Edge Detection: Algoritmul Canny

Miu Elena Adania Grupa 333

Alexandrescu Tudor Alexandru Grupa 464

Cuprins

1. Introducere

1.1 Ce este Edge Detection?

Edge detection poate fi definit drept descoperirea liniilor care delimitează obiectele (componentele) de interes din imagine.

Edge detection implică folosirea unei varietăți de metode matematice care au scopul de a identifica contururi în imagini. Contururile (edges) sunt zone în imagini care prezintă modificări bruște de contrast, cauzate de schimbări în lumină, culoare, umbră, textură. Aceste modificări pot fi folosite pentru a determina proprietăţile unei imagini şi pot fi localizate pe ambele axe ale imaginii.

Analizarea digitală a unei imagini ajută la suprimarea informaţiilor irelevante, pentru a putea găsii contururile. În edge detection, conturul poate fi confundat cu zgomot, ceea ce poate fi controlat prin modul în care setăm parametrii de filtrare. Pentru modificări foarte fine, este dificil şi durează mult să detectăm conturul, mai ales dacă imaginea conţine mult zgomot.

Edge detection implică, mai degrabă, o colecție de algoritmi, și nu un algoritm de sine stătător.

Edge detection nu este un silver bullet, parametrizarea realizându-se de la caz la caz.

1.2 Exemplu de Edge Detection

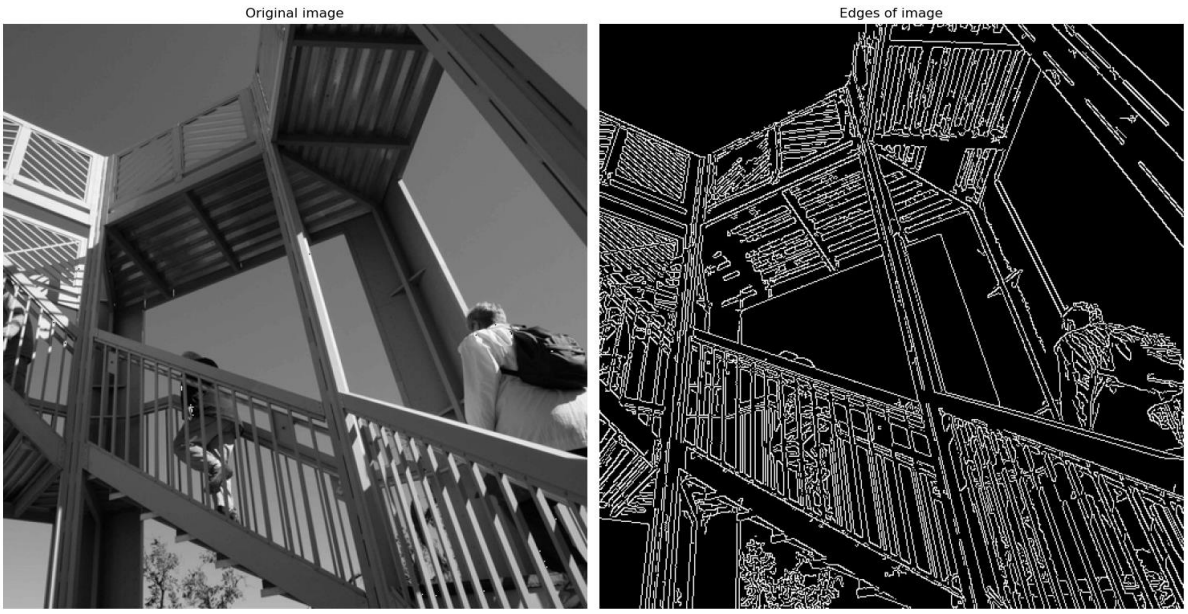


Figura 1: Edge detection utilizând algoritumul Canny

Sursa: Calcule ale autorilor folosind imaginea ascent din biblioteca scipy și aplicând cv2.Canny

