Universitatea Tehnică din Cluj-Napoca
Facultatea de Automatică și Calculatoare
Departamentul Calculatoare

# Detecția segmentelor de dreapta în imagini grayscale

Student: Roxana Rujac, grupa 30234

2025

# 1. Introducere

## 1.1 Descrierea problemei abordate

Detectarea segmentelor de dreaptă într-o imagine grayscale reprezintă o problemă esențială în procesarea imaginilor, având multiple aplicații în domenii precum viziunea computerizată, recunoașterea formelor și analiza imaginilor medicale sau satelitare. Scopul acestui proiect este de a implementa un algoritm care să identifice și să afișeze cele mai lungi N segmente de dreaptă dintr-o imagine grayscale utilizând transformata Hough și detectorul Canny.

## 1.2 Contextul problemei / Motivare / Utilitate

Identificarea structurilor liniare într-o imagine este crucială în numeroase aplicații, inclusiv în domeniul roboticii, recunoașterii caracterelor și reconstrucției 3D. Algoritmii tradiționali de detectare a marginilor, precum filtrul Sobel sau detectorul Canny, sunt eficienți în evidențierea contururilor, dar nu oferă informații despre segmentele de dreaptă complete dintr-o scenă. Pentru a obține aceste segmente, se utilizează transformata Hough, care permite detectarea liniilor chiar și în condiții de zgomot sau întreruperi parțiale ale contururilor.

Implementarea unui astfel de algoritm permite extragerea automată a structurilor liniare din imagini, ceea ce poate fi aplicat în:

- Navigația autonomă – detectarea marcajelor rutiere în sistemele de asistență pentru șoferi;
- Procesarea imaginilor medicale – identificarea structurilor anatomice în imagini de tip raze X sau tomografii;
- Cartografie și analiză geospațială – recunoașterea drumurilor și a limitelor clădirilor în imagini satelitare;
- Inspecție industrială – detectarea defectelor în structuri mecanice sau electronice.

# 2. Considerații teoretice

Acest proiect își propune implementarea unei metode eficiente pentru identificarea celor mai lungi N segmente de dreaptă într-o imagine grayscale, utilizând algoritmi bine cunoscuți din literatura de specialitate.

Pentru realizarea acestui obiectiv, se folosește **transformata Hough**, un instrument esențial în detectarea formelor geometrice, combinată cu tehnici avansate de preprocesare a imaginilor. Mai precis, se va aplica **filtrarea Gauss** pentru reducerea zgomotului, urmată de utilizarea **detectorului Canny** pentru evidențierea punctelor de muchie. Aceste metode, deja consacrate în analiza imaginilor, permit extragerea cu precizie a structurilor liniare dintr-o imagine.

## 2.1 Transformata Hough

Transformata Hough este o tehnică utilizată în procesarea imaginilor pentru a detecta forme geometrice, în special drepte, într-o imagine care conține un set de puncte de interes. Această metodă transformă problema detectării dreptelor într-o problemă de identificare a maximelor locale într-un spațiu parametric.

### Pașii algoritmului:

**Detectarea muchiilor în imagine**

- Se aplică un filtru pentru reducerea zgomotului (de exemplu, filtrul Gauss).
- Se folosește detectorul Canny pentru a identifica contururile obiectelor din imagine.

**Parametrizarea dreptelor**

- În loc de ecuația clasică a dreptei **y=ax+by**, se folosește o formulă specială care funcționează mai bine pentru detectarea liniilor:

$$\rho = x \cdot \cos(\theta) + y \cdot \sin(\theta)$$

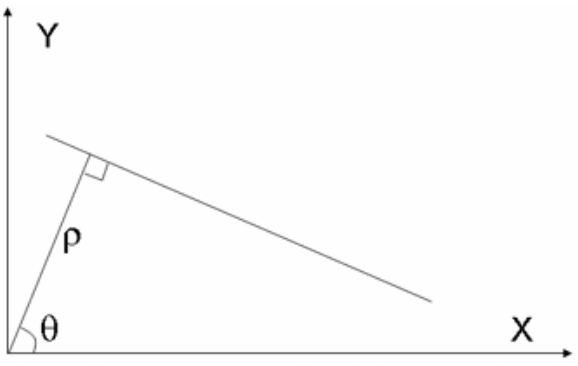



Fig. 1. Vectorul perpendicular pe dreaptă, ce trece prin origine, reprezintă dreapta prin parametrii ρ şi θ

