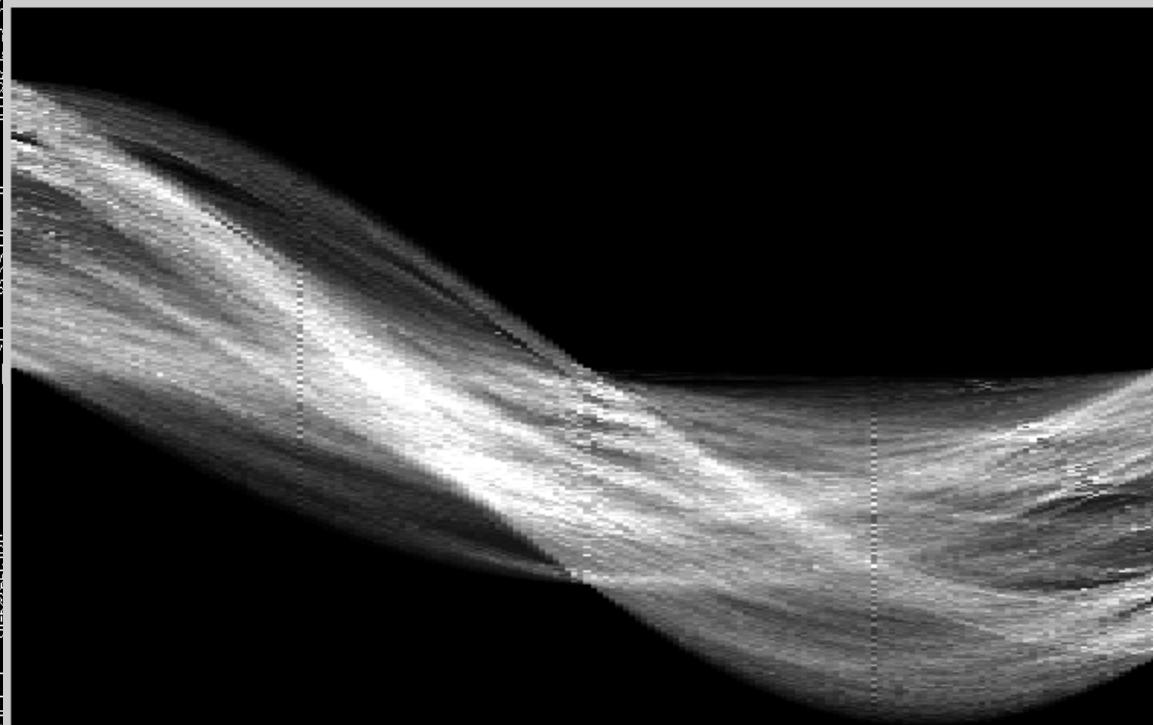
3. găsește valorile (d, θ) pentru care $H[d, \theta]$ este maxim (la fel)
4. linia detectată în imagine este dată de: $d = x \cos \theta + y \sin \theta$

Extensia 2: $H[d, \theta] +=$ magnitudine gradient (x, y)

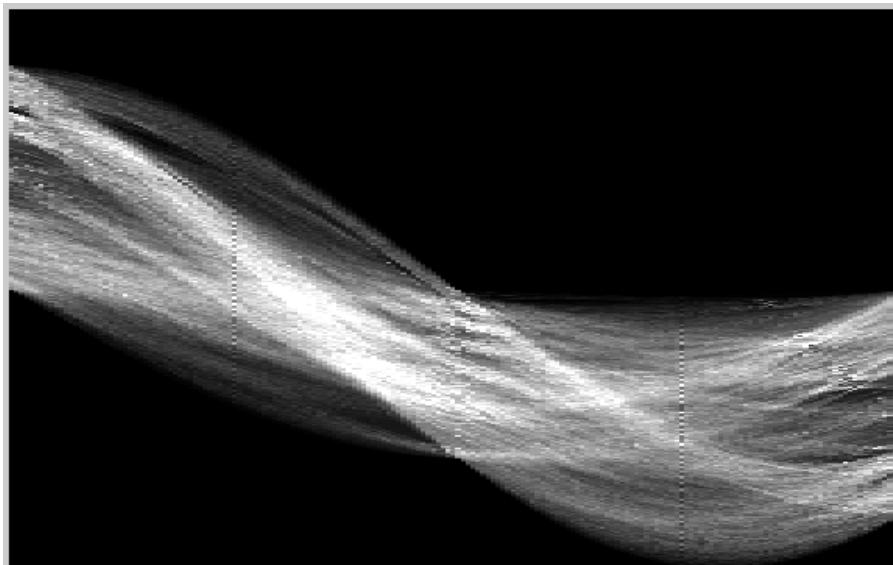
Imagine → Canny



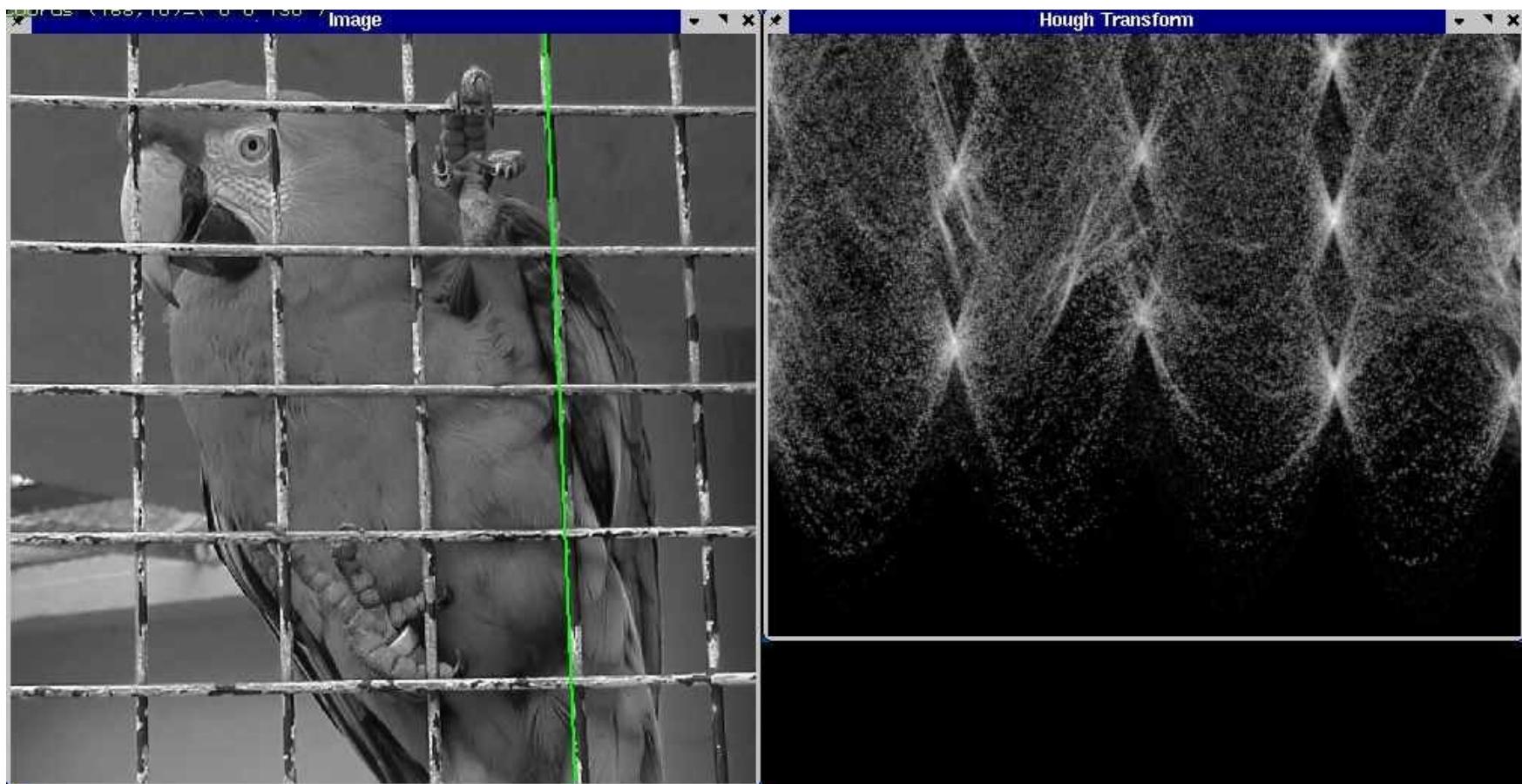
Canny → Transformata Hough



Transformata Hough → Muchii



Transformata Hough - exemplu



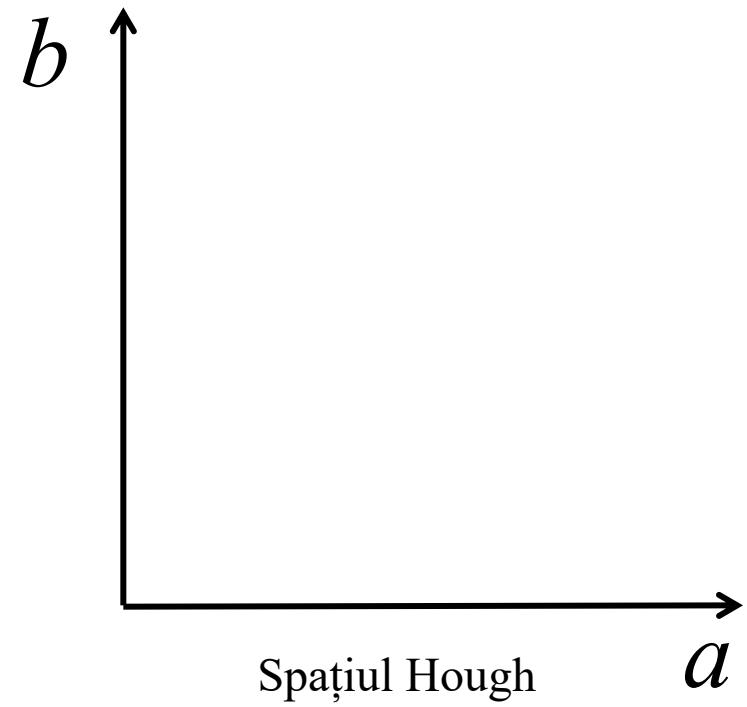
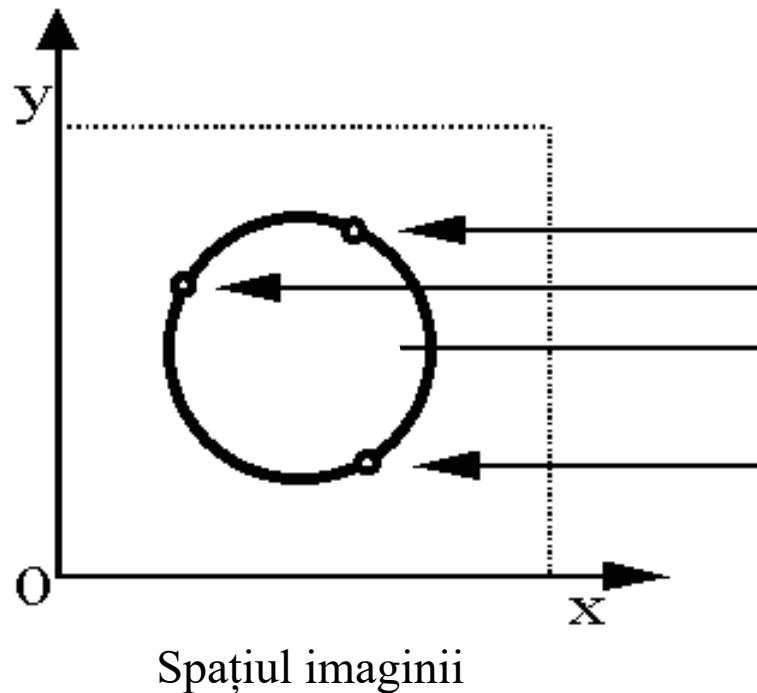
Transformata Hough pentru cercuri

- Cerc: centru (a, b) și raza r

Ecuația unui cerc?