1.3 Tipuri de Margini în Imagini

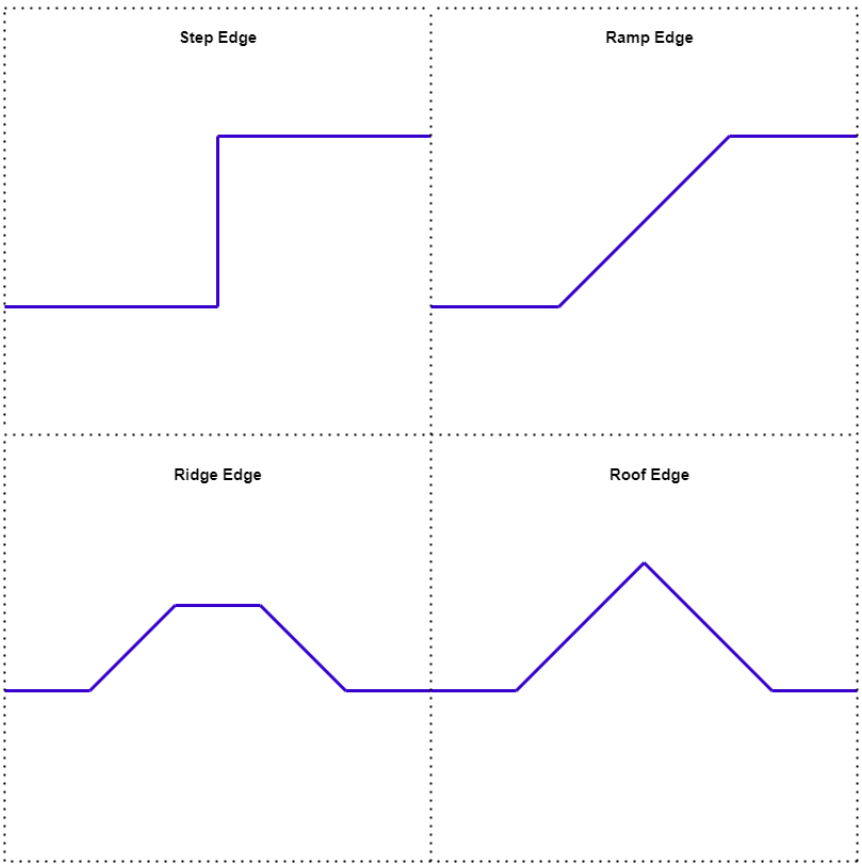


Figura 2: Tipuri de margini

Sursa: Imagine creată de autori folosind drawio.com

1. Step edge

Un step edge se formează când există o schimbare bruscă în intensitatea pixelilor.

Precum sugerează şi numele, graficul arată ca o treaptă a unei scări – există o urcare bruscă în grafic, care indică o schimbare în valorile pixelilor. Aceste tipuri de margini sunt mai uşor de identificat.

2. Ramp edge

Un ramp edge este ca un step edge, doar că schimbarea nu mai este instantanee. în schimb, schimbarea în valorile pixelilor are loc pe o distanţă scurtă şi finită.

3. Ridge edge

Un ridge edge combină două ramp edges, puse una împotriva celeilalte.

Aceasta apare când intensitatea se schimbă brusc, iar apoi revine la valoarea iniţială după o distanţă scurtă, finită.

4. Roof edge

Roof edge este un tip de ridge edge, doar că revenirea la valoarea iniţială se face mult mai rapid.

2. Justificare și Aplicații Practice

Edge detection reprezintă o funcționalitate esențială în procesarea imaginilor, precum și în machine vision și computer vision. Joacă un rol deosebit în feature detection și feature extraction.

Aplicaţii şi algoritmi de edge detection sunt folosite în diverse industrii. Ne vom uita la câteva exemple unde este utilizat edge detection în practică.

1. Medical Imaging

Utilizarea inteligenţei artificiale este un mod de a revoluţiona sistemul medical. Edge detection este folosit pentru a detecta diferite boli şi afecţiuni. De exemplu, detectarea anomaliilor din corp, cum ar fi tumorile. Edge detection evidenţiază caracteristicile imaginii, astfel încât doctorii să poată identifica diferenţele.

2. Fingerprint Recognition

Conturul din acest caz constă în liniile fine ale amprentei. Edge detection poate identifica aceste linii, făcând recunoaşterea amprentei posibilă.

3. Recunoaşterea vehiculelor

Tehnologia maşinilor self-drive este relativ nouă şi încă evoluează. Pentru ca auto-vehiculele să se conducă singure, ele trebuie să fie capabile să identifice alte auto-vehicule, vehicule, oameni, semne de circulaţie şi alte obiecte.

4. Satellite Imaging

Edge detection mai este folosit şi pentru procesarea imaginilor din satelit, pentru a reduce zgomotul, astfel făcând extragerea structurii regionale mai uşoară.

5. Robotic vision

Această aplicaţie este similară cu cea a maşinilor self-drive, unde este folosită identificarea obiectelor din imaginile unei camere.

3. Abordarea Tehnică

Pașii Algoritmului Canny pentru Edge Detection:

1. Transformarea imaginii în format gray-scale
2. Reducerea zgomotului din imagine
3. Calcularea gradientului și a direcției
4. Suprimarea non-maximelor
5. Hysteresis thresholding

3.1 Transformarea Imaginii în format Gray-Scale

Convertim imaginea în format gray-scale deoarece culorile sunt irelevante în detectarea contururilor.