- Aici, ρ este distanța de la origine până la dreaptă, iar θ este unghiul dreptei față de axa Ox.
- Se aleg intervale discrete pentru ρ și θ (ex: θ variază de la 0° la 180°, iar ρ ia valori între -R și R, unde R este diagonala imaginii).

**Construirea unui tabel (acumulator Hough)**

- Se creează o matrice 2D unde fiecare celulă corespunde unei perechi de valori (ρ,θ).
- Pentru fiecare punct de muchie detectat, se calculează toate dreptelor posibile care trec prin acel punct și se incrementează celula corespunzătoare în tabel.

**Identificarea celor mai proeminente drepte**

- Se caută valorile cele mai mari din tabel (maxime locale).
- Aceste valori corespund dreptelor cele mai evidente din imagine.

**Afișarea rezultatelor**

- Se trasează pe imagine doar cele mai semnificative drepte (de exemplu, cele mai lungi N drepte).

## 2.2 Detectorul Canny

Detectorul de muchii Canny este un algoritm popular utilizat în procesarea imaginilor pentru a detecta muchiile dintr-o imagine. O muchie reprezintă o tranziție bruscă în intensitatea pixelilor și este, adesea, o caracteristică semnificativă a unei imagini, indicând margini ale obiectelor sau detalii structurale.

### Pașii algoritmului:

- **Filtrarea imaginii cu un filtru Gaussian**: Elimină zgomotul din imagine prin aplicarea unui filtru Gaussian.

- **Calculul gradientului**: Se calculează modulul și direcția gradientului (punctele de maxim ale derivatei de ordin 1 a imaginii) pentru a identifica direcția și intensitatea schimbării în imagine.

Vom aproxima modului și direcția gradientului prin convoluția imaginii cu nucleul Sobel:

$$\nabla f_x = f(x,y) * \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad \text{Modul: } |\nabla f(x,y)| = \sqrt{(\nabla f_x(x,y))^2 + (\nabla f_y(x,y))^2}$$

$$\nabla f_y = f(x,y) * \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad \text{Direcție: } \theta(x,y) = arctg\left(\frac{\nabla f_y(x,y)}{\nabla f_x(x,y)}\right)$$

- **Suprimarea non-maximelor**: Elimină pixelii care nu sunt maxime locale în direcția gradientului.

  Are drept scop subțierea muchiilor prin păstrarea doar a punctelor de muchie care au modulul maxim al gradientului de-a lungul direcției de variație a intensității (de-a lungul direcției gradientului).

- **Binarizarea adaptivă**: Se aplică un prag adaptiv pentru a decide care pixeli sunt muchii, compensând variațiile de iluminare.

- **Prelungirea muchiilor prin histereză**: Se conectează muchiile slabe la cele tari, eliminând muchiile false.

## 2.3 Filtrul Gaussian trece-jos

Un filtru Gauss (sau kernel Gauss) este un tip de filtru utilizat pentru a aplica blurring sau o netezire asupra imaginii, folosind funcția de distribuție normală (Gauss).

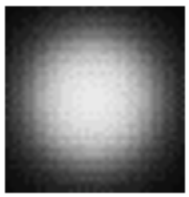
Filtrul Gaussian aplică o funcție de convoluție bazată pe distribuția normală, astfel încât valorile mai apropiate de centrul kernelului au un impact mai mare asupra pixelilor din jur, iar valorile aflate la marginea kernelului au un impact mai mic.