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- Pentru raza r fixată

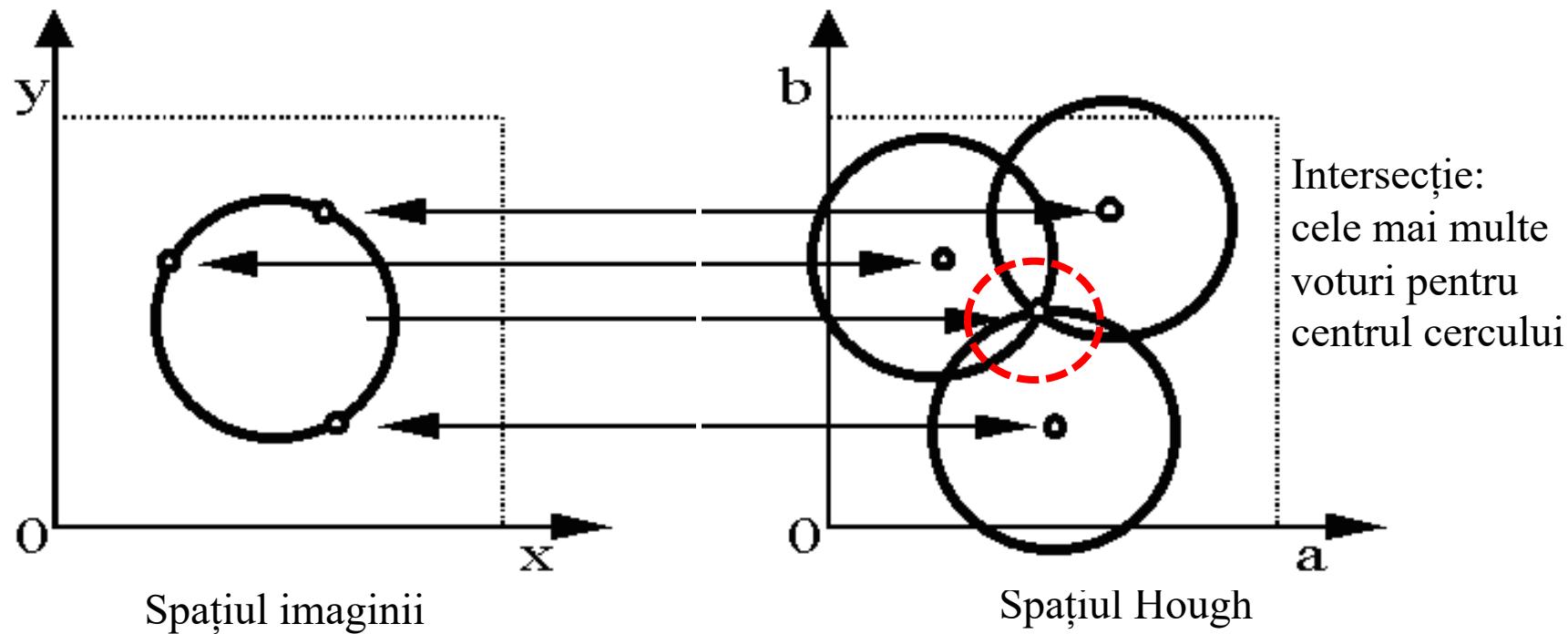


Transformata Hough pentru cercuri

- Cerc: centru (a, b) și raza r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- Pentru raza r fixată

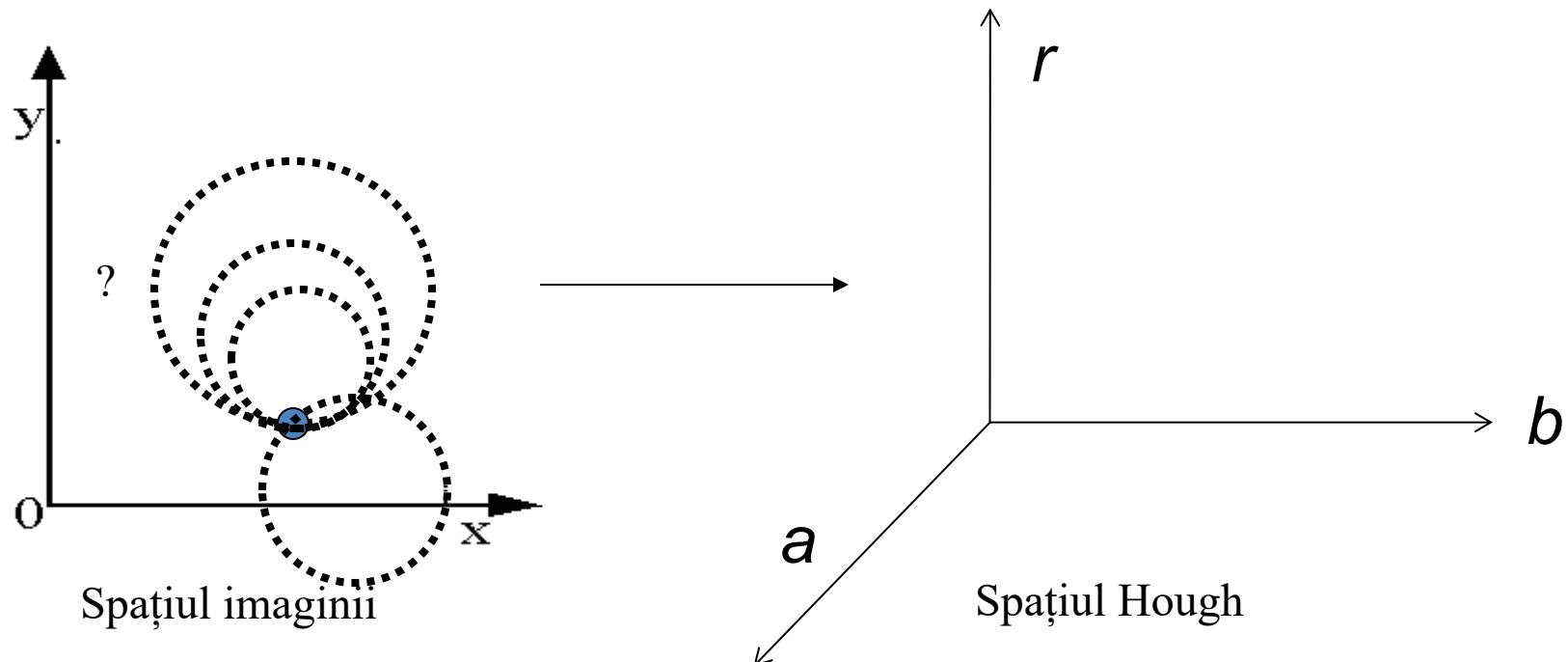


Transformata Hough pentru cercuri

- Cerc: centru (a, b) și raza r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- Pentru o rază necunoscută r

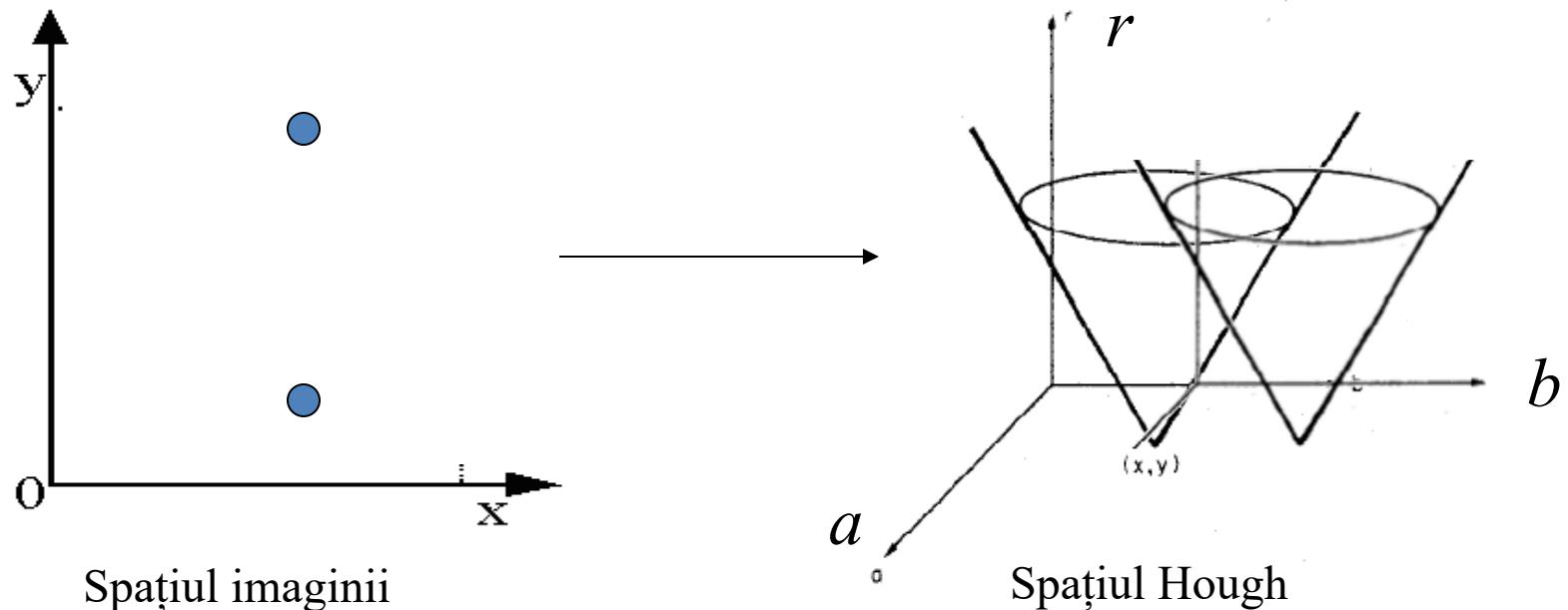


Transformata Hough pentru cercuri

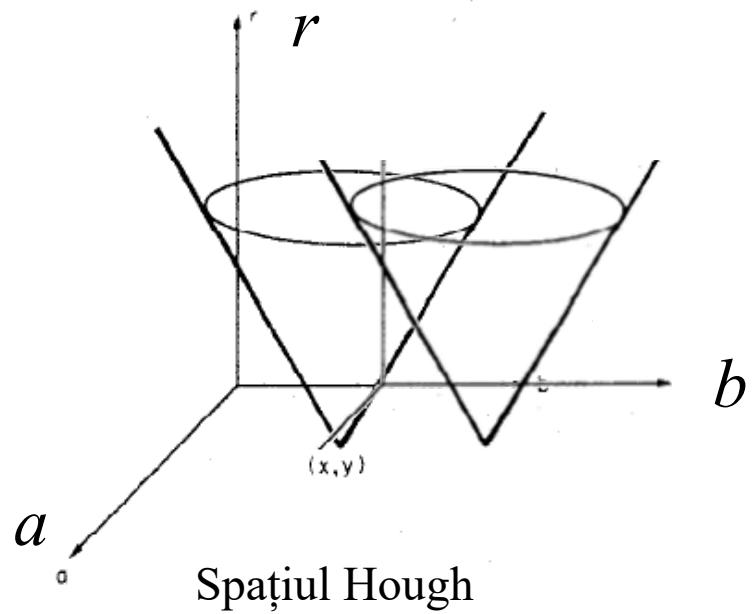
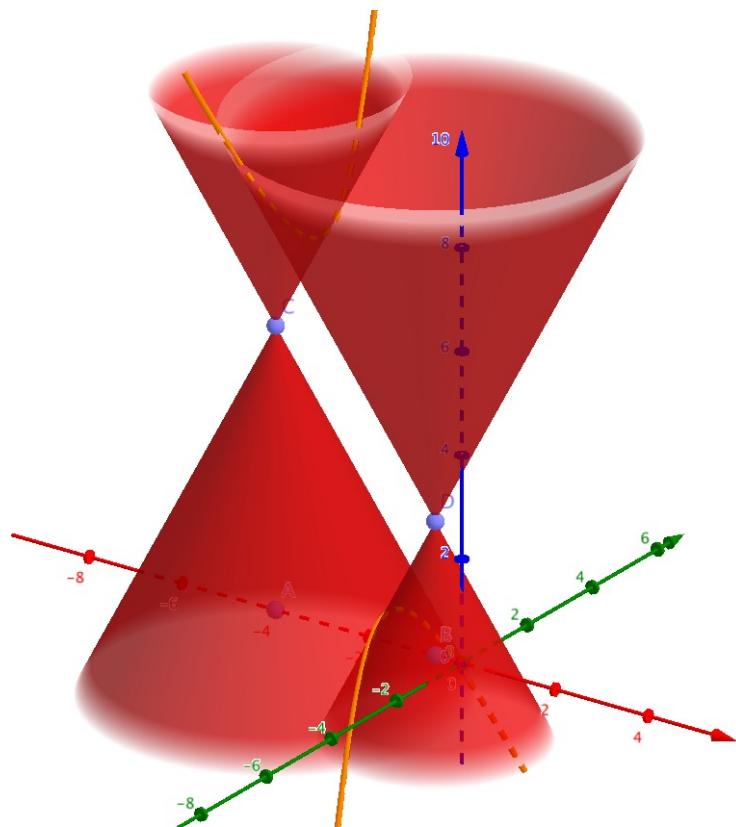
- Cerc: centru (a, b) și raza r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- Pentru o rază necunoscută r



Intersecția a două conuri

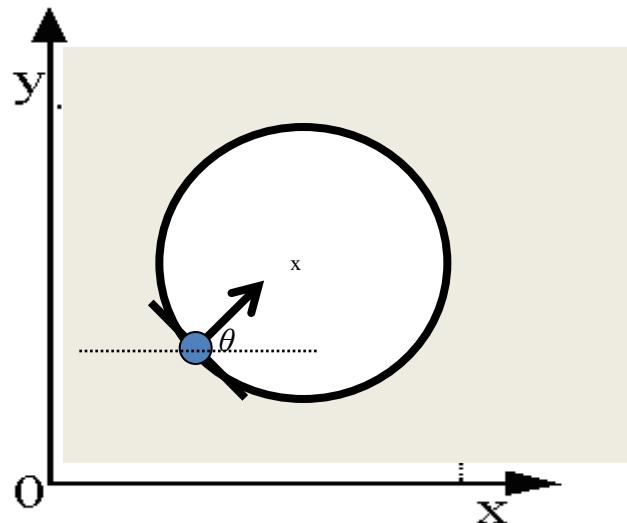


Transformata Hough pentru cercuri

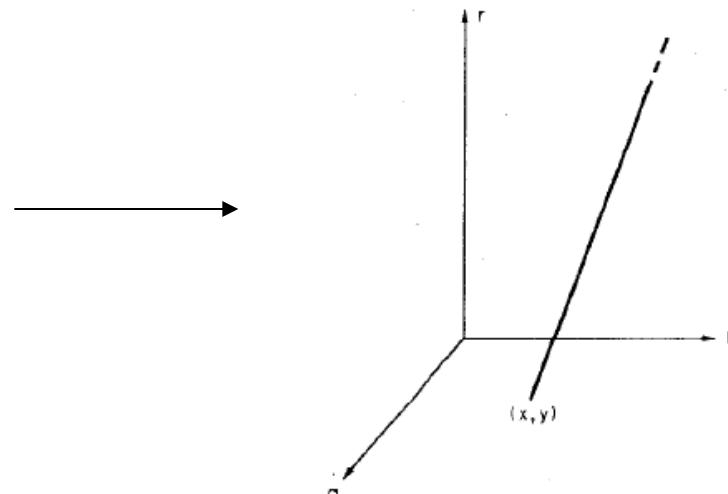
- Cerc: centru (a, b) și raza r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- Pentru o rază necunoscută r , cu direcția **cunoscută a gradientului**