3.2 Reducerea Zgomotului din Imagine

Urmatorul pas este de a reduce frecvențele înalte din imagini (noise). Putem aplica algoritmul Gaussian smoothing, care este un low pass filter. Acesta netezeşte imaginea şi reduce impactul variaţiilor mici în intensitatea pixelilor.

Facem acest pas deoarece rezultatul dorit trebuie să identifice doar contururile, şi nu ne interesează să vedem toate detaliile din imagine.

Aplicarea filtrului Gaussian smoothing se poate face prin două metode:

- Aplicând operaţia de convoluție între imagine și kernel
- Efectuând produsul elementelor în domeniul frecvenței

Filtrarea frecvențelor este lină (nu bruscă) deoarece ne bazăm pe forma distribuției normale și depinde de parametrul sigma.

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

Formula 1: Funcția de smoothing gaussiană

3.3 Calcularea Gradientului

Până acum am discutat despre partea de pre-procesare a imaginii, acum începe de fapt edge detection.

Metodele de detecţie a contururilor pot fi împarţite în două categorii:

- Bazate pe Gradient
- Bazate pe Laplacian

În metoda calculării gradientului, contururile sunt detectate cu ajutorul primei derivate a imaginii. Magnitudinea gradientului este folosită pentru a măsura intensitatea conturului.

În metoda bazată pe Laplacian, este folosită derivata a doua care face zero crossing (se intersectează cu axa Ox). În general, contururile se găsesc prin căutarea unei expresii diferenţiale neliniare care face zero-crossing.

Calcularea schimbărilor de intensitate la nivelul imaginii este echivalentă cu calcularea variației unei funcții de doi parametrii. Aceasta se poate face cu ajutorul gradientului.

Prima derivată a imaginii este folosită pentru a găsi valorile maxime şi minime în gradient.

$$\frac{\partial I}{\partial x}(x,y) = I(x+1,y) - I(x-1,y)$$
$$\frac{\partial I}{\partial x}(x,y) = I(x+1,y) - I(x-1,y) \quad (2)$$

Formula 2: Gradientul unui pixel

Pentru a calcula aproximările gradientului vom aplica peste imagine câte un filtru pentru fiecare axă, astfel vom obţine schimbările de pe orizontală, respectiv verticală.

$$G_x = h_x * A$$
$$G_y = h_y * A \quad (3)$$

Formula 3: Aproximările Gradientului
,unde h este filtrul (kernel), $*$ este operația de convoluție,
iar A este imaginea originală

Există mai mulţi operatori care pot fi folosiţi drept filtru: Operatorul Sobel, Operatorul Prewitt, Operatorul Roberts, Operatorul Scharr etc.

1. Operatorul Roberts

Acest operator constă în două matrici kernel de dimensiune 2x2.

$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \qquad (4)$$

Formula 4: hx și hy la Operatorul Roberts

Operatorul Roberts detectează uşor marginile şi orientarea, şi punctele de pe diagonală sunt păstrate. Însă, este foarte sensibil la zgomot, şi nu produce rezultate precise.

2. Operatorul Sobel

Acest operator constă în două matrici kernel de dimensiune 3x3.

$$\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \qquad (5)$$

Formula 5: hx și hy la Operatorul Sobel

Operatorul Sobel este foarte simplu de utilizat, timpul computaţional este scăzut, şi detectează uşor muchii netede. Însă, punctele de pe diagonală nu sunt mereu păstrate, este foarte sensibil la zgomot, şi detectează contururi groase şi subţiri, ceea ce nu rezultă în rezultate potrivite.

3. Operatorul Prewitt

Acest operator constă în două matrici kernel de dimensiune 3x3.

$$\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} \qquad (6)$$

Formula 6: hx și hy la Operatorul Prewitt

Operatorul Prewitt are performanţă bună în detectarea contururilor vericale şi orizontale, şi este considerat cel mai bun operator pentru a determina orientarea imaginii.

Însă, coeficientul de magnitudine este fix, şi nu poate fi schimbat, şi punctele de pe diagonală nu sunt mereu păstrate.

4. Operatorul Scharr

Acest operator constă în două matrici kernel de dimensiune 3x3.

$$\begin{pmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{pmatrix} \begin{pmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{pmatrix} \qquad (7)$$

Formula 7: hx și hy la Operatorul Scharr

Operatorul Scharr încearcă să obţină simetria raţională perfectă.

După ce calculăm aproximările pentru fiecare axă, le folosim pentru a găsi magnitudinea gradientului şi direcţia gradientului.

$$G = \sqrt{G_x^2 + G_y^2} \qquad (8)$$

Formula 7: Magnitudinea Gradientului

$$\Theta = \arctan\left(\frac{G_y}{G_x}\right) \qquad (9)$$

Formula 8: Direcția Gradientului

Pentru a combate dezavantajele operatorilor clasici, algoritmul Canny adaugă următorii doi paşi.