Funcția de distribuție Gaussiană într-o dimensiune 2D este dată de formula:

$$p(f) = \frac{1}{(2\pi\langle f\rangle)^{1/2}} \exp\left(\frac{-(f-\langle f\rangle)^2}{2\langle f\rangle}\right)$$

**Pașii algoritmului:**

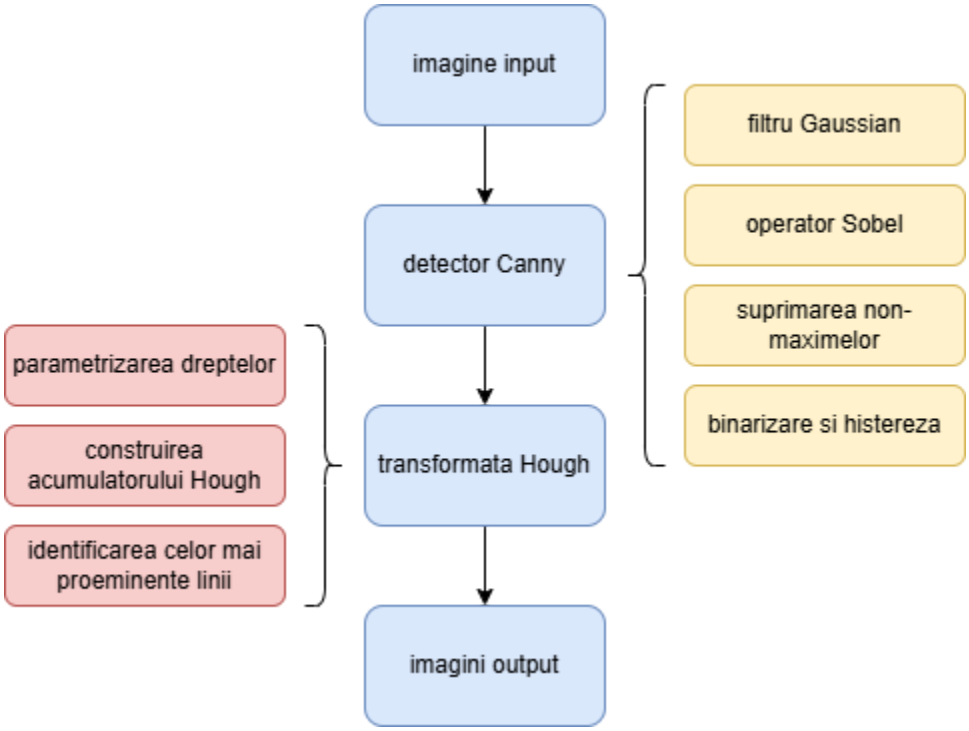
- **Crearea kernelului Gaussian** pe baza funcției de distribuție normală.
- **Aplicarea paddingului** la marginea imaginii pentru a permite convoluția pe toți pixelii.
- **Convoluția** imaginii cu kernelul Gaussian pentru fiecare pixel.
- **Înlocuirea pixelilor** originali cu valorile calculate din convoluție.
- **Obținerea imaginii filtrate**, care va arăta mai estompată (blurred).



filtru gaussian

# 3. Specificații de implementare

## 3. 1 Descrierea metodei utilizate




### 1. Filtrare Gauss

Aplicăm un kernel 5x5 pentru a netezi imaginea și a reduce zgomotul, pas esențial pentru a evita detecția de margini false.

### 2. Detecția marginilor cu operatorul Sobel

Se aplică două kerneluri 3x3 pentru a calcula gradientul în direcțiile X și Y. Acestea ne ajută să determinăm magnitudinea și direcția gradientului pentru fiecare pixel.

### 3. Suprimare non-maximă

Pixelii care nu reprezintă un maxim local în direcția gradientului sunt eliminați. Aceasta îngustează contururile la o lățime de un pixel.