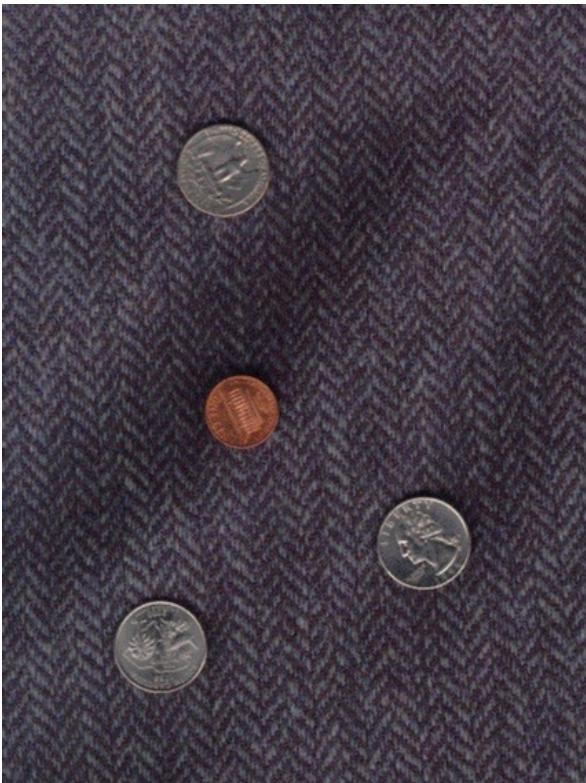
Spațiul imaginii



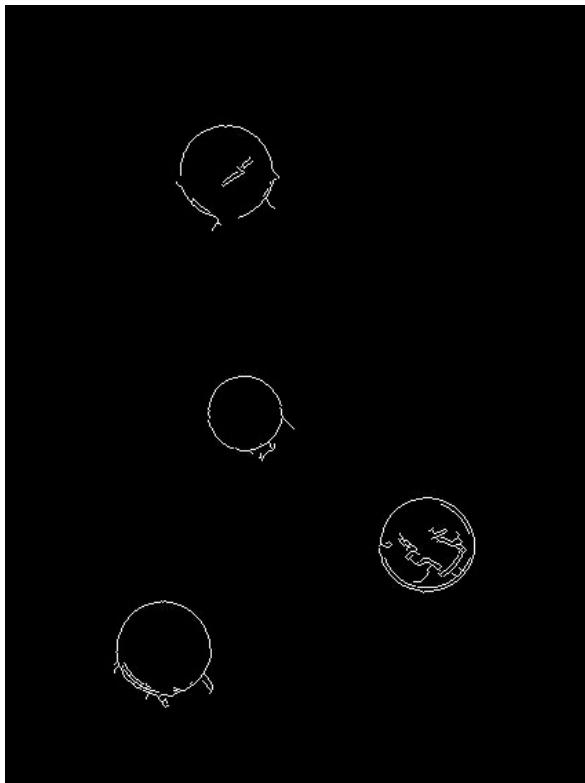
Spațiul Hough

Exemplu: detectarea cercurilor cu transformata Hough

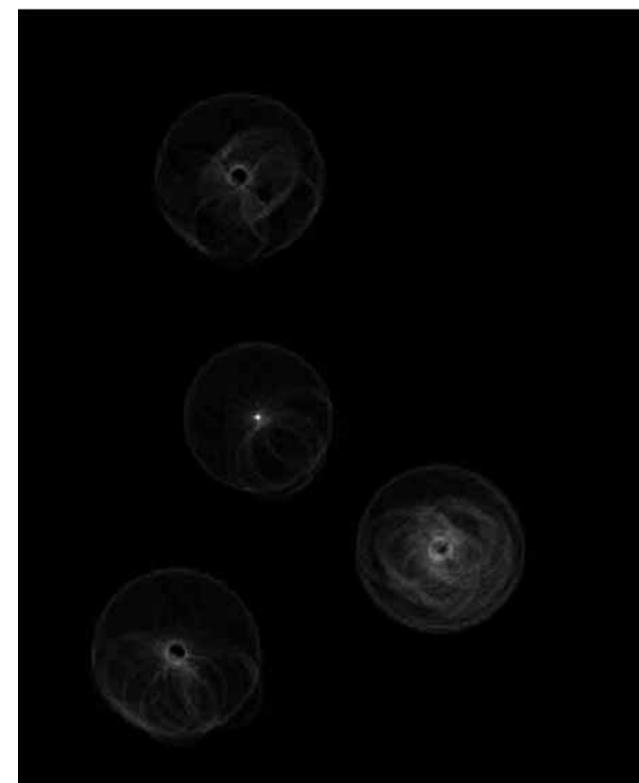
Originală



Muchii



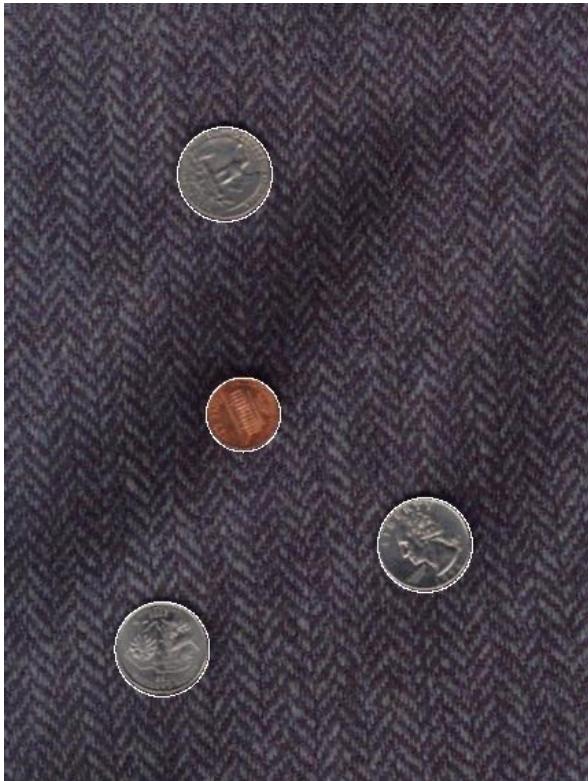
Voturi moneda 1



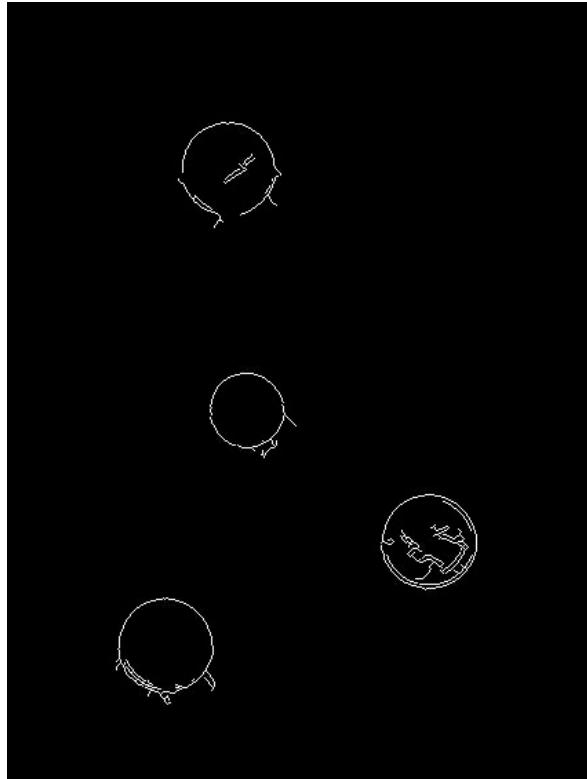
Două transformate Hough folosite, una pentru fiecare rază a monedelor.

Exemplu: detectarea cercurilor cu transformata Hough

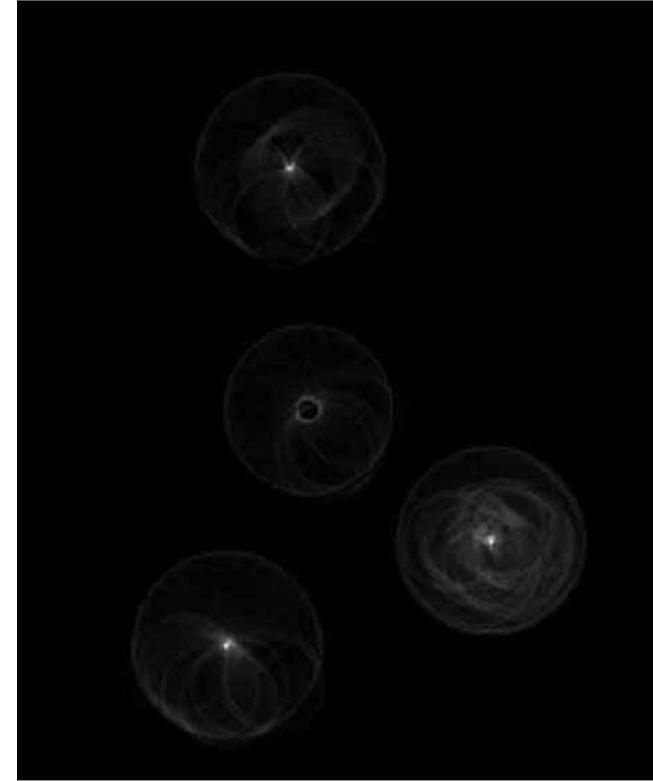
Detectare abătătoare



Muchii



Voturi moneda 2



Transformata Hough generalizată

Pattern Recognition Vol. 13, No. 2, pp. 111-122, 1981.
Printed in Great Britain.

0031-3203/81/020111-12 \$02.00/0
Pergamon Press Ltd.
© Pattern Recognition Society

GENERALIZING THE HOUGH TRANSFORM TO DETECT ARBITRARY SHAPES*

D. H. BALLARD

Computer Science Department, University of Rochester, Rochester, NY 14627, U.S.A.

(Received 10 October 1979; in revised form 9 September 1980; received for publication 23 September 1980)

Abstract—The Hough transform is a method for detecting curves by exploiting the duality between points on a curve and parameters of that curve. The initial work showed how to detect both analytic curves^(1,2) and non-analytic curves,⁽³⁾ but these methods were restricted to binary edge images. This work was generalized to the detection of some analytic curves in grey level images, specifically lines,⁽⁴⁾ circles⁽⁵⁾ and parabolas.⁽⁶⁾ The line detection case is the best known of these and has been ingeniously exploited in several applications.^(7,8,9)

We show how the boundaries of an arbitrary non-analytic shape can be used to construct a mapping between image space and Hough transform space. Such a mapping can be exploited to detect instances of that particular shape in an image. Furthermore, variations in the shape such as rotations, scale changes or figure-ground reversals correspond to straightforward transformations of this mapping. However, the most remarkable property is that such mappings can be composed to build mappings for complex shapes from the mappings of simpler component shapes. This makes the generalized Hough transform a kind of universal transform which can be used to find arbitrarily complex shapes.