3.4 Suprimarea Non-Maximelor

Suprimarea non-maximelor reprezintă subțierea contururilor de-a lungul direcției gradientului. Echivalentul său în procesarea semnalelor este: high-pass filter.

Procesul presupune compararea fiecarui pixel cu doi pixeli vecini de pe direcţia gradientului, şi verifică dacă acesta reprezintă un maxim local. Dacă un pixel este maxim local, se păstrează pentru pasul următor, altfel este eliminat prin atribuirea valorii 0.

3.5 Hysteresis thresholding

Hysteresis thresholding este utilizat pentru a identifica muchii puternice şi slabe în imagine. Reprezintă unirea punctelor de muchie de pe un contur (defragmentarea conturului).

Stabilim două praguri, min_threshold și max_threshold. Filtrăm pixelii cu un gradient slab (min_threshold) și păstrăm pixelii cu un gradient mare (max_threshold). Pixelii cu o intensitate a gradientului mai mare decât max_threshold fac parte sigur din contur şi le atribuim valoarea 255, iar pixelii cu o valoare sub min_threshold sunt eliminați prin atribuirea valorii 0. Pixelii cu valoarea cuprinsă între min_threshold și max_threshold sunt considerați parte din contur dacă sunt conectați direct cu un pixel care este sigur în contur. Altfel, aceștia sunt eliminați.

5. Tehnologii Folosite şi Rezultate

Partea de cod a proiectului a fost implementată în Python 3.11 utilizând următoarele librării:

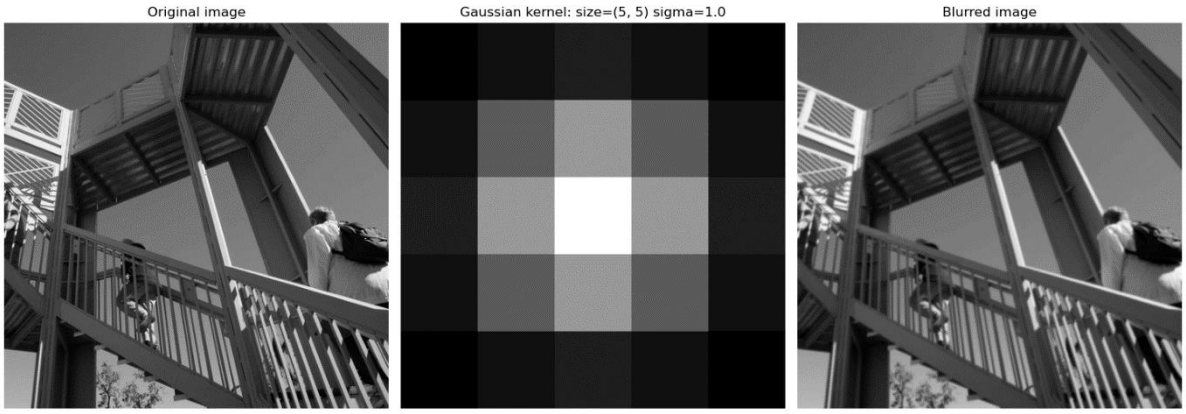
- Numpy și Scipy pentru calcul computațional
- Matplotlib – pentru plotare
- OpenCV pentru a verifica rezultatele noastre cu implementările consacrate

5.1 Implementarea Algoritmului

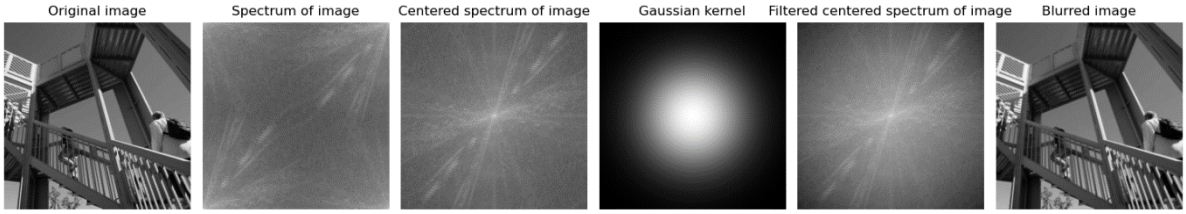
1. Reducerea zgomotului din imagine cu filtrul Gaussian

Am implementat metoda care aplică operaţia de convoluţie între imagine şi kernel, pe care am comparat-o cu metoda produsului în domeniul frecvenţei, şi cu metoda din opencv.