### 4. Pragul dublu și histerezis

Se aplică două praguri:

- *High Threshold*: pentru a identifica contururi puternice.

- *Low Threshold*: pentru a verifica vecinii contururilor slabe și a decide dacă ele aparțin unei linii reale.

### 5. Construirea acumulatorului

- Se creează o matrice 2D cu dimensiunile [360° × lungimea_diagonalei]
- Pentru fiecare pixel de muchie (x,y) detectat de Canny:
    - Se calculează $\rho = x \cdot \cos(\theta) + y \cdot \sin(\theta)$ pentru $\theta \in [0°, 359°]$
    - Se incrementează acumulatorul[θ, ρ]

### 6. Detectarea maximelor locale

- Se aplică o fereastră de dimensiunea windowSize × windowSize (7×7 implicit)
- Se identifică valorile care sunt simultan:
    - Maxime locale în fereastra dată
    - Peste pragul minim de voturi (threshold = 10)
- Liniile se sortează descrescător după numărul de voturi

## 7. Afișarea rezultatelor

- Se desenează liniile detectate în verde pe imaginea originală
- Pixelii din Canny care contribuie la linii se colorează în magenta
- Se afișează acumulatorul Hough cu maxime marcate prin cruci roșii
- Se selectează și afișează doar primele N linii cu cele mai multe voturi

## 3. 2 Concepte tehnice

**Pseudocod relevant pentru modul in care functioneaza programul:**

```
function CannyEdgeDetection(image):
    blurred = GaussianFilter(image)
    gradX, gradY, magnitude = Sobel(blurred)
    suppressed = NonMaxSuppression(magnitude, gradX, gradY)
    edges = HysteresisThreshold(suppressed, high, low)
  return edges


function HoughTransform(edges):
    for each edge pixel (x, y):
        for theta = 0 to 180:
            rho = x * cos(theta) + y * sin(theta)
            accumulator[rho, theta] += 1
    return accumulator
```

### 3.3 Ghid de utilizare

**Input**: imagine în format .bmp (grayscale).

**Parametri**:

- p – proporția din 255 pentru pragul superior.
- k – coeficient pentru pragul inferior.
- window size = 7 - dimensiunea ferestrei pentru detectarea maximelor locale
- threshold = 10 - numărul minim de voturi pentru o linie validă
- N - numărul de linii de afișat (introdus de utilizator)

**Executare**:

- Se rulează programul, se încarcă imaginea, se detectează marginile, apoi se va aplica transformata Hough pentru detecția liniilor.

**Output**:

- Fereastră cu imaginea originală.
- Fereastră cu marginile detectate.
- Imagine cu liniile detectate trasate.
- Fereastra cu acumulatorul Hough