Image processing
Parallel algorithms

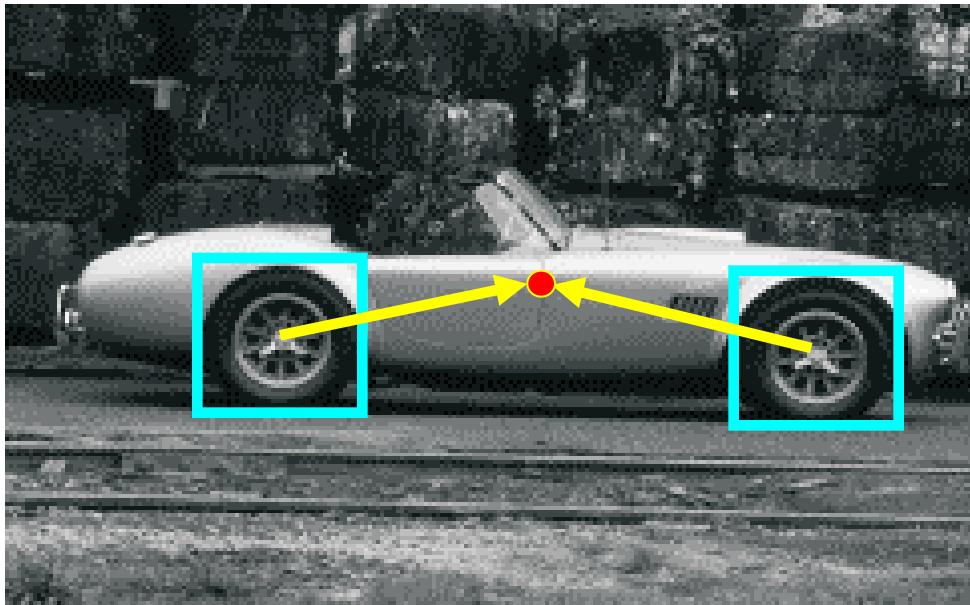
Hough transform

Shape recognition

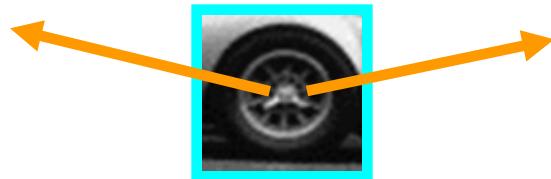
Pattern recognition

Transformata Hough generalizată

- folosește “cuvinte vizuale” care votează centrul obiectului



imagine de antrenare



“cuvânt vizual” cu vectorul de deplasare

B. Leibe, A. Leonardis, and B. Schiele, [Combined Object Categorization and Segmentation with an Implicit Shape Model](#), ECCV Workshop on Statistical Learning in Computer Vision 2004

Transformata Hough generalizată

- folosește cuvinte vizuale care votează centrul obiectului



Imagine de testare

B. Leibe, A. Leonardis, and B. Schiele, [Combined Object Categorization and Segmentation with an Implicit Shape Model](#), ECCV Workshop on Statistical Learning in Computer Vision 2004

Aplicație: detectarea linilor cu
algoritmul RANSAC

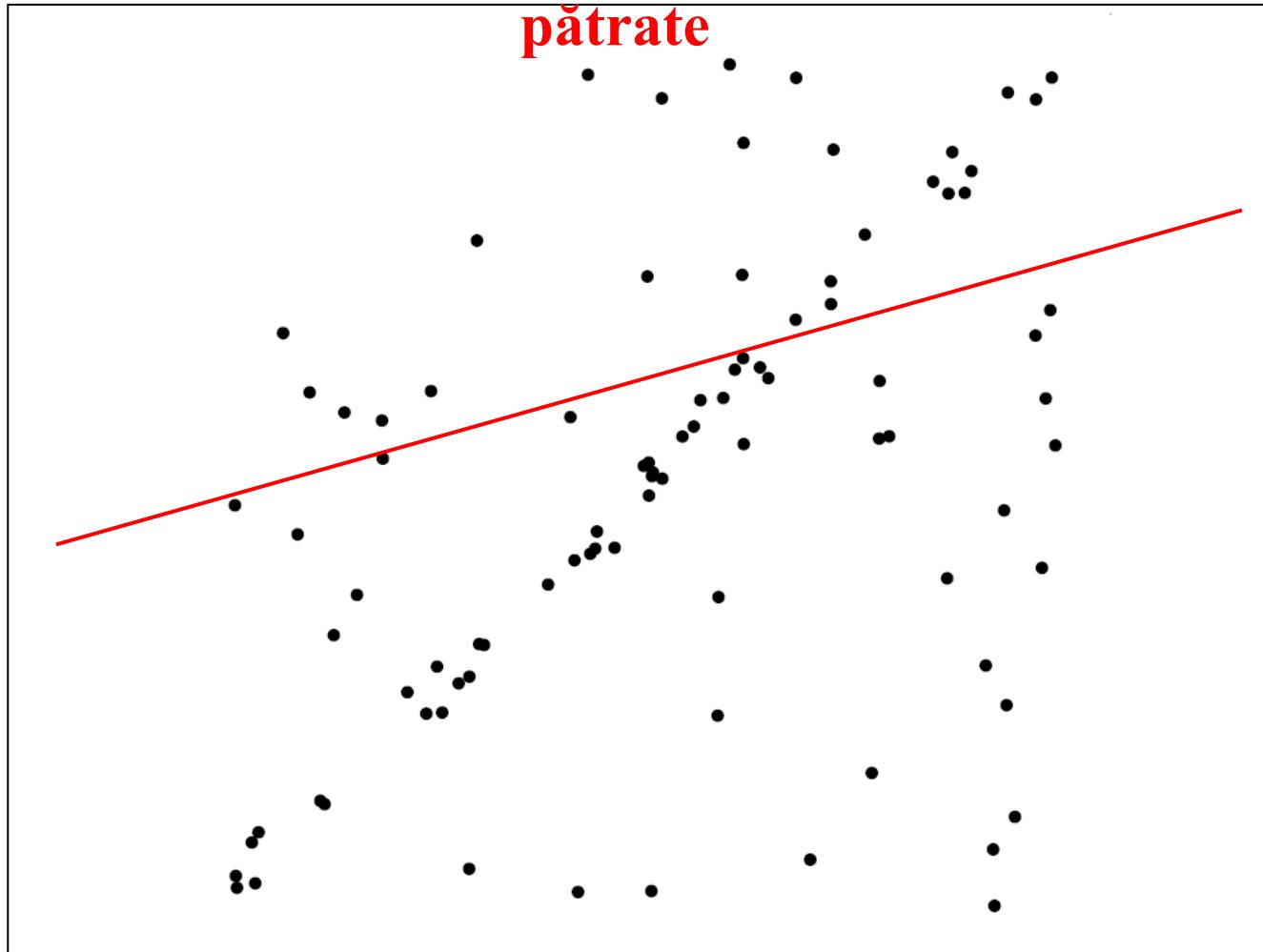
RANSAC pentru detectarea liniilor

Unde se află linia?