Pentru filtrul Gaussian prin convoluţie şi opencv, am ales un kernel de 5x5 şi sigma 1.0:



Filtrul Gaussian prin evaluarea produsului în domeniul frecvenţei:

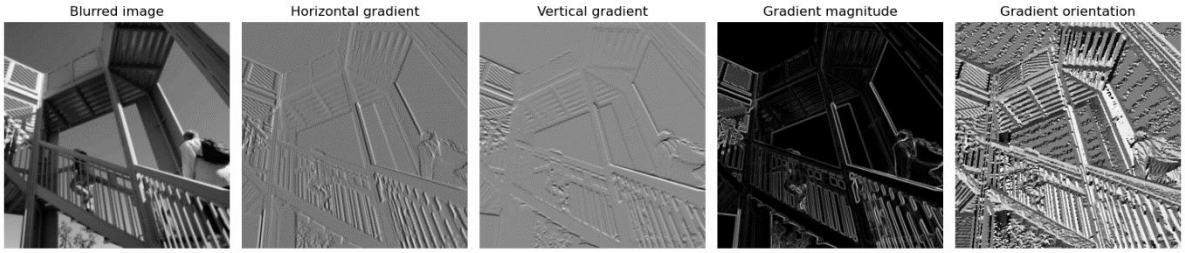


Comparăm cele trei metode pentru filtrul Gaussian:

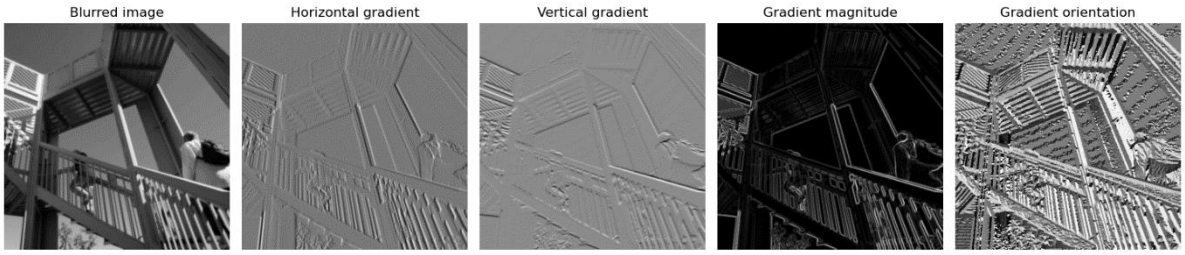
2. Calcularea Gradientului

Pentru calcularea gradientului am aplicat trei operatori pentru a putea fi comparaţi: Sobel, Prewitt şi Scharr.

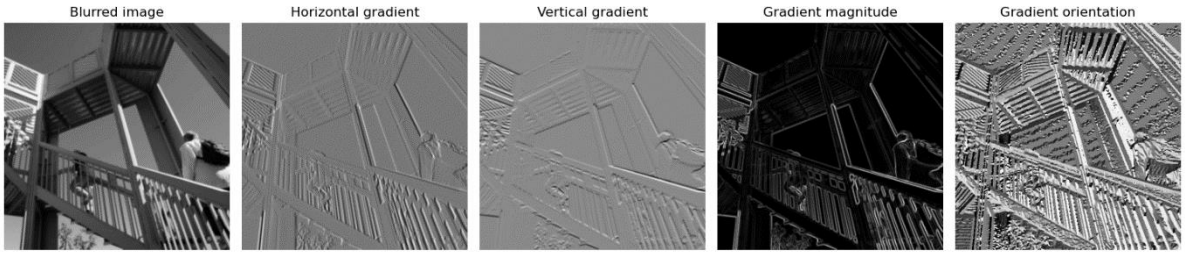
Operatorul Sobel



Operatorul Prewitt



Operatorul Scharr



3. Suprimarea Non-Maximelor

Matricea de orientare a gradientului conţine unghiuri măsurate în radiani. Valorile pot fi convertite în grade, însă noi am decis să lucrăm direct în radiani.

Fiecare pixel care nu se află pe marginea imaginii are patru vecini direcţi: la est, vest, nord şi sud. Aşadar, avem de luat în considerare patru direcţii: orizontală, verticală şi cele două diagonale ale imaginii.

Indiferent dacă orientarea gradientului este de la nord la sud sau invers, vom avea aceiaşi vecini de verificat. Astfel, axa Oy negativă poate fi ignorată (adunăm pi=3.14159 la unghiurile cu valoare negativă).

Întrucât matricea de orientare a gradientului conţine foarte multe valori care nu corespund cu cele patru direcţii din matricea de magnitudine a gradientului, cercul trigonometric trebuie să fie discretizat în regiuni în funcţie de orientarea gradientului.