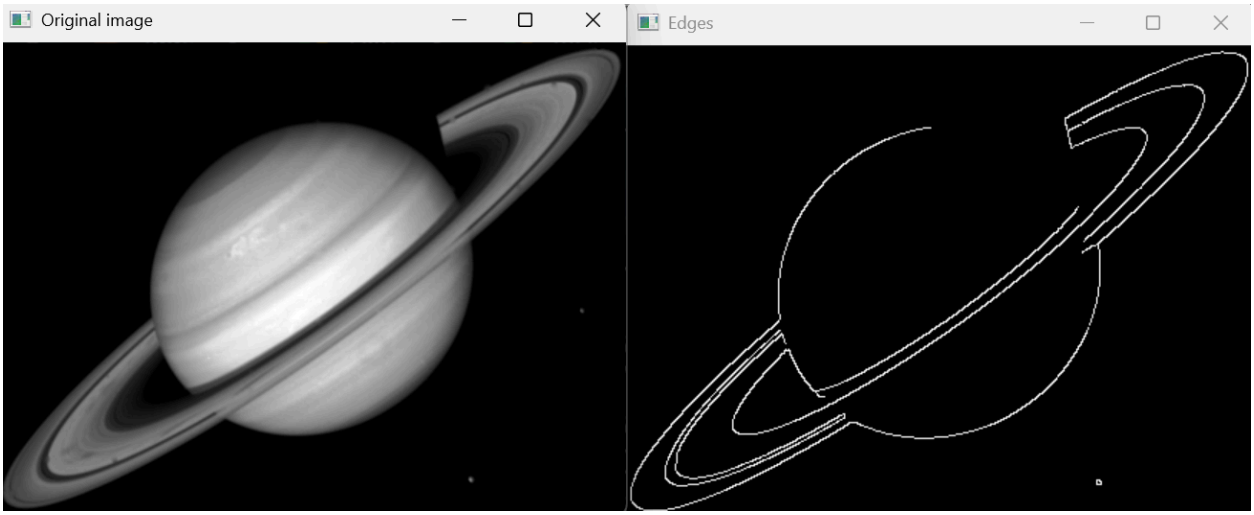
Utilizatorul trebuie sa aleagă o imagine drept input prin intermediul unei ferestre de dialog ce apare atunci cand rulam programul. Acesta poate naviga în orice folder din interiorul proiectului pentru a selecta imaginea dorită.

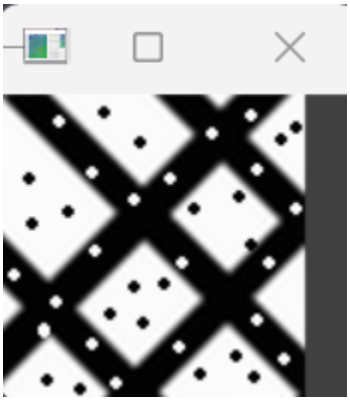
Următorul pas este introducerea unui număr N. Acesta reprezinta cate segmente se vor afisa din toate cele găsite. Segmentele se ordoneaza descrescator, astfel incat se vor afișa în final cele mai lungi N segmente găsite.

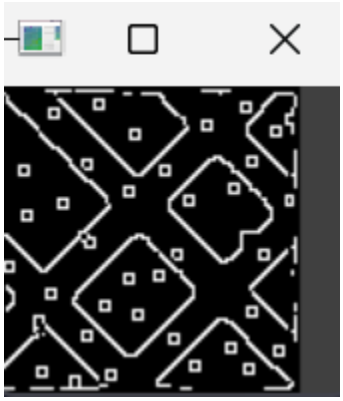
După introducerea parametrului N, se va afișa output-ul, ca 4 imagini separate.

# 4. Rezultate experimentale

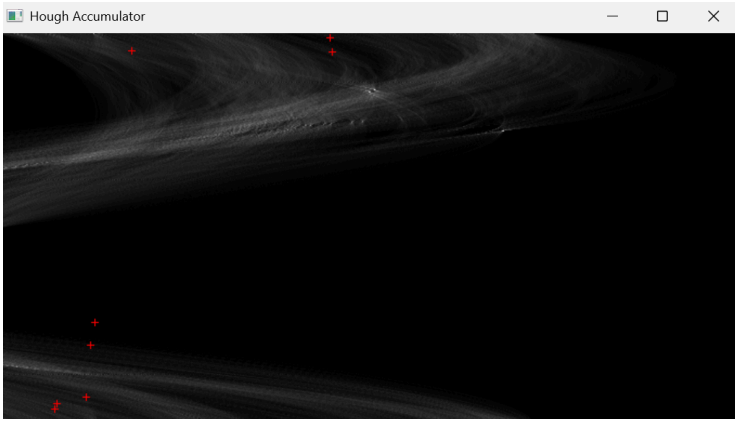
Rezultate Canny:

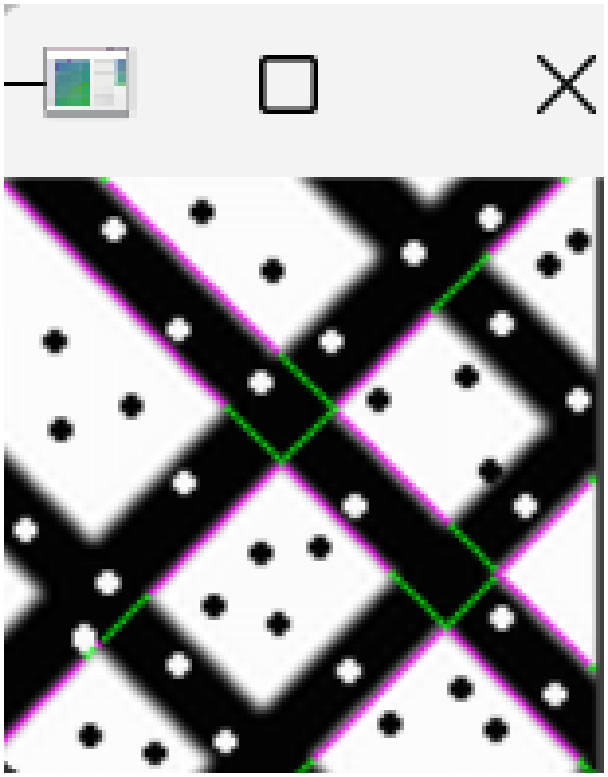



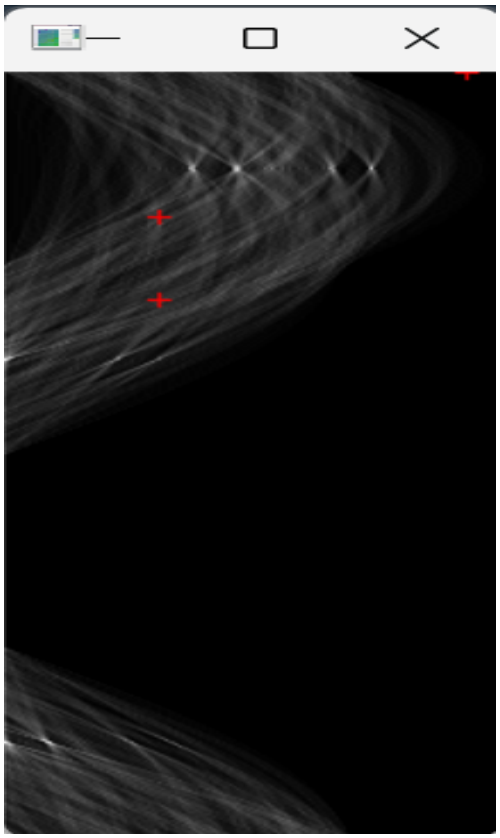





Rezultate Hough Transform:



## 5. Concluzii

Proiectul este implementarea cu succes un sistem complet de detectare a segmentelor de dreaptă în imagini grayscale, combinând eficient detectorul Canny cu transformata Hough. Toate obiectivele propuse au fost îndeplinite, demonstrând capacitatea algoritmului de a identifica și selecta cele mai semnificative N linii dintr-o imagine. Rezultatele experimentale confirmă eficiența metodei pentru diverse tipuri de imagini, de la scene rutiere la structuri geometrice complexe. Implementarea oferă o bază solidă pentru aplicații practice în domenii precum navigația autonomă, procesarea imaginilor medicale și analiza geospațială.

# Bibliografie


1. https://users.utcluj.ro/~vcristian/PI.html
2. https://users.utcluj.ro/~rdanescu/pi_c09.pdf#page=18
3. https://docs.opencv.org/4.x/dc/dd2/classcv_1_1LineIterator.html
4. https://docs.opencv.org/4.x/index.html
5. https://stackoverflow.com/questions/39114989/how-to-find-a-line-from-polar-coordinates-hough-transform-confusion