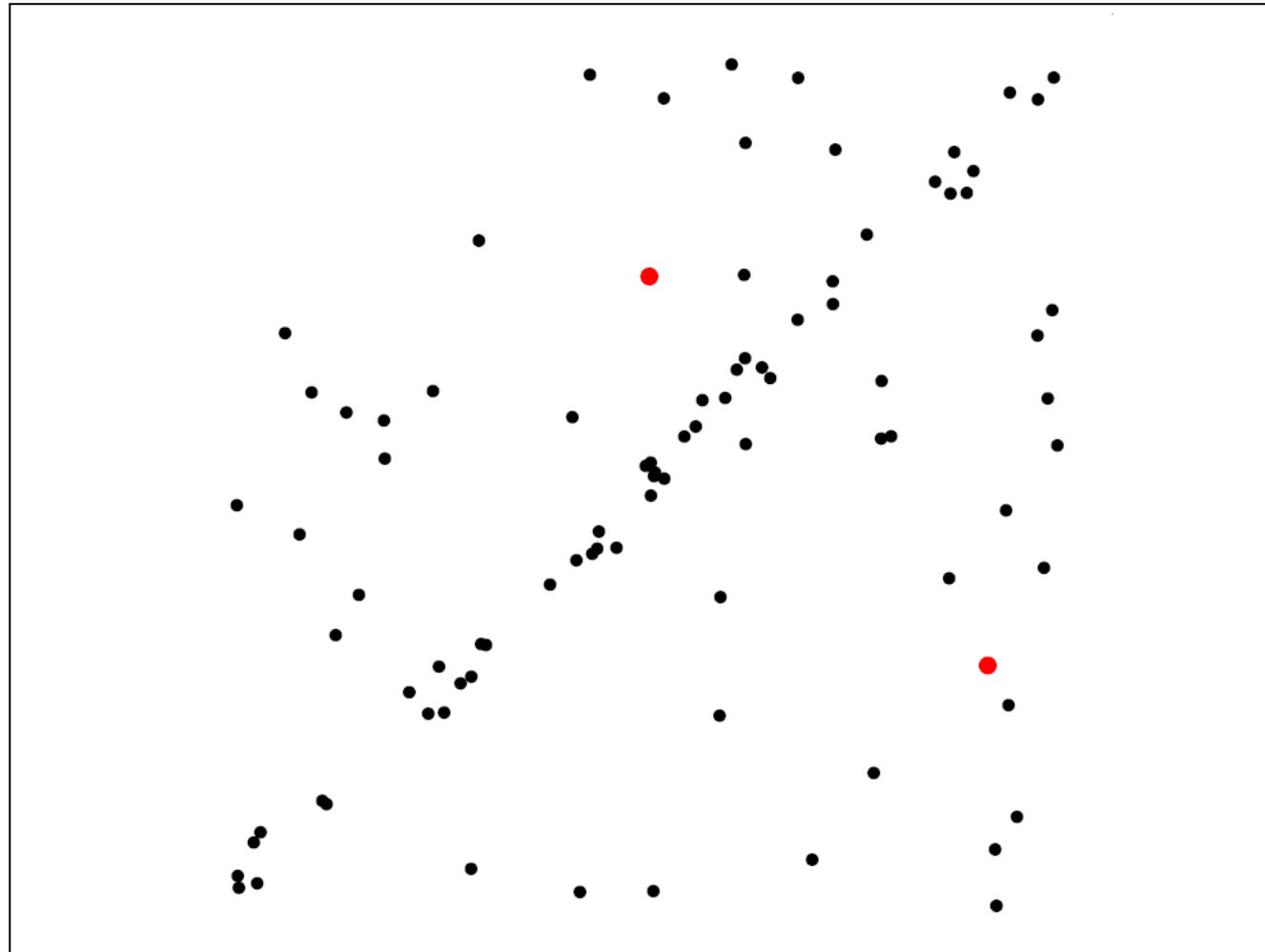


RANSAC pentru detectarea liniilor

Soluția metodei celor mai mici
pătrate

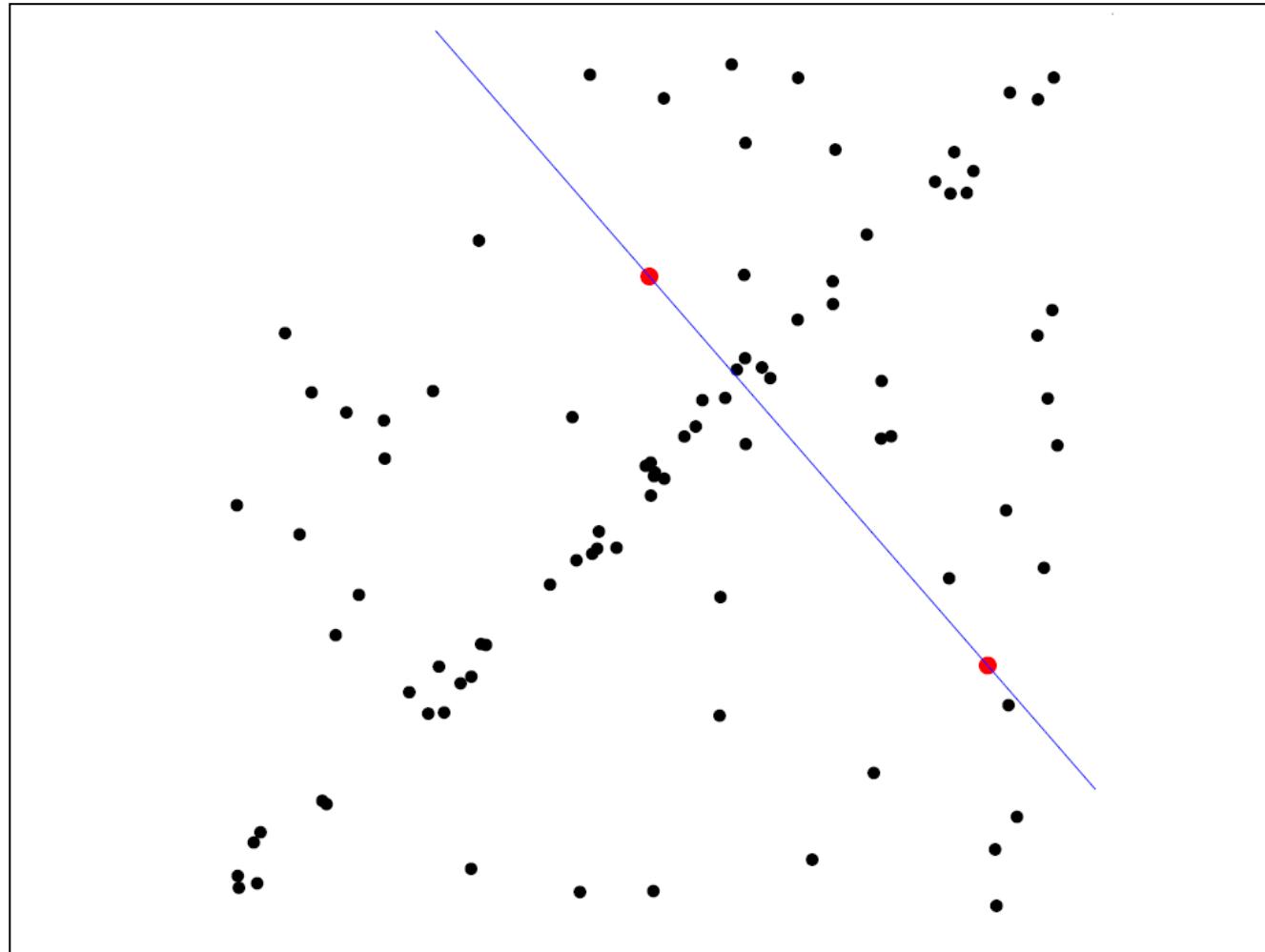


RANSAC pentru detectarea liniilor



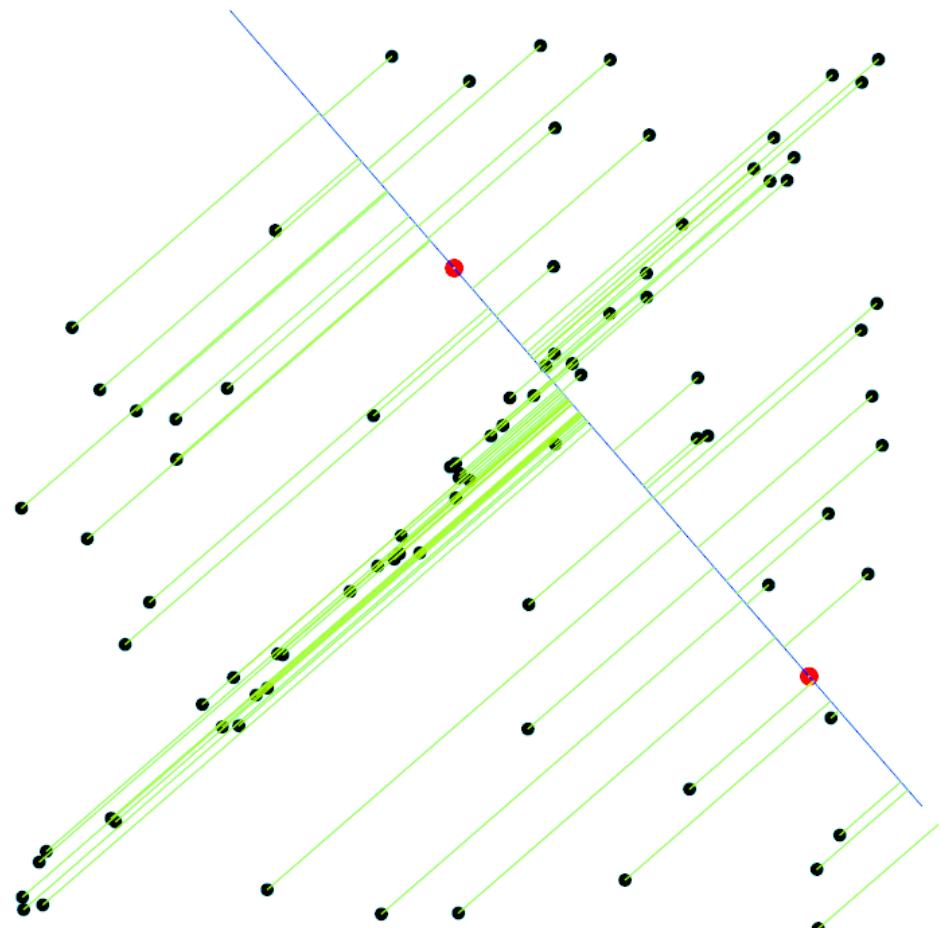
1. Selectăm aleator 2 puncte dintre toate

RANSAC pentru detectarea liniilor



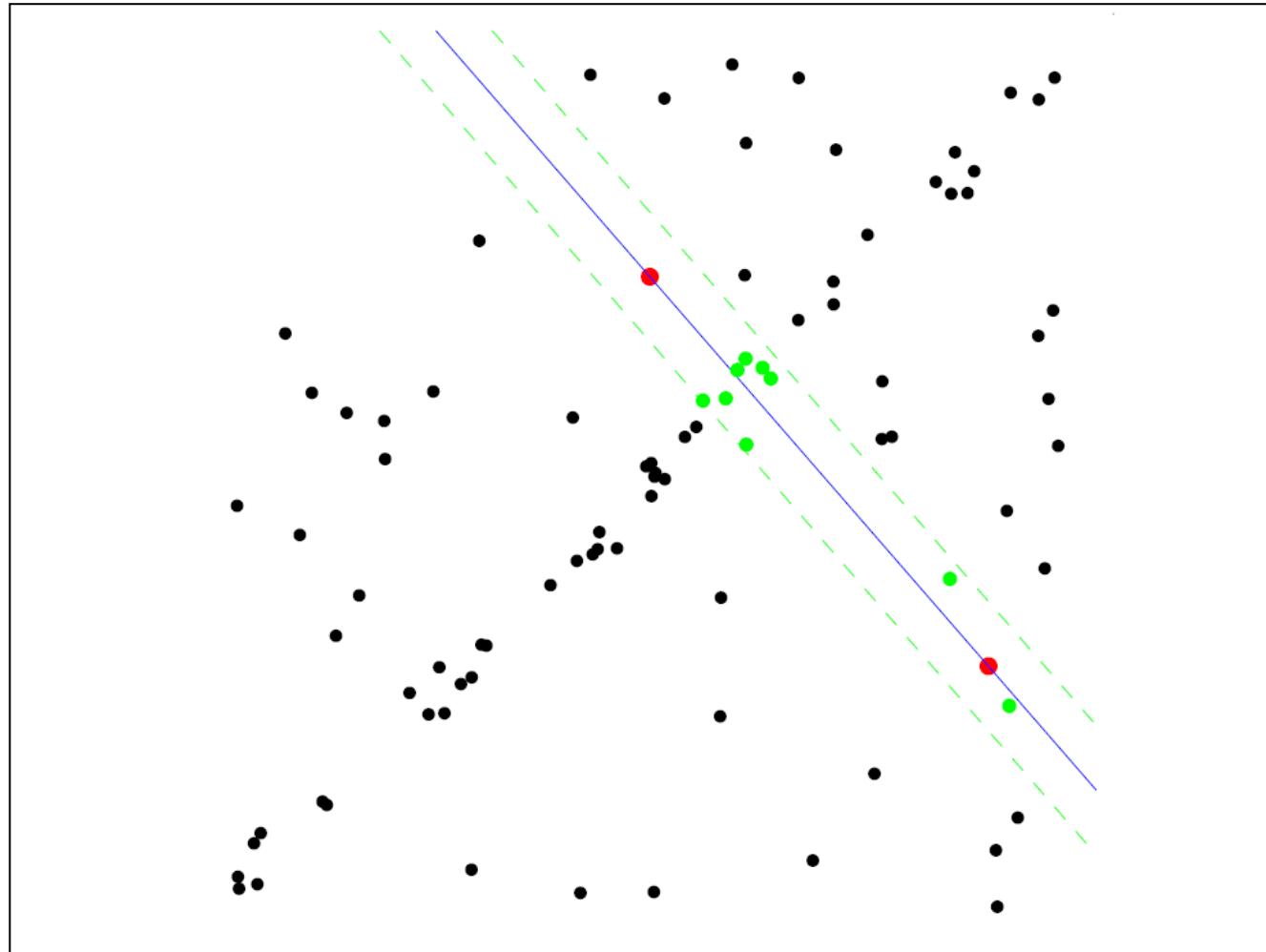
1. Selectăm aleator 2 puncte dintre toate
2. Construim dreapta

RANSAC pentru detectarea liniilor



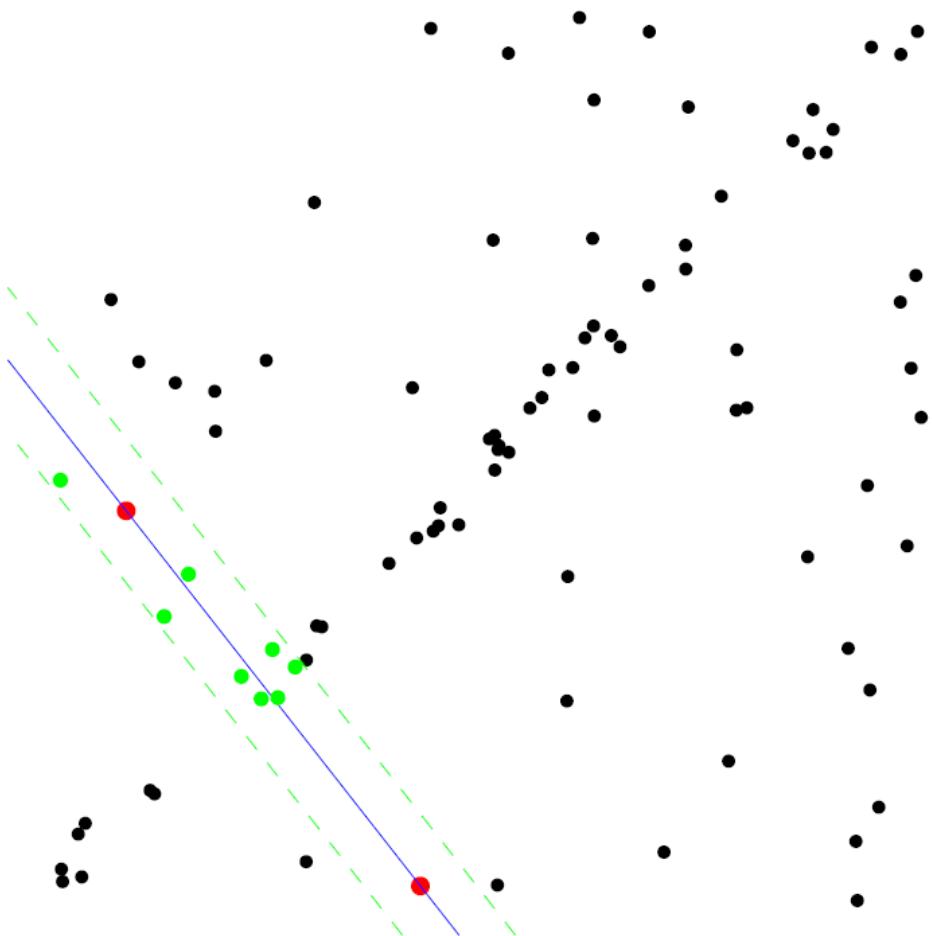
1. Selectăm aleator 2 puncte dintre toate
2. Construim dreapta
3. Calculăm funcția de eroare

RANSAC pentru detectarea liniilor



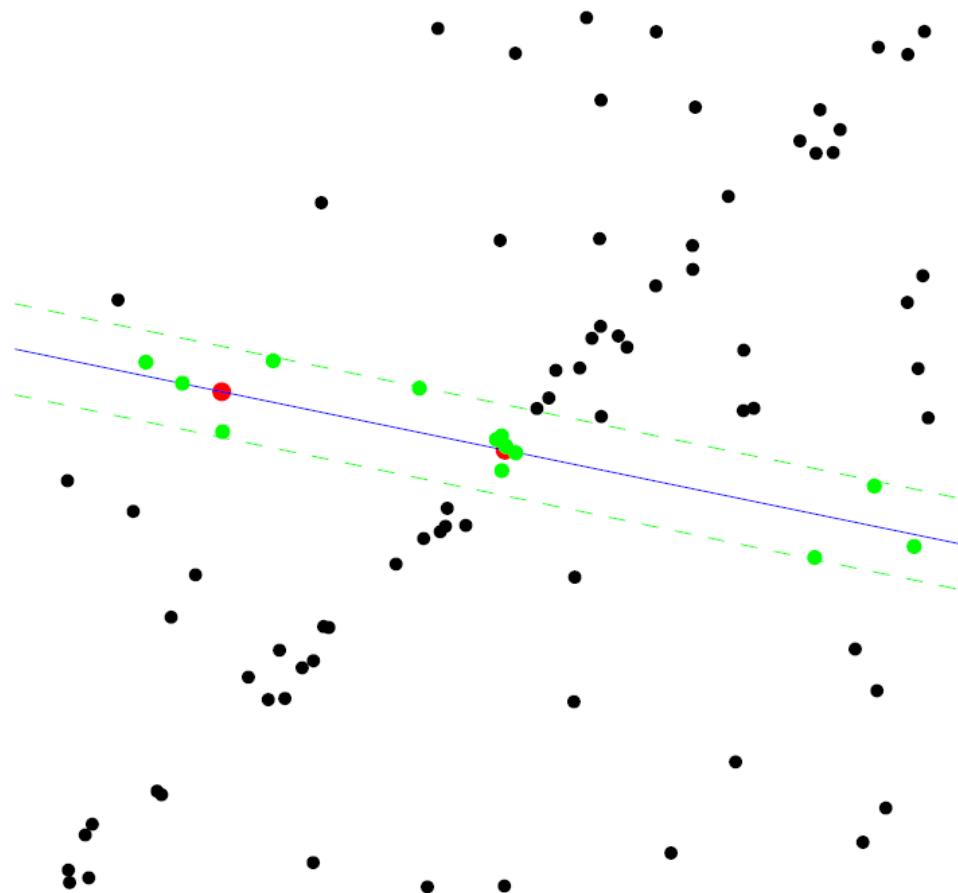
1. Selectăm aleator 2 puncte dintre toate
2. Construim dreapta
3. Calculăm funcția de eroare
4. Numărăm punctele inlier consistente