Se vor forma următoarele regiuni:

- regiune centrată în 0pi radiani
- regiune centrată în 0.25pi radiani
- regiune centrată în 0.5pi radiani
- regiune centrată în 0.75pi radiani
- regiune centrată în pi radiani

Echivalent, putem obţine intervale pentru cele patru regiuni utilizând paşi de incrementare pentru 0.125pi radiani:

- pentru valori între 0pi radiani şi 0.125pi radiani -> regiunea centrată în 0pi radiani
- pentru valori între 0.125pi radiani şi 0.375pi radiani -> regiunea centrată în 0.25pi radiani
- pentru valori între 0.375pi radiani şi 0.625pi radiani -> regiunea centrată în 0.5pi radiani
- pentru valori între 0.625pi radiani şi 0.875pi radiani -> regiunea centrată în 0.75pi radiani
- pentru valori între 0.875pi radiani şi pi radiani -> regiunea centrată în pi radiani

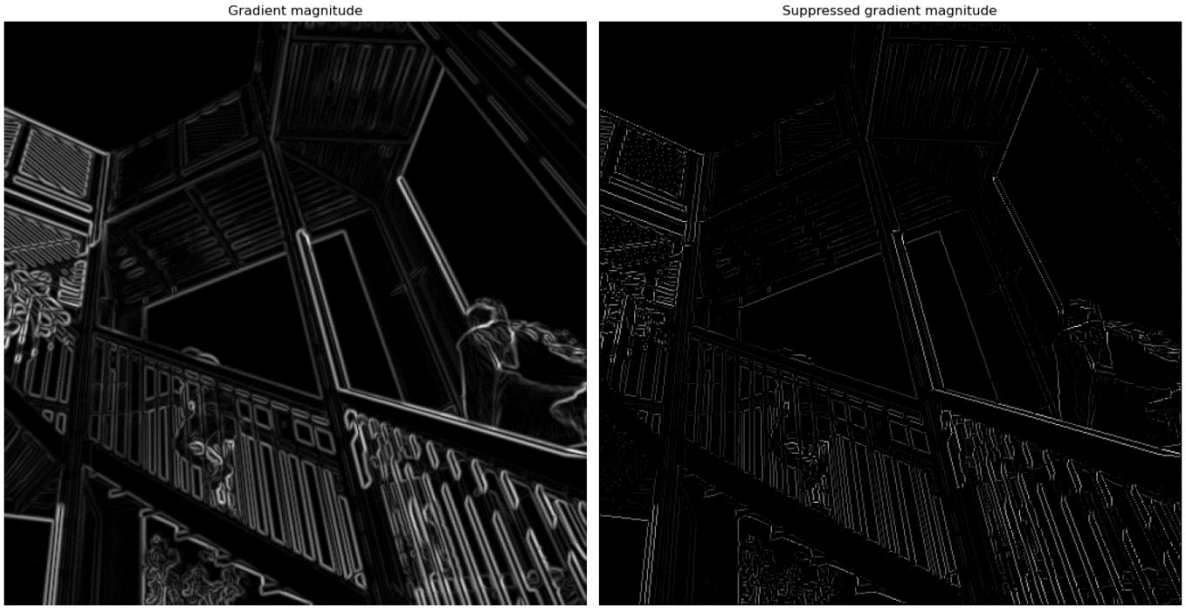
Se verifică pentru fiecare pixel din matricea de magnitudine a gradientului, dacă este maxim local între cei doi vecini de pe direcţia gradientului.

Am aplicat acest pas pe rezultatele de la pasul anterior pentru fiecare operator.

Suprimarea Non-Maximelor cu Operatorul Sobel

Suprimarea Non-Maximelor cu Operatorul Prewitt

Suprimarea Non-Maximelor cu Operatorul Scharr



4. Hysteresis Thresholding şi unirea muchiilor

Pentru acest pas am ales pragurile min_threshold = 5.0 şi max_threshold = 15.0, pe baza cărora filtrăm pixelii cu valori mici.

Avem trei tipuri de pixeli:

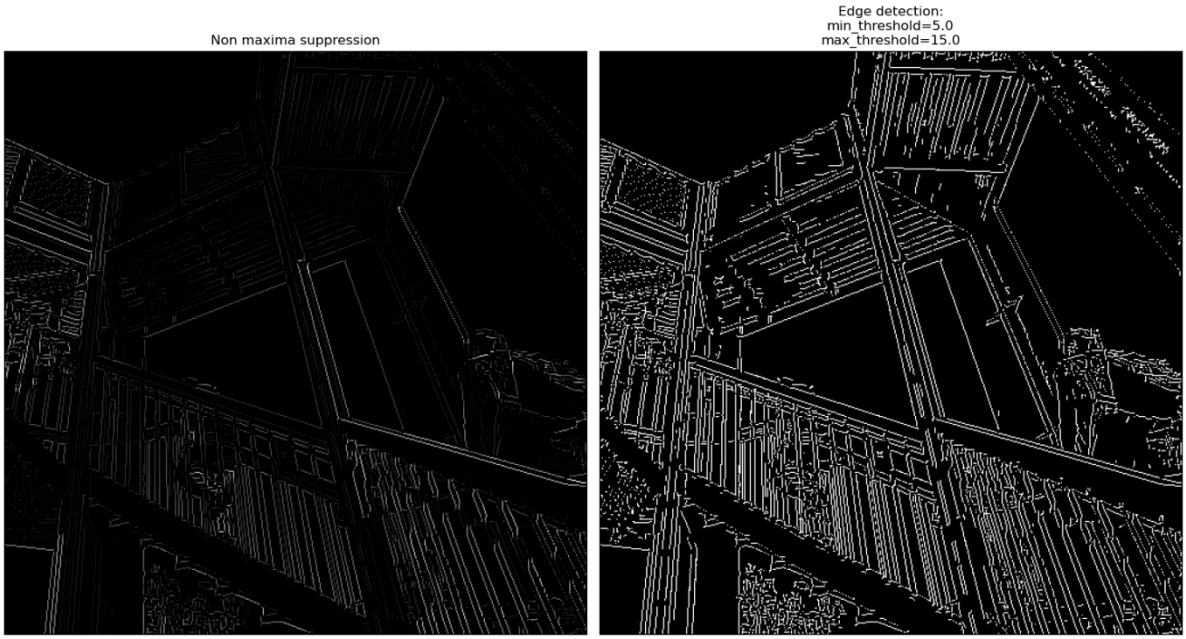
- non muchie - pixelii cu valoare mai mică decât min_threshold, aceştia sunt eliminaţi
- strong - pixelii cu valoare mai mare decat max_threshold, aceştia fac sigur parte din contur
- weak - pixelii au valori cuprinse între cele două praguri

Considerăm un pixel weak ca parte din contur doar dacă este conectat direct de un pixel strong, dintre cei opt vecini ai săi.

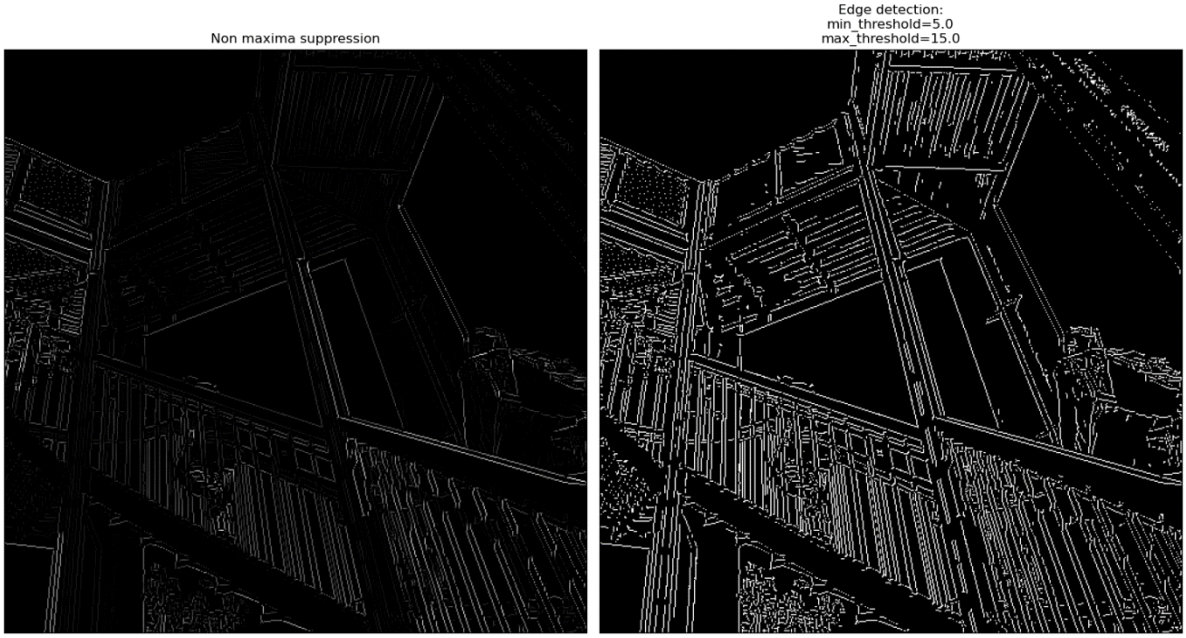
Am aplicat acest pas pe rezultatele de la pasul anterior, pentru fiecare operator.