RANSAC pentru detectarea liniilor



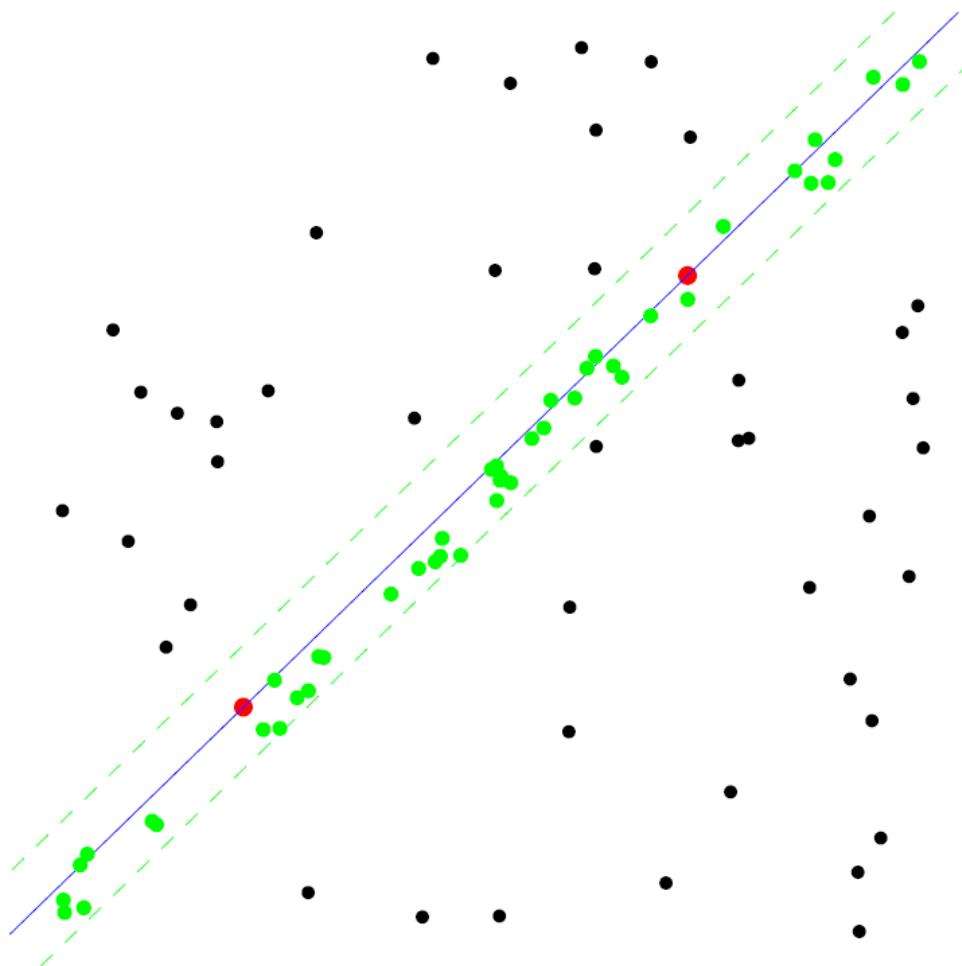
1. Selectăm aleator 2 puncte dintre toate
2. Construim dreapta
3. Calculăm funcția de eroare
4. Numărăm punctele inlier consistentă
5. Repetă 1-4 pentru un număr de ori dat

RANSAC pentru detectarea liniilor



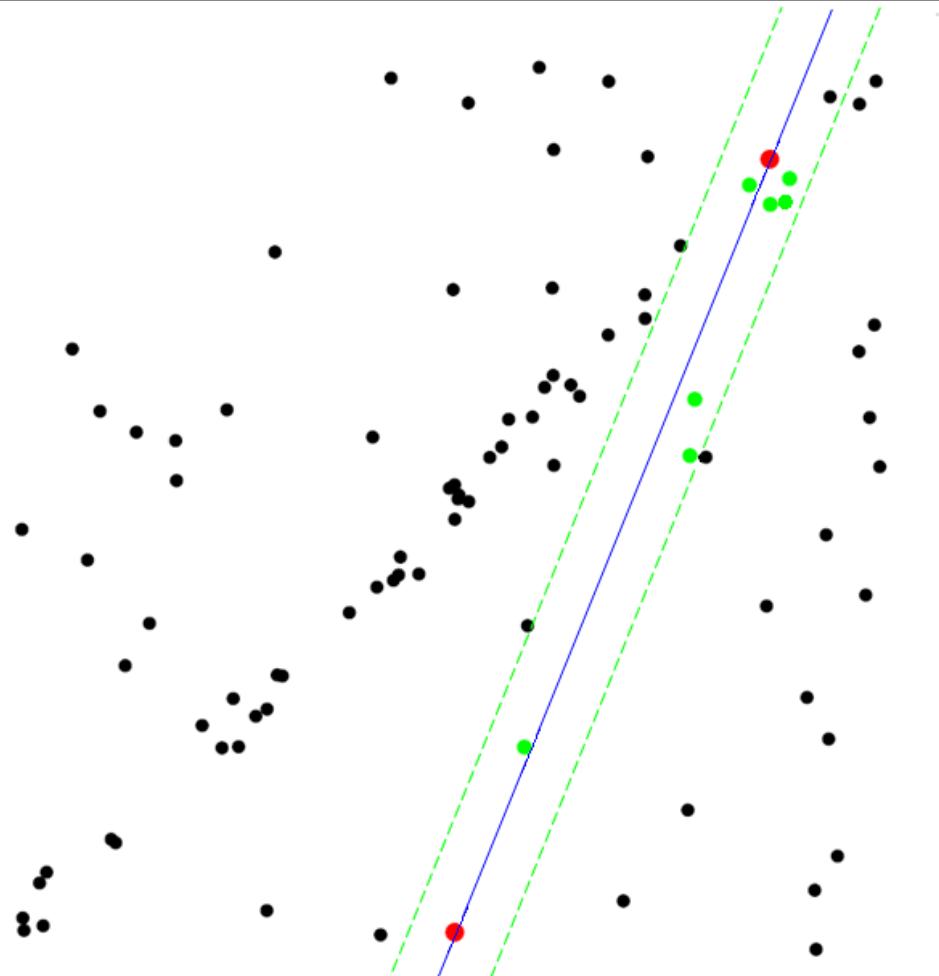
1. Selectăm aleator 2 puncte dintre toate
2. Construim dreapta
3. Calculăm funcția de eroare
4. Numărăm punctele inlier consistentă
5. Repetă 1-4 pentru un număr de ori dat

RANSAC pentru detectarea liniilor



1. Selectăm aleator 2 puncte dintre toate
2. Construim dreapta
3. Calculăm funcția de eroare
4. Numărăm punctele inlier consistentă
5. Repetă 1-4 pentru un număr de ori dat

RANSAC pentru detectarea liniilor



1. Selectăm aleator 2 puncte dintre toate
2. Construim dreapta
3. Calculăm funcția de eroare
4. Numărăm punctele inlier consistentă
5. Repetă 1-4 pentru un număr de ori dat

RANSAC pentru detectarea liniilor

- RANdom SAmple Consensus
- **idee principală**: vrem să evităm impactul punctelor outlier, ne concentrăm să găsim punctele “inlier”, și le folosim doar pe acestea la găsirea parametrilor modelului
- **intuiție**: dacă un punct outlier este inclus în dreaptă, numărul de puncte inlier va fi foarte mic (ipoteza nu este susținută de date)

RANSAC pentru detectarea liniilor

Repetă de N ori:

1. Selectează 2 puncte la întâmplare
2. Găsește ecuației dreptei care trece prin aceste două puncte
3. Găsește punctele inlier pentru această ipoteză din cele rămase: punctele a căror distanță față de dreaptă este $< t$ (threshold)
4. Dacă există mai mult de d puncte inlier, acceptă ipoteza. Rafinează ipoteza pe baza tuturor punctelor inlier.
(Păstrează ipoteza cu cea mai mare număr de puncte inlier)

Algoritmul RANSAC: avantaje și dezavantaje

Avantaje

- simplu și general
- aplicabil în multe probleme (detectarea liniilor, construirea de panorame: găsirea de corespondențe – calculul homografiei - 8 parametri)
- funcționează destul de bine în practică

Dezavantaje

- numărul de parametri ai unui model poate fi foarte mare
- nu merge bine când numărul de puncte inlier este foarte mic (raportul inlier/outlier este mic)
- numărul minim de puncte ce definește un model nu furnizează întotdeaună ipoteze bune

Algoritmul RANSAC: ce N alegem?

- de câte ori **N** trebuie să repetăm RANSAC pentru a fi siguri (cu o probabilitate mare) că am găsit cea mai bună ipoteză?
- presupunem că procentul de puncte provenite din dreaptă din numărul total de puncte este **r** (= #puncte inlier/nr puncte total)
- o ipoteză este definită de **k** puncte (**k=2** pentru dreaptă)
- probabilitatea ca o selecție aleatoare de **k** puncte să fie corectă **r^k**
- probabilitatea ca o selecție aleatoare de **k** puncte să fie greșită **$1 - r^k$**
- probabilitatea ca toate selecțiile să fie greșite: **$(1 - r^k)^N$**
- probabilitatea ca cel puțin o selecție să fie corectă: **$1 - (1 - r^k)^N$**
- alege **N** astfel încât **$p = 1 - (1 - r^k)^N$** este foarte mare ($1-0.01=0.99$)

RANSAC: Calculul lui N (p=0.99)

k	Proportia de outlieri (1-r)						
	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

Aplicații – detectarea liniilor



https://www.youtube.com/watch?v=SFqAAseL_1g

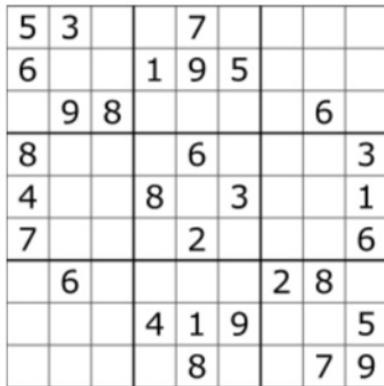
Aplicații - Extragerea informațiilor vizuale din careuri Sudoku

Sudoku is currently one of the most famous puzzles in the world. The most popular version consists on a 9×9 grid made up of 3×3 subgrids, but the general case, an $n^2 \times n^2$ grid with $n \times n$ subgrids is considered. Some cells contain numbers, which can be considered as input data. The goal is to fill in the empty cells, one number in each, so that each column, row, and subgrid contains the numbers 1 through 9 exactly once (numbers 1 to n^2 in the general case). If the input data are correct, the sudoku has one and only one solution.

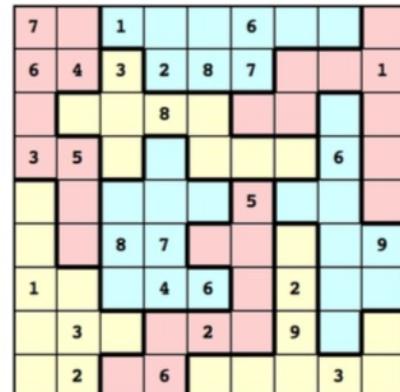
5	3		7					
6			1	9	5			
	9	8				6		
8			6				3	
4		8	3				1	
7			2			6		
	6			2	8			
		4	1	9			5	
		8			7	9		

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

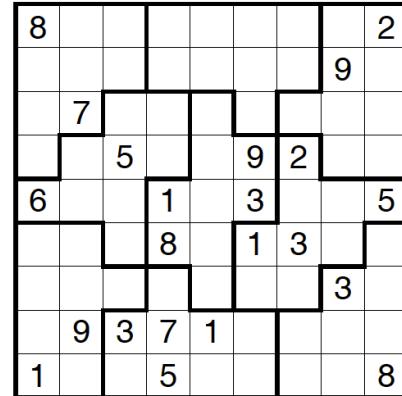
Multe variante de careuri Sudoku



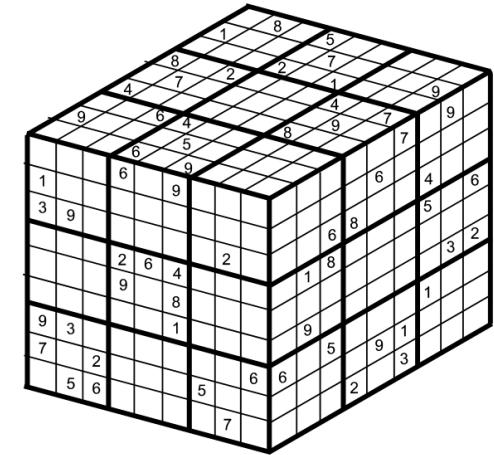
classic



jigsaw



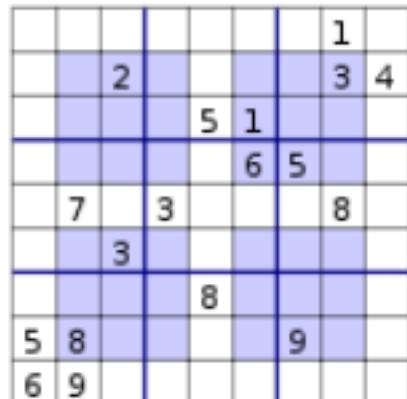
jigsaw



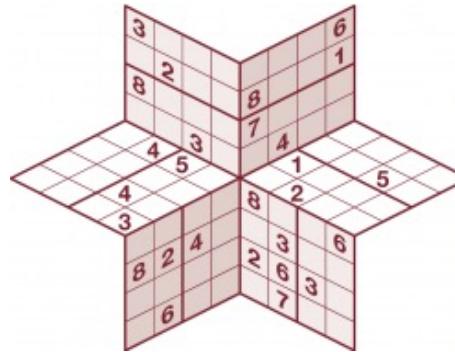
cube



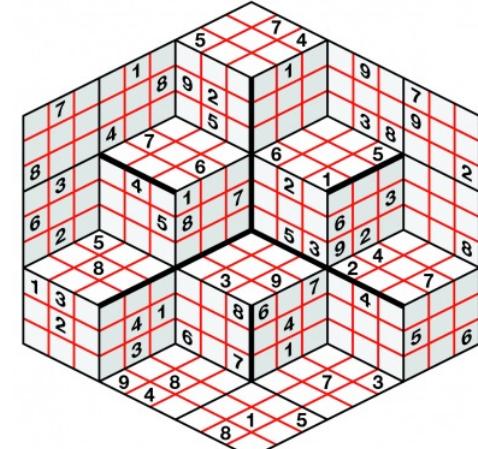
wordoku



hypersudoku



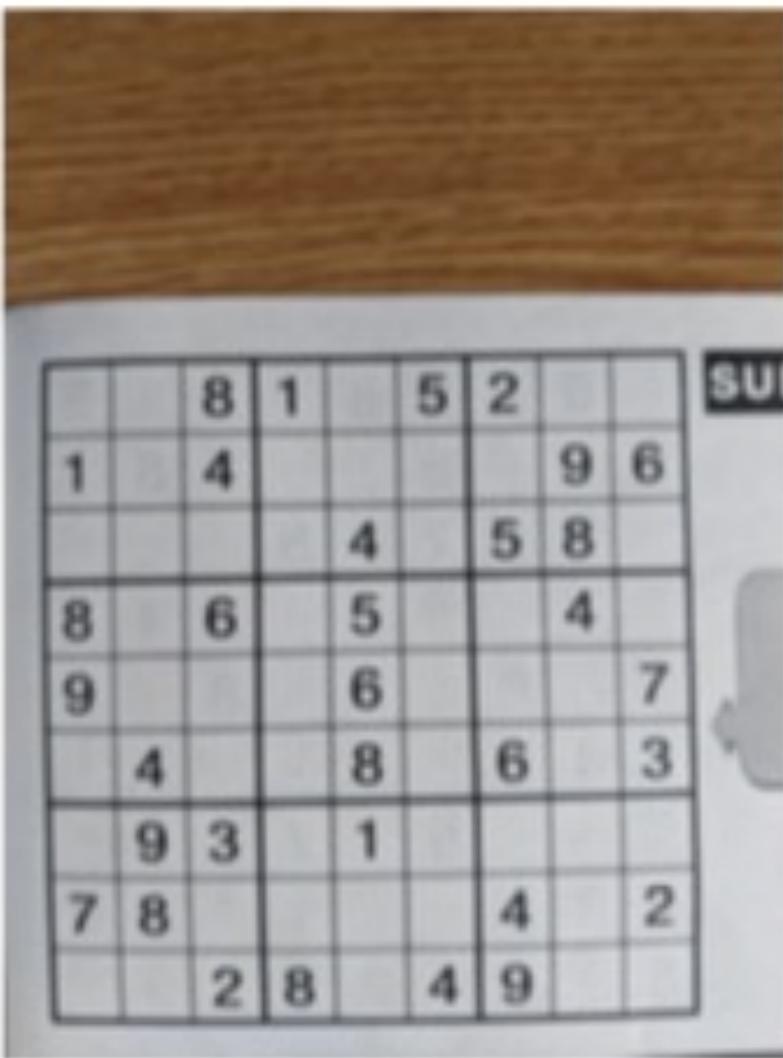
3D star



3D

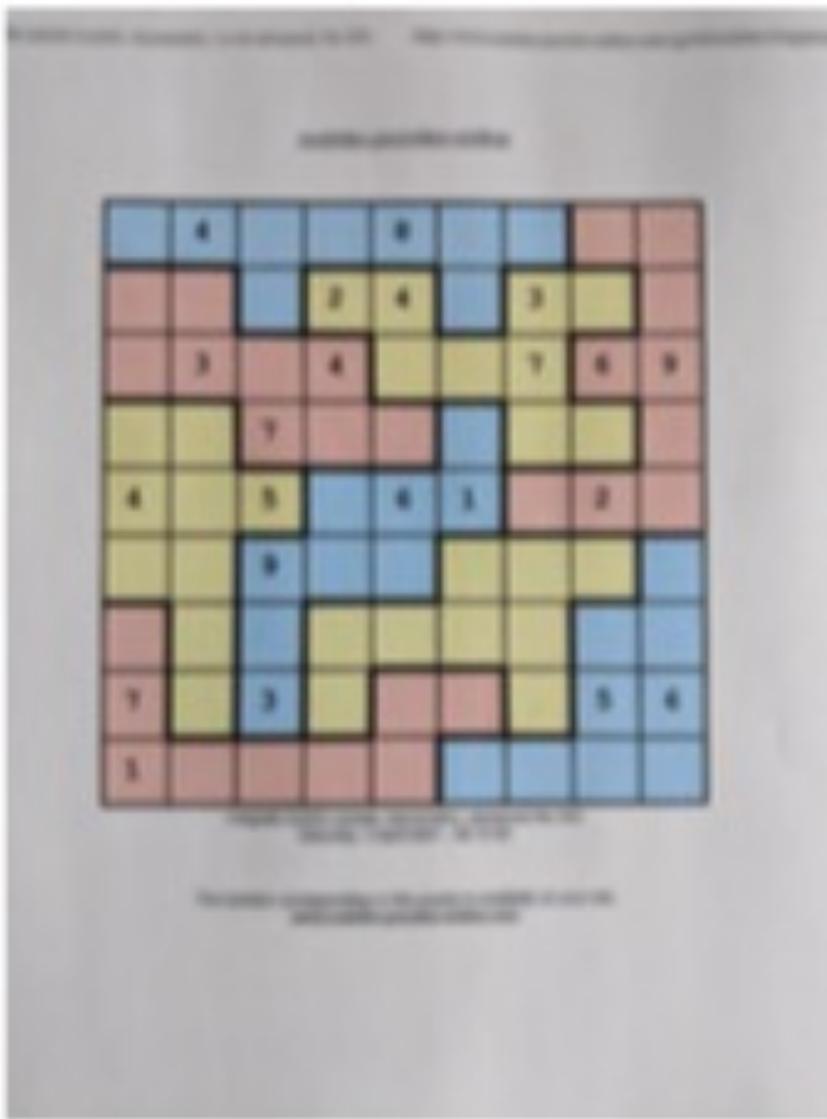
- many more...

Classic Sudoku



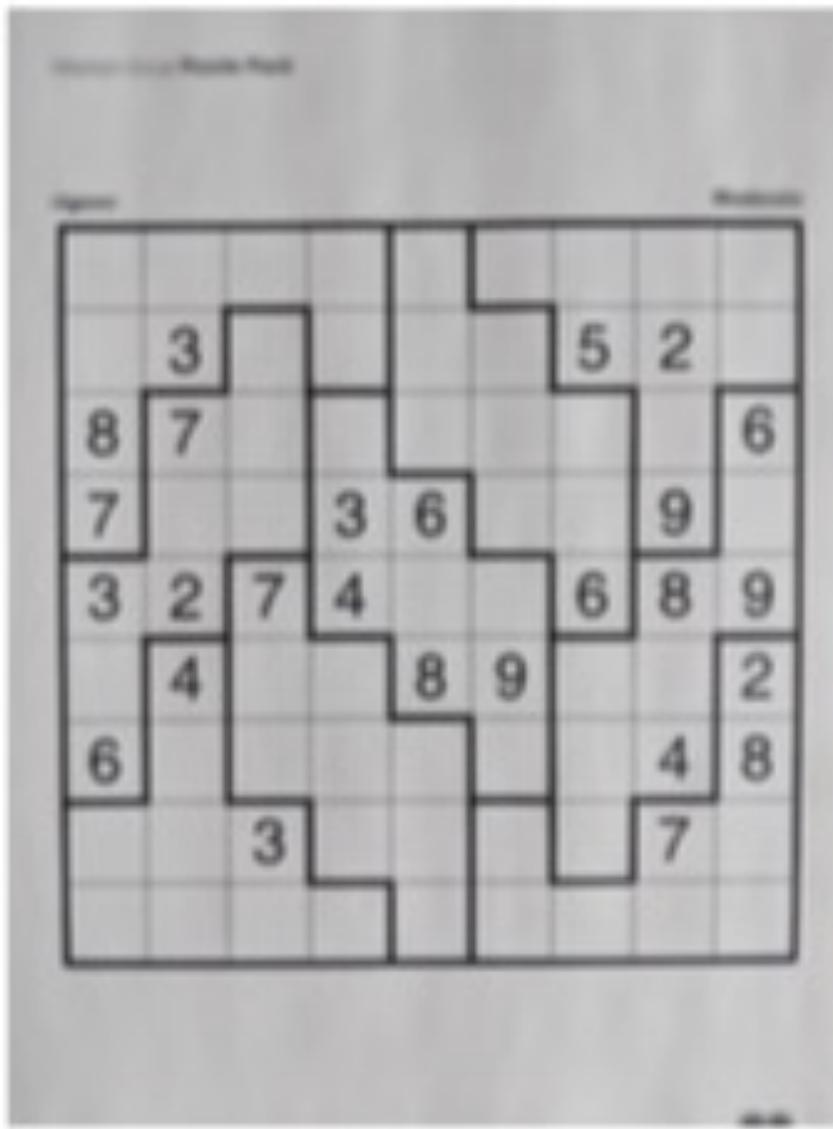
00XX0XX00
X0X0000XX
0000X0XX0
X0X0X00X0
X000X000X
0X00X0X0X
0XX0X0000
XX0000X0X
00XX0XX00

Jigsaw Sudoku



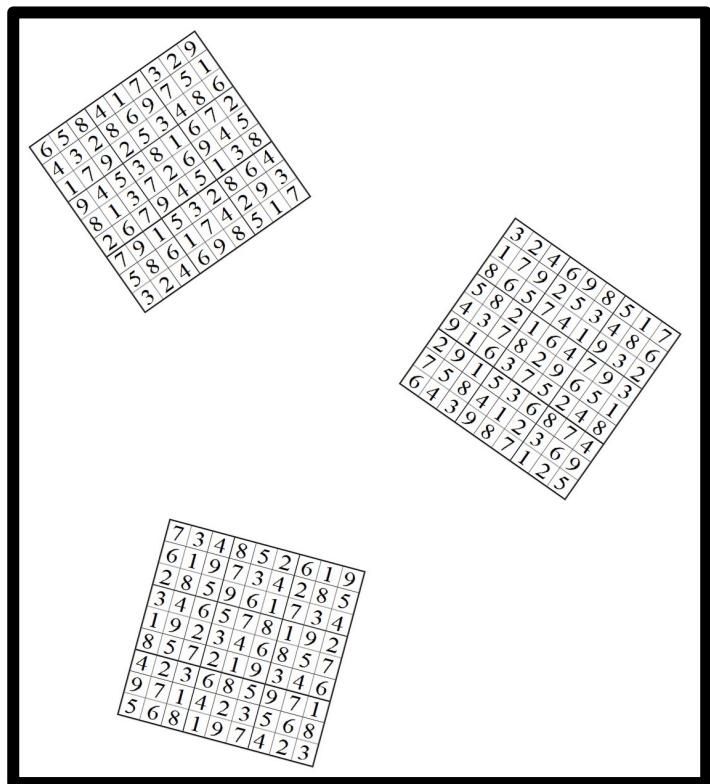
1o1x1o1o1x1o1o2o2o
3o3o1o4x4x1o4x4o2o
3o3x3o3x4o4o4x2x2x
5o5o3x3o3o6o4o4o2o
5x5o5x6o6x6x2o2x2o
5o5o6x6o6o7o7o7o8o
9o5o6o7o7o7o7o8o8o
9x5o6x7o9o9o7o8x8x
9x9o9o9o9o8o8o8o8o8o

Jigsaw Sudoku

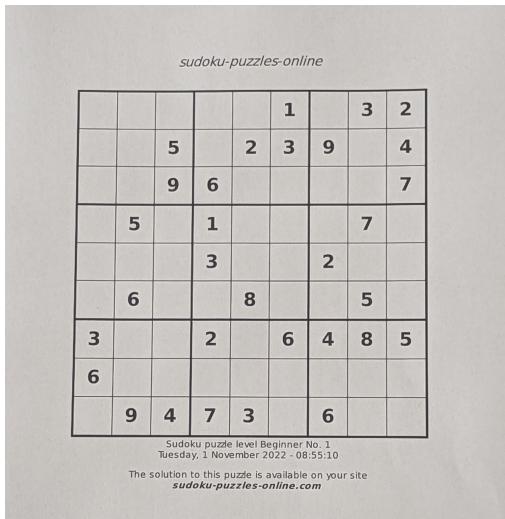


1o1o1o1o2o3o3o3o3o
1o1x4o1o2o2o3x3x3o
1x4x4o5o2o2o2o3o6x
1x4o4o5x5x2o2o3x6o
4x4x7x5xmo5o2x6x6x
4o8x7o7o5x5x6o6o9x
4x8o7o7o7o5o6o6x9x
8o8o8x7o7o9o6o9x9o
8o8o8o8o7o9o9o9o9o