Edge Detection Operatorul Sobel

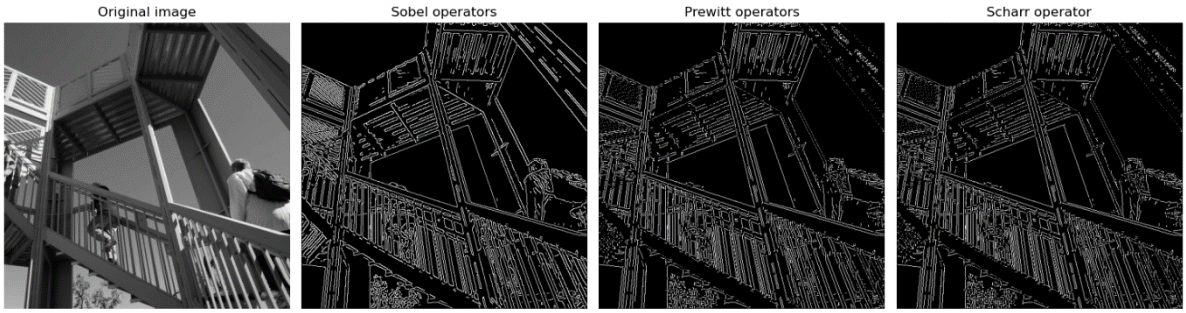
Edge Detection Operatorul Prewitt



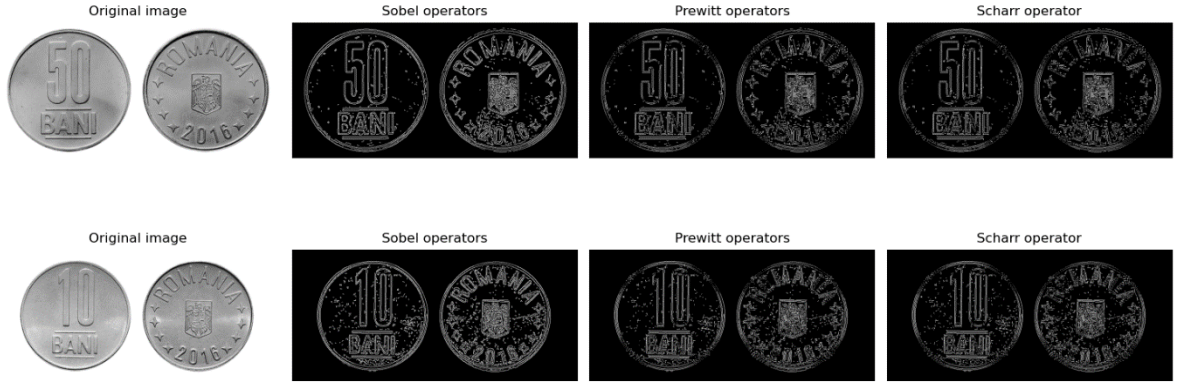
Edge Detection Operatorul Scharr



Comparăm cele trei rezultate cu imaginea originală:



Am aplicat algoritmul nostru şi pe alte două exemple:



Putem observa că pentru exemplele noastre, Algoritmul Canny a returnat cele mai bune rezultate folosind Operatorul Sobel.

6. Concluzii

Edge detection este utilizat pentru a detecta conturul elementelor de interes din imagini. Acesta este frecvent utilizat în practică, la probleme de feature detection și feature extraction. Este foarte util pentru diverse industrii, de la domeniul medical până la securitate şi computer vision.

Există mai mulţi algoritmi, însă algoritmul Canny este cel mai cunoscut, şi cel mai utilizat algoritm de edge detection. El reprezintă mai degrabă o colecţie de algoritmi parametrizabili, astfel poate fi personalizat în funcţie de preferinţe, de datele de input, sau de scopul final.

Bibliografie

1. Machine Vision by E. R. Davies
2. Ziou, Djemel & Tabbone, Salvatore. (1998). 'Edge detection techniques: An overview'. International Journal of Patttern Recognition and Image Analysis. 4. 537-559.
3. Mutneja, Vikram. (2015). Methods of Image Edge Detection: A Review. Journal of Electrical & Electronic Systems. 04. 10.4172/2332-0796.1000150.
4. Adrian Rosebrock. (2021). OpenCV Edge Detection (cv2.Canny)
5. Rabia Gül. What Are The Applications of Edge Detection?
6. Rohit Krishna. (2023). Coding Canny Edge Detection Algorithm from scratch in Python.
7. Shrivakshan, G.T. & Chandrasekar, Chandramouli. (2012). A Comparison of various Edge Detection Techniques used in Image Processing. International Journal of Computer Science Issues. 9. 269-276.
8. Kishor Kumar, Dr Vijaya Lakashmi. (2022). An Efficient Implementation Of Edge Detection Algorithm For Image Processing Using Fpga. Journal of Positive School Psychology Vol. 6, No. 10, 3110-3119.
9. Imaginile cu monede au fost descărcate de pe site-ul BNR.
10. Formulele au fost generate în LaTeX, proiectul se află în docs/formule latex.
11. Link Proiect GitHub: Alexandrescu Tudor Alexandru & Miu Elena Adania