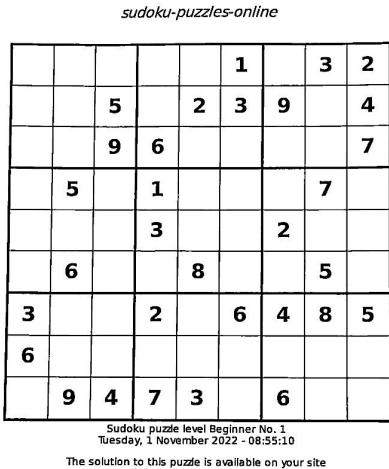
Sudoku Cube



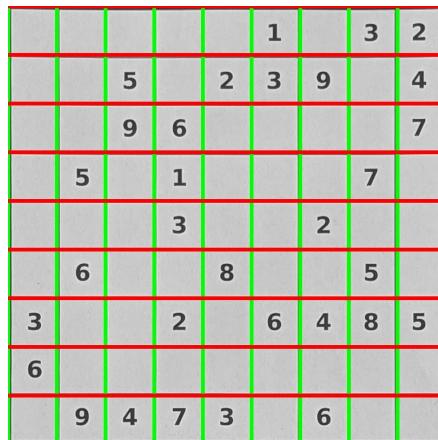
Aplicații: laborator săptămâna 6



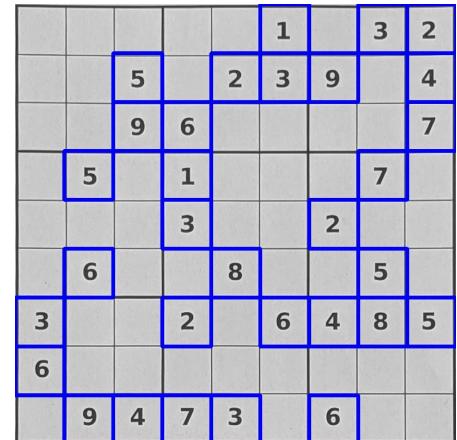
Imagine inițială



Gaussian blur + eroziune
(operator morfologic)



Canny edge detector

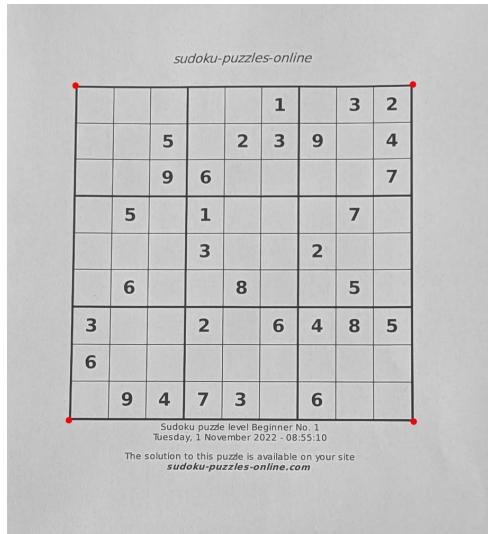


Perspectivă + detectare de linii

Cel mai mare contur închis

Clasificare celule

Aplicații: laborator săptămâna 6



Cel mai mare contur închis

					1	3	2	
			5	2	3	9	4	
	9	6					7	
5		1				7		
	3			2				
6		8			5			
3		2	6	4	8	5		
6								
9	4	7	3	6				

Perspectivă + detectare de linii

					1	3	2	
			5	2	3	9	4	
	9	6					7	
5		1				7		
	3			2				
6		8			5			
3		2	6	4	8	5		
6								
9	4	7	3	6				

Clasificare celule
filled vs empty

```
[[ 'o', 'o', 'o', 'o', 'o', '1', 'o', '3', '2'],
 ['o', 'o', '5', 'o', '2', '3', '9', 'o', '4'],
 ['o', 'o', '9', '6', 'o', 'o', 'o', 'o', '7'],
 ['o', '5', 'o', '1', 'o', 'o', 'o', '7', 'o'],
 ['o', 'o', 'o', '3', 'o', 'o', '2', 'o', 'o'],
 ['o', '6', 'o', 'o', '8', 'o', 'o', '5', 'o'],
 ['3', 'o', 'o', '2', 'o', '6', '4', '8', '5'],
 ['6', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o'],
 ['o', '9', '4', '7', '3', 'o', '6', 'o', 'o]]
```

Recunoaștere cifre în celule

Aplicații: lucrare de licență

Cuprins

Lucrare de licență

SOLUȚIE PENTRU SUDOKU 3D, BAZATĂ PE COMPUTER VISION

Absolvent

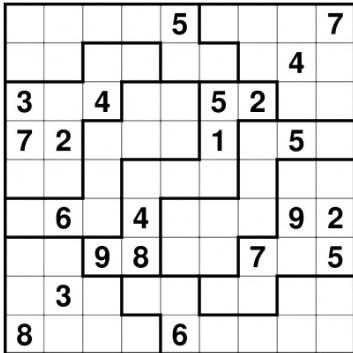
Rusu Andrei-Cristian

¹³Coordonator științific

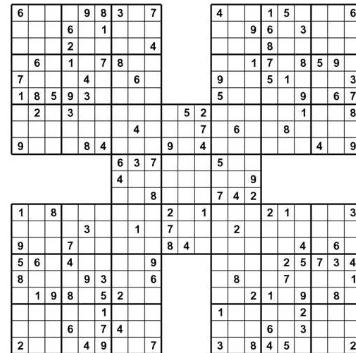
Conf. Dr. Alexe Bogdan

1	Introducere	5
1.1	Informații istorice	5
1.2	Sudoku 3D	6
2	Baza de date și preprocesarea datelor	8
3	Terminologie și convenții	11
3.1	Axele	11
3.2	Notația punctelor fețelor	11
3.3	Segmentele îngroșate	12
4	Extragerea informațiilor din imagine	13
4.1	Calcularea coordonatelor vârfurilor cubului	13
4.2	Identificarea segmentelor îngroșate	14
4.3	Extragerea vectorilor de deplasare a fețelor	15
4.4	Determinarea existenței unei fețe definite de patru puncte și extragerea acesteia	15
4.5	Extragerea drumurilor folosind segmentele îngroșate	18
4.5.1	Extragerea punctelor de început ale fețelor	18
4.5.2	Algoritmul	21
4.5.3	Formulele pentru calculul punctelor fețelor din drumuri	21
4.6	Extragerea cifrelor din imagini cu fețe	27
4.6.1	Preprocesarea imaginii	27
4.6.2	Extragerea informației din celule	28
4.6.3	Clasificarea cifrelor din imagini	29
5	Generarea soluției	31
5.1	Algoritmul	31
5.2	Verificarea validității unei configurații	33
5.3	Rezultate	34
6	Concluzie	37
	Bibliografie	38

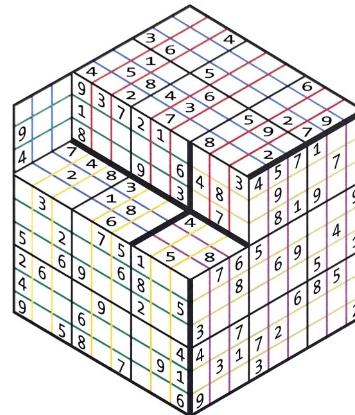
Aplicații: lucrare de licență



(a)



(b)



(c)

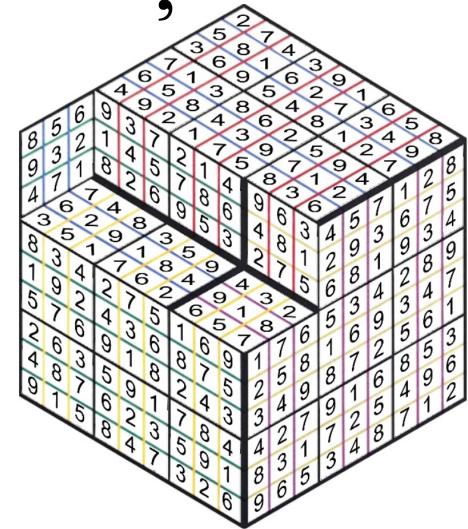
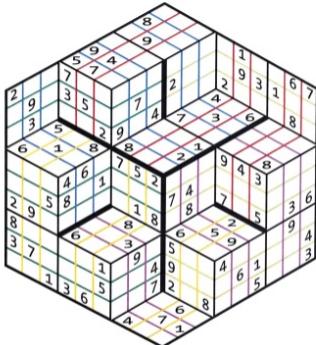
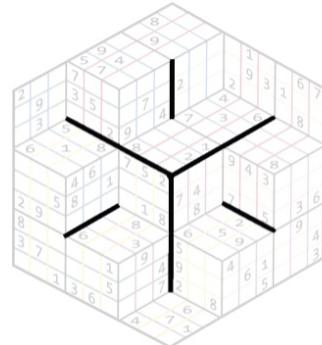


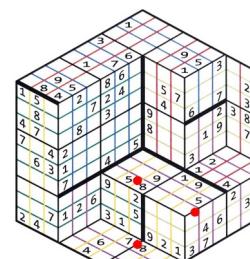
Figura 1.4: Soluția pentru configurația din **Figura 1.3c**



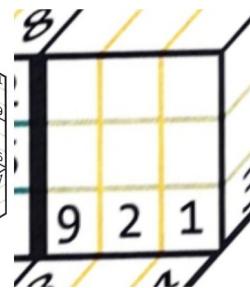
(a)



(b)



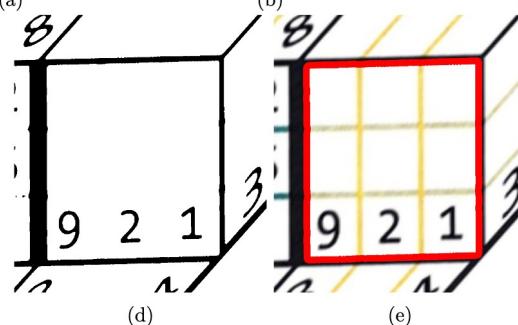
(a)



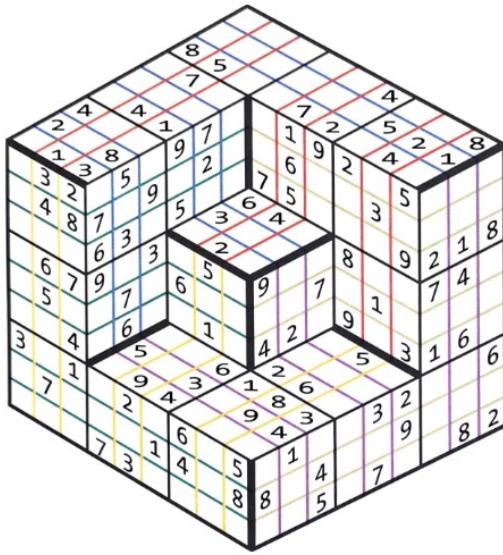
(a)



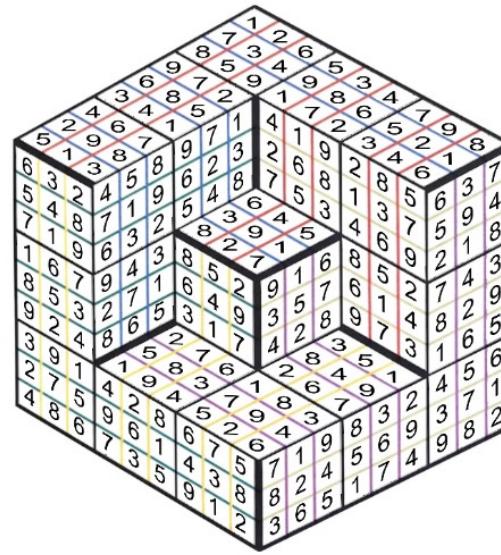
(c)



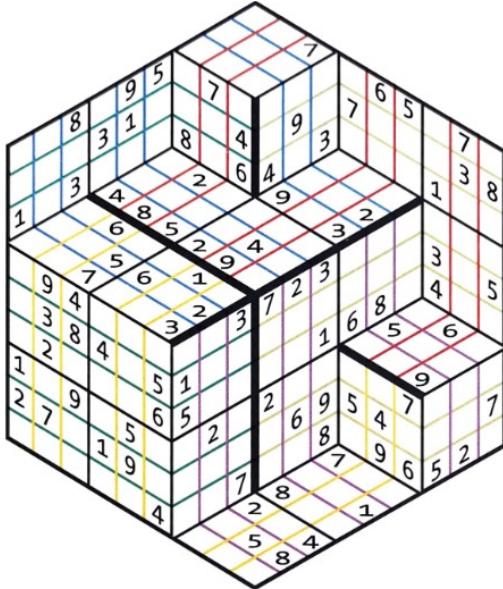
(d)



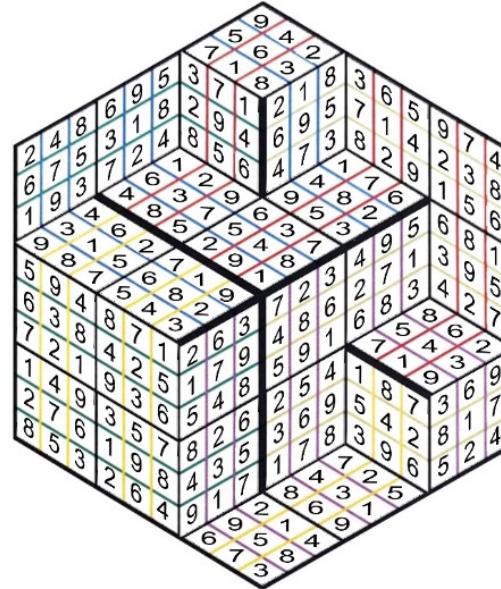
(a)



(b)



(c)



(d)

Figura 5.8: Configurațiile pentru care soluția a fost generată cel mai rapid, respectiv cel mai lent: a) imaginea 13; b) soluția, generată în 0.85s; c) imaginea 316; d) soluția, generată în 7.98s.